

Ioan Dumitrache (Ed.)

Advances in Intelligent Control Systems and Computer Science

 Springer

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Ioan Dumitrache (Ed.)

Advances in Intelligent Control Systems and Computer Science

 Springer

Editor
Ioan Dumitrache
Control Engineering and
Computer Science Faculty
University "POLITEHNICA" of Bucharest
Bucharest
Romania

ISSN 2194-5357

e-ISSN 2194-5365

ISBN 978-3-642-32547-2

e-ISBN 978-3-642-32548-9

DOI 10.1007/978-3-642-32548-9

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012944262

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

On the Use of Stochastic Complexity in Spectral Analysis of Radial Velocity Data	1
<i>Vili Forsell, Ciprian Doru Giurcăneanu</i>	
On an Input Driven Hierarchy of Hybrid Automata	15
<i>Virginia Ecaterina Oltean, Radu Dobrescu, Dan Popescu</i>	
Improvement of Statistical and Fractal Features for Texture Classification	31
<i>Dan Popescu, Radu Dobrescu, Nicoleta Angelescu</i>	
Towards a PIO II Criterion: Improving the Pilot Modeling	45
<i>Adrian Toader, Ioan Ursu</i>	
Human Skin Detection Using Texture Information and Vector Processing Techniques by Neural Networks	59
<i>C.M. Dumitrescu, Ioan Dumitrache</i>	
MDL Based Structure Selection of Union of Ellipse Models for Scaled and Smoothed Histological Images	77
<i>Jenni Hukkanen, Edmond Sabo, Ioan Tabus</i>	
Retinopathy of Prematurity: Fractal Analysis of Images in Different Stages of the Disease	91
<i>Nebojša T. Milošević, Maja Olujić, Ana Oros, Herbert F. Jelinek</i>	
Time Series Analysis of Blood Flow in Kidney Graft: On the Emergence of Deterministic Chaos during the Phase of Acute Rejection	103
<i>Przemysław Waliszewski</i>	

Nonlinear Model Based Predictive Control of Visual Servoing Systems Using Image Moments	115
<i>Cosmin Copot, Corneliu Lazar, Adrian Burlacu</i>	
Mobile Robot Navigation Using Graph Search Techniques over an Approximate Cell Decomposition of the Free Space	129
<i>Radu Robotin, Gheorghe Lazea, Petru Dobra</i>	
Stability Analysis Software Platform Dedicated for a Hexapod Robot ...	143
<i>Sorin Mănoiu-Olaru, Mircea Nițulescu</i>	
A Comparison of Adaptive Supervisory Switching Control Schemes for High Maneuverability Aircrafts	157
<i>Andrei-Sorin Neamtu, Adrian-Mihail Stoica</i>	
Box-Counting and Multifractal Analysis in Neuronal and Glial Classification	177
<i>Herbert F. Jelínek, Nebojša T. Milošević, Audrey Karperien, Bojana Krstonošić</i>	
Comparison between Titan and Cobalt Hydroxyapatite-Coated Shoulder Prosthesis, during External and Internal Rotation	191
<i>Mihaela Manisor, Cosmin Marcu, Gheorghe Tomoaia, Liviu Miclea</i>	
Artificial Emotion Simulation Model and Agent Architecture: Extended	207
<i>Valentin Lungu</i>	
Remote Monitoring and Control System for Environment Applications	223
<i>Alexandru Dumitrașcu, Dan Ștefănoiu, Janetta Culiță</i>	
H_{∞} Control of an Induction Heating Inverter	235
<i>Tibor Szelitzky, Eva-Henrietta Dulf</i>	
Forecasting Energy Consumption in Dwellings	251
<i>Nicoleta Arghira, Stéphane Ploix, Ioana Făgărășan, Sergiu Stelian Iliescu</i>	
Modelling and Composed Recursive Model Free Control for the Anaerobic Digestion Process	265
<i>Haoping Wang, Boyko Kalchev, Yang Tian, Ivan Simeonov, Nicolai Christov</i>	
Grid Based Hydrologic Model Calibration and Execution	279
<i>Danut Mihon, Victor Bacu, Denisa Rodila, Teodor Stefanut, Karim Abbaspour, Elham Rouholahnejad, Dorian Gorgan</i>	

Composite Application for Water Resource Management	295
<i>Mariana Mocanu, Marian Muste, Vasile Lungu, Radu Drobot</i>	
Energy Efficiency Dependency Analysis for a Data Center	307
<i>Iulia Dumitru, Stéphane Ploix, Ioana Făgărășan, Sergiu Stelian Iliescu</i>	
Intelligent Building Management: A Service-Oriented Approach	323
<i>Catalin Chera, Laurentiu Bucur, Serban Petrescu</i>	
Intelligent Forecasting of Indoors Ecological Processes	339
<i>Janetta Culiță, Dan Ștefănoiu, Alexandru Dumitrașcu</i>	
Urban Traffic Monitoring and Control as a Cyber-Physical System Approach	355
<i>Simona Iuliana Caramihai, Ioan Dumitrache</i>	
Digital Motor Control Library	367
<i>Radu Duma, Petru Dobra, Mirela Trusca, Ioan Valentin Sita</i>	
From HTML to 3DMMO – A Roadmap Full of Challenges	379
<i>Alin Dragos Bogdan Moldoveanu, Florica Moldoveanu, Victor Asavei, Alexandru Egner, Anca Morar</i>	
Schedulability Guarantees for Dependable Distributed Real-Time Systems under Error Bursts	393
<i>Huseyin Aysan, Radu Dobrin, Sasikumar Punnekkat</i>	
Improvements of Instruction Scheduling	407
<i>Bogdan Ditu, Nicolae Tapus</i>	
Re-scheduling Service for Distributed Systems	423
<i>Florin Pop, Ciprian Dobre, Catalin Negru, Valentin Cristea</i>	
Java Reflection Performance Analysis Using Different Java Development	439
<i>Cătălin Tudose, Carmen Odubășteanu, Serban Radu</i>	
Multi GPGPU Optimizations for 3D MMO Virtual Spaces	453
<i>Victor Asavei, Florica Moldoveanu, Alin Moldoveanu, Alexandru Egner, Anca Morar</i>	
Analyzing Social Networks Using Non-Metric Multidimensional Scaling	463
<i>Florin Radulescu, Cristian Turcitu</i>	

**Domain-Knowledge Optimized Simulated Annealing for
Network-on-Chip Application Mapping** 473
Ciprian Radu, Lucian Vințan

**A Comparison of Multi-objective Algorithms for the Automatic
Design Space Exploration of a Superscalar System** 489
Horia Calborean, Ralf Jahr, Theo Ungerer, Lucian Vințan

Author Index 503

On the Use of Stochastic Complexity in Spectral Analysis of Radial Velocity Data

Vili Forsell¹ and Ciprian Doru Giurcăneanu²

¹ Department of Signal Processing, Tampere University of Technology,
P.O. Box 553, FIN-33101 Tampere, Finland
vili.forsell@tut.fi

² Helsinki Institute for Information Technology, HIIT, University of Helsinki,
P.O. Box 68, FIN-00014 Helsinki, Finland
ciprian.giurcaneanu@hiit.fi

Abstract. The periodogram and its variants have been extensively used in the past for estimating the power spectral density. In this book chapter, we consider the case when the measurements are not equidistantly spaced on the time-axis, and we focus on testing the significance of the periodogram peaks. Because it is known that the standard tests of hypothesis testing are not suitable for this problem, we propose the use of Stochastic Complexity (SC). The performance of SC is evaluated in comparison with the Bayesian Information Criterion (BIC), which has been employed in the previous literature to solve the same problem. The numerical experiments on radial velocity measurements demonstrate that SC compares favorably with BIC.

Keywords: periodogram, Bayesian Information Criterion, Stochastic Complexity, radial velocity data, extrasolar planet.

1 Introduction

The history of using a periodogram for spectrum estimation can be traced back to 1900 when it was introduced by Schuster [1]. A modified form of it, the so-called Lomb-Scargle periodogram [2,3], is considered to be more suitable for the case when the real-valued measurements are not equidistantly spaced on the time-axis. It is well-known that the two previously mentioned forms of the periodogram have limited resolution capabilities due to the width of the main beam of the spectral window. Additionally, spurious peaks can occur due to the sidelobes of the spectral window [1]. Various improvements have been proposed to limit the effects of these drawbacks. For example, in [4] was introduced Real-valued Iterative Adaptive Approach (RIAA) which can be interpreted as an iteratively weighted periodogram. For an overview of the existing methods, we refer to [5].

It is important to remark that, up to now, most of the research effort was focused on estimating the power spectral density, whereas the problem of testing the significance of the periodogram peaks was less investigated. In [4], it was

noticed that the use of RIAA in combination with the Bayesian Information Criterion - BIC [6] does not produce good results. This is because the idealizing assumptions made in the derivation of BIC do not hold in practice. The solution adopted in [4] was to heuristically modify the expression of BIC. In [7], the method from [4] was further developed as follows: (i) instead of the real-valued algorithm (RIAA), its complex-valued variant (IAA) was employed; (ii) IAA estimates were refined by applying a relaxation maximum likelihood algorithm - RELAX [8]; (iii) the significance of the spectral peaks was tested with a carefully designed Generalized Likelihood Ratio Test (GLRT). Hence, the modified BIC was not employed albeit some heuristics were still involved.

In this book chapter, we focus on the use of Stochastic Complexity (SC) for estimating the number of sinusoidal terms. Our choice is mainly determined by the fact that SC has already been proven to be successful in many practical applications. Some recent examples can be found in [9,10]. In the case of the problem addressed here, the major difficulty is the computation of SC [11]. It was already shown in [12] that, for sinusoidal regression models, the only practical approach is the one which evaluates SC with the method from [13]. The key point is that the penalty term within the SC formula involves the determinant of the Fisher Information Matrix (FIM). Depending on the considered assumptions, FIM can be given by three different formulas [12,14]. However, the irregular sampling pattern makes the problem more complicated than the one from [12,14], where the discussion was restricted to the case of uniformly sampled measurements.

The interest for the spectral analysis of irregularly sampled data is not purely theoretical, but it is motivated by various applications from astronomy, seismology, paleoclimatology, genomics and laser Doppler velocimetry (see [5] for a comprehensive list of references). In the experimental part of this book chapter, we will focus on the analysis of radial velocity data.

The rest of the book chapter is organized as follows. Section 2 is devoted to the motivation of the work. In Section 3, we present two methods for estimating the spectrum of non-uniformly sampled data. The first one corresponds to the Least Squares (LS) solution, while the second one is obtained by applying the RIAA algorithm. Section 4 describes the BIC and SC criteria. The settings for our experiments are outlined in Section 5. Then, in Section 6, we investigate the estimation capabilities of various methods by conducting experiments on radial velocity data. Section 7 concludes the work.

2 Motivation of the Work

One of the most active research areas in astrophysics concerns the detection of planets located outside the Solar System. For obvious reasons, they are called extrasolar planets, or shortly exoplanets. To gain more insight, we resort to some statistics which are available in [15].

In this context, we mention that, by March 2012, the astronomers have confirmed the detection of more than 750 exoplanets. However, the number of

discoveries has increased significantly during the recent years: In 2010, more than 100 exoplanets have been discovered, whereas before 2006 no more than 50 exoplanets were detected in the same year.

Another interesting aspect is related to the detection methods. It seems that the most commonly employed technique is based on the analysis of the radial velocity data. The key idea is that an exoplanet orbiting a star could make the observed radial velocity of the star to oscillate within a certain range. Since the variations of the radial velocity cause Doppler shifts in the light emitted by the star, one might detect the exoplanet (or the exoplanets) revolving the star by analyzing the spectrum of the measured Doppler displacements.

In theory, this was well-known as early as 1952 [16], but only the recent advances in spectrometer technology and observational techniques have allowed the astrophysicists to use the radial velocity data for discovering new exoplanets. In practical applications, there are some limitations, and this makes the radial velocity data to be measured at non-uniformly spaced time intervals.

For the sake of concreteness, we will next consider the case of the three super-Earths orbiting HD40307, and which have been discovered in 2008. They are more massive than the Earth, and this is why they are dubbed ``super-Earths``. However, they are less massive than Uranus and Neptune, which are about 15 Earth masses [17].

Example (Planetary System HD40307). The astronomers from the European Southern Observatory (ESO) have used the High Accuracy Radial velocity Planet Searcher (HARPS) spectrograph which is attached to the La Silla telescope (in Chile) to observe the star HD40307 for about five years. The total number of the obtained measurements was 135. After discarding the observations deemed to be unreliable, a subset of 129 measurements recorded during about 878 days has been retained for further processing [17]. The retained measurements are publicly available in [18], and they are also plotted in Fig.1. Remark in Fig. 1 (a) the non-uniform sampling pattern that limits the application of the customary spectral analysis methods [1]. We discuss in the following sections how this difficulty can be circumvented.

3 Estimating the Power Spectral Density

Assume that the real-valued measurements $y(t_1), \dots, y(t_N)$ are available and they have been recorded at the moments t_1, \dots, t_N , which are not equidistantly spaced on the time-axis. Additionally, we have $\sum_{n=1}^N y(t_n) = 0$. With the convention that the operator $(\cdot)^T$ denotes transposition, we take the vector of observations to be

$$\mathbf{y} = [y(t_1) \cdots y(t_N)]^T.$$

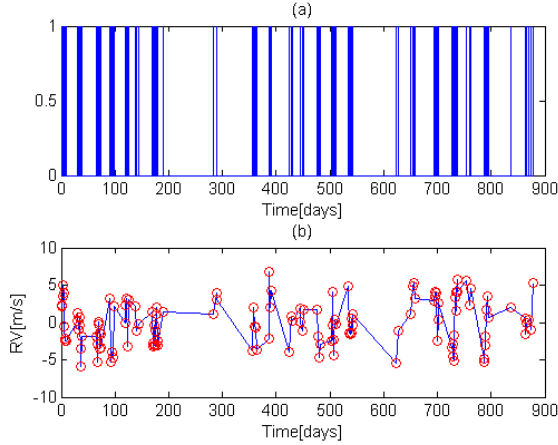


Fig. 1. Measurements for the planetary system HD40307: (a) The *sampling pattern* – the distance between two consecutive bars represents the sampling interval; (b) The *radial velocity data* after the mean value was subtracted from the measurements.

We define a uniform grid on the frequency axis such that its step size is given by $\Delta\omega$. Let K denote the number of grid points needed to cover the frequency interval $[0, \omega_{max}]$. If for $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the largest integer less than or equal to x , then $K = \lfloor \omega_{max}/\Delta\omega \rfloor$. We will discuss later how $\Delta\omega$ and ω_{max} can be selected.

Let $\omega_k = k \times \Delta\omega$ be an arbitrary point on the frequency grid defined above. Here, we discuss two different estimators for the power spectrum at frequency ω_k . The first one corresponds to the LS solution, while the second one is obtained by applying the RIAA algorithm. The presentation of the estimation procedures follows closely [4]. We need the supplementary notations:

$$\begin{aligned} \mathbf{c}_k &= [\cos(\omega_k t_1) \cdots \cos(\omega_k t_N)]^\top, \\ \mathbf{s}_k &= [\sin(\omega_k t_1) \cdots \sin(\omega_k t_N)]^\top, \\ \mathbf{A}_k &= [\mathbf{c}_k \quad \mathbf{s}_k]. \end{aligned}$$

The LS solution finds the vector of linear parameters

$$\boldsymbol{\theta}_k = [a_k \quad b_k]^\top \quad (1)$$

minimizing $\|\mathbf{y} - \mathbf{A}_k \boldsymbol{\theta}_k\|^2$, where $\|\cdot\|$ denotes the Euclidean norm. Hence, we get

$$\hat{\boldsymbol{\theta}}_k = (\mathbf{A}_k^\top \mathbf{A}_k)^{-1} \mathbf{A}_k^\top \mathbf{y}, \quad (2)$$

for which the corresponding power spectral estimate is given by

$$P_{LS}(\omega_k) = \frac{1}{N} \hat{\boldsymbol{\theta}}_k^\top (\mathbf{A}_k^\top \mathbf{A}_k) \hat{\boldsymbol{\theta}}_k. \quad (3)$$

The estimate $\hat{\boldsymbol{\theta}}_k$ from (2) is also used to initialize the RIAA algorithm. More precisely, it is re-denoted $\hat{\boldsymbol{\theta}}_k^0$ to emphasize that it is the output of the RIAA iteration indexed by $i = 0$. Then, for each iteration indexed by $i = 0, 1, 2, \dots$, two computation steps are performed. First, the matrix $\hat{\boldsymbol{\Gamma}}^i$ is calculated with formula:

$$\hat{\boldsymbol{\Gamma}}^i = \sum_{k=1}^K \mathbf{A}_k \mathbf{D}_k^i \mathbf{A}_k^T,$$

where \mathbf{D}_k^i is a 2×2 diagonal matrix whose non-zero entries equal $\|\hat{\boldsymbol{\theta}}_k^i\|^2/2$. Second, the improved estimates of the linear parameters are computed as follows:

$$\hat{\boldsymbol{\theta}}_k^{i+1} = \left[\mathbf{A}_k^T (\hat{\boldsymbol{\Gamma}}^i)^{-1} \mathbf{A}_k \right]^{-1} \left[\mathbf{A}_k^T (\hat{\boldsymbol{\Gamma}}^i)^{-1} \mathbf{y} \right],$$

where $k \in \{1, \dots, K\}$. The iterative process is stopped when a certain convergence criterion is satisfied. Conventionally, the estimates obtained at convergence are denoted by $\hat{\boldsymbol{\theta}}_k^c$. Similar to (3), the RIAA periodogram is given by

$$P_{RIAA}(\omega_k) = \frac{1}{N} (\hat{\boldsymbol{\theta}}_k^c)^T (\mathbf{A}_k^T \mathbf{A}_k) (\hat{\boldsymbol{\theta}}_k^c), \quad k \in \{1, \dots, K\}.$$

To illustrate the performance of both LS and RIAA, we use the dataset recorded for the star HD40307. We will discuss how to select the significant peaks of the spectra estimated by the LS and RIAA methods in Section 4.

Example (cont.). The velocity data from Fig. 1 (b) are used to compute the LS periodogram and the RIAA periodogram. In our settings, $\omega_{max} = \pi$, $\Delta\omega = \frac{2\pi}{10^3}$, and the number of iterations for the RIAA algorithm is 20. The estimated spectra are plotted in Fig. 2, where we have adopted the same convention as in the previous literature by changing the unit in the time domain from “day” to “second”. In Fig. 2, it is evident that RIAA is superior to LS, in the sense that

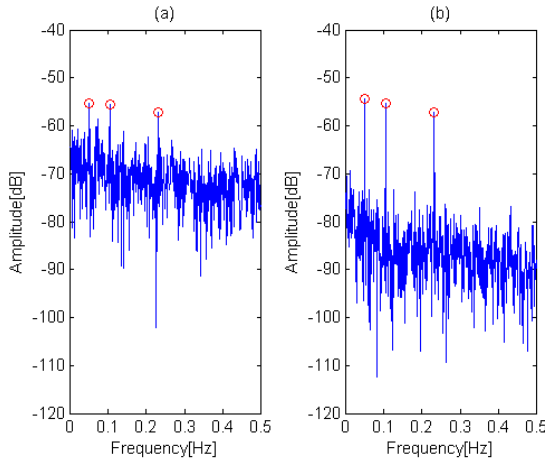


Fig. 2. Spectral estimates for the planetary system HD40307 obtained by using the measurements plotted in Fig. 1 (b). The *LS periodogram* is presented in the left panel, while the *RIAA periodogram* is presented in the right panel. In both panels, the estimated spectra are represented with *solid lines* and the *circles* show the positions of the true frequencies.

RIAA separates better the dominant peaks from the spurious ones. Remark also that the dominant peaks are located in positions which almost coincide with the true frequencies that correspond to the orbital cycles of the three super-Earths, namely 4.3, 9.6 and 20.5 days.

4 Testing the Significance of the Periodogram Peaks

It was already pointed out in [4] that the standard tests of hypothesis testing are not suitable for this problem, and Stoica et al. proposed the following application of the BIC. Let $P(\omega_1), \dots, P(\omega_K)$ denote the values taken by either the LS or the RIAA periodogram at the points of the frequency grid. For simplicity, we use the notation P_k instead of $P(\omega_k)$ for all $k \in \{1, \dots, K\}$. Additionally, we consider a permutation of the indexes such that the associated periodogram values are ordered decreasingly:

$$P_{(1)} \geq P_{(2)} \geq \dots \geq P_{(K)}.$$

For an arbitrary $M \in \{1, \dots, K\}$, the residual sum of squares has the expression:

$$R\check{S}_M = \sum_{m=1}^M \|\mathbf{y} - \mathbf{A}_{(m)} \check{\boldsymbol{\theta}}_{(m)}\|^2, \quad (4)$$

where $\check{\boldsymbol{\theta}}_{(m)}$ stands for the LS estimate $\hat{\boldsymbol{\theta}}_{(m)}^0$ or the RIAA estimate $\hat{\boldsymbol{\theta}}_{(m)}^c$. The definition above is extended also for $M = 0$ by taking $R\check{S}_0 = \|\mathbf{y}\|^2$. The optimum number of sinusoids (\check{M}) is chosen as follows:

$$\check{M} = \arg \min_{M=0,1,2,\dots} BIC_\rho(M), \text{ where } BIC_\rho(M) = \frac{N}{2} \ln R\check{S}_M + \frac{\rho M}{2} \ln N.$$

In the classical formula of BIC, the value of ρ is five (see [19] and the references therein). However, some ad-hoc modifications were proposed in [4] such that ρ is one in the case when the parameters are estimated with the LS formula from (2). Moreover, it is recommended in [4] to choose $\rho = 4$ when the RIAA algorithm is applied.

We propose to estimate the number of sinusoids by applying the SC criterion which was derived in [12] by relying on the results from [13]. To this end, we operate the following change of variables. If a_k and b_k are the linear parameters from (1), then we define: $\phi_k = -\arctan(b_k/a_k)$ and $\alpha_k = a_k/\cos\phi_k$. To solve the problem of phase ambiguity, we take $\phi_k \in [-\pi, \pi)$ and $\alpha_k > 0$. Hence, the parameters of the k -th sine-wave are $\boldsymbol{\xi}_k = [\alpha_k \ \omega_k \ \phi_k]^\top$.

The selection rule which we want to apply involves the determinant of the block-diagonal FIM (see [12] and the references therein):

$$J_{M,N}(\boldsymbol{\zeta}) = \begin{bmatrix} J_N(\boldsymbol{\xi}_{(1)}) & & & \\ & \ddots & & \\ & & J_N(\boldsymbol{\xi}_{(M)}) & \\ & & & J_N(\boldsymbol{\tau}) \end{bmatrix}, \text{ where } \boldsymbol{\zeta} = [\boldsymbol{\xi}_{(1)}^\top \ \dots \ \boldsymbol{\xi}_{(M)}^\top \ \boldsymbol{\tau}^\top]^\top,$$

$$\begin{aligned}
\mathbf{J}_N(\xi_{(m)}) &= \mathbf{Q}_N \mathbf{G}(\xi_{(m)}, \tau) \mathbf{Q}_N, \\
\mathbf{Q}_N &= \begin{bmatrix} N^{1/2} & 0 & 0 \\ 0 & N^{3/2} & 0 \\ 0 & 0 & N^{1/2} \end{bmatrix}, \\
\mathbf{G}(\xi_{(m)}, \tau) &= SNR_{(m)} \begin{bmatrix} 1/\alpha_{(m)}^2 & 0 & 0 \\ 0 & 1/3 & 1/2 \\ 0 & 1/2 & 1 \end{bmatrix}, \\
\mathbf{J}_N(\tau) &= \frac{N}{2\tau^2}.
\end{aligned}$$

In the equations above, the index m satisfies the double inequality $1 \leq m \leq M$. More importantly, it is assumed that, besides the M sinusoidal components with frequencies $\omega_{(1)}, \dots, \omega_{(M)}$, the data \mathbf{y} contain Gaussian noise whose variance is denoted by τ . Additionally, $SNR_{(m)} = \alpha_{(m)}^2/(2\tau)$ is the local signal-to-noise ratio for the m -th sine-wave.

The estimates for the entries of ζ can be obtained straightforwardly. Let $\check{\zeta}$ be the vector obtained by replacing the entries of ζ with their estimated values. It is worth mentioning that $\check{\tau} = R\check{S}_M/N$, where $R\check{S}_M$ has been defined in (4). Then SC is computed with the formula:

$$SC(M) = \frac{N}{2} \ln R\check{S}_M + \ln |\mathbf{J}_{M,N}(\check{\zeta})|^{1/2} + \sum_{i=1}^{3M+1} \ln \left(|\check{\zeta}_i| + N^{-\frac{1}{4}} \right).$$

Obviously, the optimum value of M is the one which minimizes the expression above.

5 Settings for the Numerical Examples

A common feature for all the datasets is the highly irregular sampling pattern, which poses troubles when one wants to choose the value of ω_{max} in order to define the frequency grid. In the previous literature [4, 20], it is recommended to select ω_{max} by relying on the properties of the spectral window

$$W(\omega) = |\sum_{n=1}^N \exp(j\omega t_n)|^2,$$

where $j = \sqrt{-1}$. As the maximum value of ω , which equals N^2 , is reached for $\omega = 0$, ω_{max} can be taken to be the smallest positive frequency ω that satisfies the condition $W(2\omega) \approx N^2$. Because we observed experimentally that such an approach leads to values of ω_{max} which are too large, we decided to choose $\omega_{max} = \pi$.

However, the estimation results obtained when $\omega_{max} = \pi$ should be interpreted with caution. In our experiments, we have considered datasets for which the existing knowledge guarantees that $\omega_{max} = \pi$ is a reasonable choice. We have found at least one case when we noticed that it is more appropriate to choose a

larger value for ω_{max} . This was the 86-samples dataset measured for the HD41004 AB system [21]. We do not plan to discuss here the estimation results obtained with this dataset, because HD41004 AB is truly peculiar in the sense that it is a visual double system. Its analysis should be based on the extended set of measurements, and not on the original dataset of 86 samples (see for example [22]).

Another important parameter of the tested algorithms is $\Delta\omega$. Typically, $\Delta\omega$ is chosen to be ten times smaller than $(2\pi)/(t_N - t_1)$ [4]. Remark that the smaller is $\Delta\omega$, the larger is the computational burden, especially in the case of RIAA algorithm. By considering the values of the difference $t_N - t_1$ for the measurements used in experiments, we have chosen $\Delta\omega$ to be $(2\pi)/10^3$.

We also mention that, in our experimental settings, the number of iterations for the RIAA algorithm equals 20.

The obtained results are presented in Section 6.

6 Experimental Results

We illustrate the application of the algorithms described in the previous sections by using radial velocity measurements that are publicly available. Their complete description can be found in the astrophysics literature. Because of the limited typographic space, we do not provide details for each dataset, but the references where the interested reader can find the full information are indicated in Table 1. In each case, we outline in the same table the number of available measurements (N), as well as the difference (in days) between the last and the first sampling times ($t_N - t_1$).

Given that the number of observations for a dataset is N , estimating the number of sinusoids reduces to selecting from the set $\{0, 1, \dots, \min\left(10, \left\lfloor \frac{N-2}{3} \right\rfloor\right)\}$ the value which minimizes either the BIC or the SC criterion. Table 1 shows the estimation results and, for ease of comparison, the true number of sinusoids (M). A complete analysis would require a careful consideration of the background knowledge from the astrophysics literature. However, we restrict the discussion to the performance of the tested algorithms.

Notice that for $M = 1$, the use of the RIAA periodogram does not produce results which are superior to those obtained with the LS periodogram. This is not surprising because, in the case of a single sinusoidal signal in white Gaussian noise, the LS estimate coincides with the maximum likelihood estimate [4].

It is also interesting to observe that the combination LS+BIC has a slight tendency to overestimate the number of sine-waves when $\rho = 1$. This suggests that the value of one which was empirically chosen for ρ is too small. On the contrary, LS+SC does not overestimate, but fails to detect the presence of the sinusoid for the dataset HD210277.

When $M = 2$, the application of RIAA is clearly beneficial in the sense that M is correctly estimated by SC and also by the two variants of BIC. The only exception is the overestimation produced for G1176 by RIAA+BIC when $\rho = 4$.

A similar behavior of RIAA+BIC can be observed also for HD40307 which, in our experiments, is the only dataset with $M = 3$. The results obtained for GI176 and HD40307 show that, in some cases, the ad-hoc selection of $\rho = 4$ does not penalize enough.

Recall that, in the previous sections, we have analyzed more carefully the case of the planetary system HD40307. Because of the limited space, we cannot perform a similar analysis for all datasets, and we restrict our investigation to the accuracy of the frequencies estimated when the number of sine-waves is chosen correctly. The estimation errors are shown in Table 2 for all the cases considered in Table 1, with the exception of HD187123 for which there is a slight ambiguity concerning one of the two frequencies involved [23, 24]. The results presented in Table 2 confirm that the estimations are reasonably good.

Table 1. Number of sine-waves estimated by BIC and SC when they are applied to the outcomes of the LS and RIAA algorithms. The column ‘‘Reference’’ points out the sources where the complete information on the datasets can be found. In each case, N is the sample size and Δt is the difference between the last and the first sampling times ($t_N - t_1$). The estimated value of M is written in bold if it coincides with the true M , underlined in the case of underestimation, and overlined in the case of overestimation.

Dataset	References	N	Δt	M	Estimated M				
					LS		RIAA		
					BIC $\rho = 1$	SC	BIC $\rho = 4$	BIC $\rho = 5$	SC
HD195019	[25]	19	2239.9	1	1	1	1	1	1
BD170063	[26]	26	1760.2	1	1	1	1	1	1
HD23596	[27]	39	1856.9	1	1	1	1	1	1
HD73267	[26]	39	1586.7	1	1	1	1	1	1
HD50554	[27]	41	1951.7	1	1	1	1	1	1
HD139357	[28]	49	1286.6	1	1	1	1	1	1
HD145377	[26]	64	1106.1	1	1	1	1	1	1
HD217014	[23]	153	3277.0	1	<u>2</u>	1	1	1	1
HD192263	[29]	181	1237.7	1	1	1	<u>2</u>	1	1
HD179949	[25]	17	2844.2	1	1	1	<u>2</u>	<u>0</u>	1
HD117176	[23]	35	2625.9	1	<u>2</u>	1	1	<u>0</u>	1

Table 1. (continued)

HD131664	[26]	41	1462.9	1	1	1	$\bar{2}$	$\bar{2}$	1
42DRACONIS	[28]	45	1208.9	1	1	1	$\bar{3}$	1	$\bar{3}$
HD162020	[30]	46	842.8	1	1	1	$\bar{2}$	$\bar{2}$	$\bar{2}$
HD153950	[26]	49	1791.3	1	1	1	$\bar{2}$	$\bar{2}$	$\bar{2}$
HD80606b	[31]	67	2805.2	1	1	1	$\bar{2}$	$\bar{2}$	$\bar{2}$
HD141937	[30]	81	881.7	1	1	1	$\bar{5}$	$\bar{5}$	$\bar{4}$
HD209458	[23]	187	1883.8	1	1	1	$\bar{3}$	$\bar{3}$	$\bar{3}$
HD210277	[25, 32]	21	2396.3	1	1	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
HD106252	[27]	40	2242.8	1	$\bar{2}$	1	<u>0</u>	<u>0</u>	<u>0</u>
HD20868	[26]	48	1705.2	1	$\bar{2}$	1	$\bar{4}$	$\bar{4}$	$\bar{4}$
HD190228	[27]	51	1945.7	1	$\bar{2}$	1	$\bar{2}$	$\bar{2}$	$\bar{2}$
G1176	[33]	57	1442.0	2	2	2	$\bar{3}$	2	2
HD45364	[34]	58	1582.8	2	2	<u>1</u>	2	2	2
GJ674	[35]	32	824.8	2	<u>1</u>	<u>1</u>	2	2	2
HD187123	[23, 24]	57	1801.1	2	<u>1</u>	<u>1</u>	2	2	2
HD40307	[18]	129	877.7	3	3	3	$\bar{4}$	3	3

Table 2. Errors in estimating the frequencies when the number of sine-waves is correctly selected. The true frequencies are calculated based on the orbital cycle values which are taken from the astrophysics literature. The symbol “-” is used when the number of sine-waves is either underestimated or overestimated.

Data Set	Orbital Cycles (Days)	Frequency (Hz)	Error (Hz)				
			LS		RIAA		
			BIC	SC	BIC	BIC	SC
$\rho = 1$	$\rho = 4$	$\rho = 5$					
HD195019	18.2008	0.0549	0.001	0.001	0.001	0.001	0.001
BD170063	655.6	0.0015	0.001	0.001	0.001	0.001	0.001
HD23596	1565	0.0006	0.001	0.001	0.001	0.001	0.001

Table 2. (continued)

HD73267	1260	0.0008	0.001	0.001	0.001	0.001	0.001
HD50554	1293	0.0008	0.001	0.001	0.001	0.001	0.001
HD139357	1125.7	0.0009	0.001	0.001	0.001	0.001	0.001
HD145377	103.95	0.0096	0.001	0.001	0.001	0.001	0.001
HD217014	4.23077	0.2364	-	0.034	0.034	0.034	0.034
			-	-	-	-	-
HD192263	24.348	0.0411	0.001	0.001	-	0.001	0.001
			-	-	-	-	-
HD179949	3.09250	0.3234	0.296	0.296	-	-	0.296
HD117176	116.689	0.0086	-	0.020	0.020	-	0.020
HD131664	1951	0.0005	0.001	0.001	-	-	0.001
			-	-	-	-	-
42DRACONIS	479.1	0.0021	0.001	0.001	-	0.001	-
HD162020	8.428198	0.1186	0.001	0.001	-	-	-
			-	-	-	-	-
HD153950	499.4	0.0020	0.001	0.001	-	-	-
HD80606b	111.436	0.0090	0.020	0.020	-	-	-
HD141937	653.22	0.0015	0.003	0.003	-	-	-
HD209458	3.5246	0.2837	0.001	0.001	-	-	-
HD210277	442.1	0.0023	0.050	-	-	-	-
HD106252	1600	0.0006	-	0.002	-	-	-
HD20868	380.85	0.0026	-	0.010	-	-	-
HD190228	1146	0.0009	-	0.001	-	-	-
<hr/>							
G1176	40	0.0250	0.000	0.000	-	0.000	0.000
	8.7836	0.1138	0.001	0.001	-	0.001	0.001
HD45364	342.85	0.0029	0.001	-	0.001	0.001	0.001
	226.93	0.0044	0.029	-	0.029	0.029	0.029
GJ674	34.8467	0.0287	-	-	0.001	0.001	0.001
			-	-	-	-	-
	4.6938	0.2130	-	-	0.001	0.001	0.001
<hr/>							
HD40307	20.46	0.0489	0.001	0.001	-	0.001	0.001
	9.620	0.1040	0.001	0.001	-	0.001	0.001
	4.3115	0.2319	0.001	0.001	-	0.001	0.001
<hr/>							

7 Conclusion

The experiments conducted on real life data have shown that SC is superior to BIC, especially because SC does not involve any empirical tuning of the penalty term. It remains to investigate more theoretically the capabilities of SC in solving the estimation problem which was addressed by this book chapter.

Acknowledgements. The work of V. Forsell was supported by the Academy of Finland under Project No. 213462. Part of the work of C.D. Giurcăneanu was done when he was with the Department of Signal Processing, Tampere University of Technology, and it was supported by the Academy of Finland, Project Nos. 113572, 134767, 213462.

References

1. Stoica, P., Moses, R.: *Spectral Analysis of Signals*. Prentice-Hall (2005)
2. Lomb, N.R.: Least-squares Frequency Analysis of Unequally Spaced Data. *Astrophysics and Space Science* 39(2), 447–462 (1976)
3. Scargle, J.D.: Studies in Astronomical Time Series Analysis. II - Statistical Aspects of Spectral Analysis of Unevenly Spaced Data. *Astrophysical Journal* 263(pt. 1), 835–853 (1982)
4. Stoica, P., Li, J., He, H.: Spectral Analysis of Nonuniformly Sampled Data: A New Approach Versus the Periodogram. *IEEE Transactions on Signal Processing* 57(3), 843–858 (2009)
5. Babu, P., Stoica, P.: Spectral Analysis of Nonuniformly Sampled Data - A Review. *Digital Signal Processing* 20(2), 359–378 (2010)
6. Schwarz, G.: Estimating the Dimension of the Model. *Ann. Stat.* 6, 461–464 (1978)
7. Babu, P., Stoica, P., Li, J., Chen, Z., Ge, J.: Analysis of Radial Velocity Data by a Novel Adaptive Approach. *The Astronomical Journal* 139(2), 783 (2010)
8. Li, J., Stoica, P.: Efficient Mixed-spectrum Estimation with Applications to Target Feature Extraction. *IEEE Transactions on Signal Processing* 44(2), 281–295 (1996)
9. Ghido, F., Tabus, I.: Performance of Sparse Modeling Algorithms for Predictive Coding. In: *Proc. of 18th Int. Conf. on Control Systems and Computer Science*, vol. 2, pp. 931–935. Politehnica Press (2011)
10. Hukkanen, J., Sabo, E., Tabus, I.: MDL Based Structure Selection of Union of Ellipse Models at Multiple Scale Descriptions. In: *Proc. of 18th Int. Conf. on Control Systems and Computer Science*, vol. 2, pp. 947–952. Politehnica Press (2011)
11. Rissanen, J.: *Information and Complexity in Statistical Modeling*. Springer (2007)
12. Giurcăneanu, C.D.: Estimation of Sinusoidal Regression Models by Stochastic Complexity. In: Grünwald, P., Myllymäki, P., Tabus, I., Weinberger, M., Yu, B. (eds.) *Festschrift in Honor of Jorma Rissanen on the Occasion of His 75th Birthday*. TICSP, vol. 38, pp. 229–249. Tampere International Center for Signal Processing (2008)
13. Qian, G., Künsch, H.R.: Some Notes on Rissanen's Stochastic Complexity. *IEEE Transactions on Information Theory* 44(2), 782–786 (1998)
14. Giurcăneanu, C.D.: Stochastic Complexity for the Estimation of Sine-waves in Colored Noise. In: *Proc. of 2007 Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1097–1100. IEEE (2007)

15. Scheider, J.: The Extrasolar Planets Encyclopaedia, <http://exoplanet.eu/catalog.php>
16. Struve, O.: Proposal for a Project of High-Precision Stellar Radial Velocity Work. The Observatory 72, 199–200 (1952)
17. Mayor, M., Udry, S., Lovis, C., Pepe, F., Queloz, D., Benz, W., Bertaux, J.-L., Bouchy, F., Mordasini, C., Segransan, D.: The HARPS Search for Southern Extrasolar Planets XIII. A Planetary System with 3 Super-Earths (4.2, 6.9, and 9.2 M \oplus). *Astronomy and Astrophysics* 493, 639–644 (2009)
18. Harvard University Center for Astrophysics: The VizieR Catalogue Service, <http://vizier.cfa.harvard.edu/viz-bin/nph-Plot/Vgraph/txt?J%2FA%2bA%2f493%2f639%2f.%2ftable1&P=0>
19. Stoica, P., Selen, Y.: A Review of Information Criterion Rules. *IEEE Signal Processing Magazine* 21(4), 36–47 (2004)
20. Eyer, L., Bartholdi, P.: Variable Stars: Which Nyquist Frequency? *Astron. Astrophys.* 135(suppl. 1), 1–3 (1999)
21. Santos, N.C., Mayor, M., Naef, D., Pepe, F., Queloz, D., Udry, S., Burnet, M., Clausen, J.V., Helt, B.E., Olsen, E.H., Pritchard, J.D.: The CORALIE Survey for Southern Extra-solar Planets. IX. A 1.3-day Period Brown Dwarf Disguised as a Planet. *Astronomy and Astrophysics* 392, 215–229 (2002)
22. Zucker, S., Mazeh, T., Santos, N.C., Udry, S., Mayor, M.: Multi-order TODCOR: Application to Observations Taken with the CORALIE Echelle Spectrograph II. A Planet in the System HD 41004. *Astronomy and Astrophysics* 426, 695–698 (2004)
23. Naef, D., Mayor, M., Beuzit, J.L., Perrier, C., Queloz, D., Sivan, J.P., Udry, S.: The ELODIE Survey for Northern Extra-solar Planets III. Three Planetary Candidates Detected with ELODIE. *Astronomy and Astrophysics* 414, 351–359 (2004)
24. Wright, J.T., Marcy, G.W., Fischer, D.A., Butler, R.P., Vogt, S.S., Tinney, C.G., Jones, H.R.A., Carter, B.D., Johnson, J.A., McCarthy, C., Apps, K.: Four New Exoplanets and Hints of Additional Substellar Companions to Exoplanet Host Stars. *The Astrophysical Journal* 657, 533–545 (2007)
25. Wittenmyer, R.A., Endl, M., Cochran, W.D.: Long-Period Objects in the Extrasolar Planetary Systems 47 Ursae Majoris and 14 Herculis. *The Astrophysical Journal* 654, 625–632 (2007)
26. Moutou, C., Mayor, M., Curto, G.L., Udry, S., Bouchy, F., Benz, W., Lovis, C., Naef, D., Pepe, F., Queloz, D., Santos, N.C.: The HARPS Search for Southern Extrasolar Planets XV. Six Long-period Giant Planets Around BD 170063, HD 20868, HD 73267, HD 131664, HD 145377, and HD 153950. *Astronomy and Astrophysics* 496, 513–519 (2009)
27. Perrier, C., Sivan, J.-P., Naef, D., Beuzit, J.L., Mayor, M., Queloz, D., Udry, S.: The ELODIE Survey for Northern Extra-solar Planets I. Six New Extra-solar Planet Candidates. *Astronomy and Astrophysics* 410, 1039–1049 (2003)
28. Döllinger, M.P., Hatzes, A.P., Pasquini, L., Guenther, E.W., Hartmann, M., Girardi, L.: Planetary Companion Candidates Around the K Giant Stars 42 Draconis and HD 139357. *Astronomy and Astrophysics* 499, 935–942 (2009)
29. Santos, N.C., Udry, S., Mayor, M., Naef, D., Pepe, F., Queloz, D., Burki, G., Cramer, N., Nicolet, B.: The CORALIE Survey for Southern Extra-solar Planets XI. The Return of the Giant Planet Orbiting HD 192263. *Astronomy and Astrophysics* 406, 373–381 (2003)

30. Udry, S., Mayor, M., Naef, D., Pepe, F., Queloz, D., Santos, N.C., Burnet, M.: The CORALIE Survey for Southern Extra-solar Planets VIII. The Very Low-mass Companions of HD 141937, HD 162020, HD 168443 and HD 202206: Brown Dwarfs or "Superplanets"? *Astronomy and Astrophysics* 390, 267–279 (2002)
31. Moutou, C., Hébrard, G., Bouchy, F., Eggenberger, A., Boisse, I., Bonfils, X., Gravallon, D., Ehrenreich, D., Forveille, T., Delfosse, X., Desort, M., Lagrange, A.-M., Lovis, C., Mayor, M., Pepe, F., Perrier, C., Pont, F., Queloz, D., Santos, N.C., Ségransan, D., Udry, S., Vidal-Madjar, A.: Photometric and Spectroscopic Detection of the Primary Transit of the 111-day-period Planet HD 80606b. *Astronomy and Astrophysics* 498, L5–L8 (2009)
32. Marcy, G.W., Butler, R.P., Vogt, S.S., Fischer, D., Liu, M.C.: Two New Candidate Planets in Eccentric Orbits. *The Astrophysical Journal* 520, 239–247 (1999)
33. Forveille, T., Bonfils, X., Delfosse, X., Gillon, M., Udry, S., Bouchy, F., Lovis, C., Mayor, M., Pepe, F., Perrier, C., Queloz, D., Santos, N., Bertaux, J.-L.: The HARPS Search for Southern Extra-solar Planets XIV. Gl176b, a Super-Earth Rather Than a Neptune, and at a Different Period. *Astronomy and Astrophysics* 493, 645–650 (2009)
34. Correia, A.C.M., Udry, S., Mayor, M., Benz, W., Bertaux, J.-L., Bouchy, F., Laskar, J., Lovis, C., Mordasini, C., Pepe, F., Queloz, D.: The HARPS Search for Southern Extra-solar Planets XVI. HD 45364, a Pair of Planets in a 3:2 Mean Motion Resonance. *Astronomy and Astrophysics* 496, 521–526 (2009)
35. Bonfils, X., Mayor, M., Delfosse, X., Forveille, T., Gillon, M., Perrier, C., Udry, S., Bouchy, F., Lovis, C., Pepe, F., Queloz, D., Santos, N.C., Bertaux, J.-L.: The HARPS Search for Southern Extra-solar Planets - X. A $m \sin i = 11M_{\oplus}$ Planet Around the Nearby Spotted M Dwarf GJ 674. *Astronomy and Astrophysics* 474, 293–299 (2007)

On an Input Driven Hierarchy of Hybrid Automata

Virginia Ecaterina Oltean, Radu Dobrescu, and Dan Popescu

Politehnica University of Bucharest, Faculty of Control and Computers,
Spl.Independenței 313, sector 6,
77206 Bucharest, Romania
{ecaterina_oltean, rd_dobrescu, dan_popescu_2002}@yahoo.com

Abstract. The hybrid automata modelling framework for hybrid systems can describe a continuous time system receiving a switching control. A piecewise constant signal generator can also be described as a timed hybrid automaton. A special situation, frequent in automotive control, occurs when a continuous time system with intrinsic hybrid nature receives, in open loop, a switching input. The switching input has a determinant role, but the hybrid behaviour reflects also the structure of the controlled system. Starting from the classic formalism of autonomous and timed automata, this paper proposes the concept of hierarchy of two automata, as a model describing the interaction of a second order oscillating system with a switching signal generator. The hierarchy reflects the fundamental cause of oscillations.

Keywords: hybrid automaton, differential equation, switched input, hybrid time set.

1 Introduction

Hybrid systems, combining event-driven with time driven dynamics, have emerged in the past two decades as an important and complex tool for both behaviour analysis and design of systems in various areas, from the study of cell biology [1] to air craft management [2] and behavioural models [3].

Unlike the classic continuous systems, dominated by the differential equations formalism, there is no unique or central modelling framework describing hybrid systems.

According to [4], modelling languages for hybrid systems have to be *descriptive* - i.e. to have a modelling capability -, *composable* - i.e. to allow building larger models consisting of simpler components - and *abstractable* - i.e. to allow design problems for composite models to be refined to design problems for individual components. Several modelling languages for hybrid systems have been developed

in the literature, which play more emphasis on different aspects, depending on the applications and problems they are designed to address. Examples are the mixed logic-dynamic (MLD) framework [5] for embedded control systems [6], the hybrid supervision approach [7], inspired from control systems or Hytech [8], dedicated mainly to verification tasks and inspired from computer science.

Hybrid automata are a fairly rich general modelling tool, in terms of descriptive power and they appear in the literature in various forms [9], [10], [11], [12]. They represent dynamical systems that describe the evolution in time of the valuations of a set of discrete and continuous variables. The most frequently encountered hybrid automata models are *autonomous*, i.e. with no inputs and no outputs and therefore they are *not* suited for the study of composition and abstraction properties.

However, depending on the research focus, some authors have proposed hybrid automata models that include the interaction with the external environment: examples are the model introduced, for specific control purposes, in [10], or the hybrid input-output automata in [13], which permits the description of closed loop control systems, with hybrid plant and hybrid controller. These models have a more descriptive power and they are more complex.

Hybrid automata, as abstractions of systems based on phased operations, involve both continuous “flows”, determined by local differential equations and discrete “jumps” determined by a directed graph. The jumps are determined by the continuous state evolution leaving a specified region, which is associated, as state invariance domain, to the local differential law. Hence classic autonomous hybrid automata can capture either autonomous switched systems, or closed loop control systems with switching control law.

A special situation occurs when an external switched signal is injected, in open loop, as control input of a continuous system, thus forcing the system to “jump”; in this case, the controlled system gets a hybrid nature due to the switching input, and the discrete transitions can be modelled by including a clock into the continuous dynamics. A more complicated case arises when, additionally, the continuous system has itself an intrinsic hybrid nature, which “comes to life” at each switching of the input signal. This can arise, for example, when testing, with piecewise-constant input signals, special classes of automotive systems, which already contain nonlinear switching components, like the clutch in the structure of an automobile power train [14]; the resulting model is a *piecewise affine linear model* [15], [16], [17] with piecewise-constant open loop inputs. In such a complex hybrid system, the difficulty comes from the fact that the “jumps” of the input signal differ from the “jumps” due to the intrinsic hybrid nature of the plant, but the first ones play a determinant role.

For the description of this class of complex hybrid systems, this paper proposes a hierarchy of two hybrid autonomous automata, with a dominant, higher priority *timed* hybrid automaton [18], describing the dynamics of the input signal and incorporating a controlled, lower priority hybrid automaton, which represents the plant with intrinsic hybrid nature [19]. The model can be extended, for the case of continuous systems with intrinsic hybrid nature and vector-valued inputs, to a

multilevel hierarchy of hybrid incorporated automata, each one, except the last one, with an associated clock, managing the vector valued switching signal. This approach is different from the analysis presented in [20], which focuses on the bridge between hybrid automata and piecewise affine models, with emphasis on the model uncertainty associated to hybrid automata transitions.

The paper is structured as follows. In section 2 a review of the hybrid automata models is presented. In section 3 is introduced the concept of hierarchy of two hybrid automata, composed of a second order system with piecewise constant input signal, followed by concluding remarks.

2 The Hybrid Automata Model - A Review

2.1 A Reminder of the Branicky Classification of Hybrid Systems

In the classification proposed by Branicky [21], [22], the hybrid behaviour is born by adding discrete phenomena to continuous dynamics. Recall that a continuous system is usually modelled by an ordinary differential equation (ODE)

$$\dot{x}(t) = f(x(t)), \quad (1)$$

where $x(t) \in X$ is the state vector at time $t \in \mathbf{R}$, $X \subseteq \mathbf{R}^n$ is the state space and $f: X \rightarrow TX$ is a vector field. The existence and uniqueness of the solution are assumed. Traditionally, a solution of (1) is a differentiable function $\varphi: (t_1, t_2) \rightarrow X$ satisfying $\dot{\varphi}(t) = f(\varphi(t))$, $\forall t \in (t_1, t_2)$, and $\varphi(\cdot)$ satisfies the initial condition x_0 at $t_0 \in \mathbf{R}$ if $t_1 < t_0 < t_2$ and $\varphi(t_0) = x_0$ [23]. However, in ODE modelling physical systems dynamics one considers frequently solutions over closed finite intervals, $\varphi: [t_0, T] \rightarrow X$, $T > 0$, differentiable over (t_0, T) and right and left differentiable in t_0 and T , respectively [24].

In order to introduce the discrete phenomena, which drive to hybrid behaviour, the continuous dynamics is modelled by the ODE

$$\dot{x} = \xi(t), \quad t \geq 0, \quad (2)$$

where $x(\cdot)$ is the continuous part of the hybrid state and the vector field $\xi(\cdot)$ is assumed to depend on x , on the eventual continuous control u and on the discrete phenomena.

The classification of Branicky considers four types of discrete phenomena:

- *autonomous switching*, where the vector field ξ changes discontinuously, when the state x hits a boundary or enters a region in the state space;
- *autonomous jumps*, where the state x changes discontinuously;

- *controlled switching*, where a control u switches the vector field ξ discontinuously and
- *controlled jumps*, where a control u changes the state x discontinuously.

The first two situations occur, generally, when the state x hits a boundary or enters a region in the state space, in a free evolution.

2.2 Autonomous Hybrid Automata - A Classic Example

The definition below is presented, with slightly different notations and nuances, in [12], [4], [25] and [26]. The presentation is detailed for a clarity concerning the model introduced in next section.

Definition 1. An autonomous hybrid automaton is a collection $H = (Q, X, f, Init, D, E, G, R)$, where

- $Q = \{q_1, q_2, \dots\}$ is a set of discrete states;
- $X = \mathbf{R}^n$ is a set of continuous states;
- $f(\cdot, \cdot): Q \times X \rightarrow TX$ is a family of vector fields;
- $Init \subseteq Q \times X$ is a set of initial states;
- $D(\cdot): Q \rightarrow 2^X$ is a domain application;
- $E \subseteq Q \times Q$ is a set of edges;
- $G(\cdot): E \rightarrow 2^X$ is a guard condition;
- $R(\cdot, \cdot): E \times X \rightarrow 2^X$ is a reset map.

A state of H is $(q, x) \in Q \times X$ and 2^X is the power set of X . Also, it is assumed that for all $q \in Q$, $f(q, \cdot)$ is Lipschitz continuous, for all $e \in E$, $G(e) \neq \emptyset$ and for all $x \in G(e)$, $R(e, x) \neq \emptyset$. The domain application is called also invariant, in the computer science literature.

Autonomous hybrid automata define possible evolutions of their states expressed as *hybrid trajectories*. Starting from an initial value $(q_0, x_0) \in Init$, the continuous state evolves according to the state equation $\dot{x} = f(q_0, x)$, $x(0) = x_0$, and the discrete state q remains constant at q_0 , as long as $x(t) \in D(q_0)$. If at some point the continuous state x reaches the guard $G(q_0, q_1) \subseteq X$ at some edge $(q_0, q_1) \in E$, a discrete transition can take place and the discrete state may change value to q_1 . At the same time, the continuous state gets reset to some value $R(q_0, q_1, x) \subseteq X$. After this discrete transition, continuous evolution resumes and the whole process is repeated. This is reflected as an *execution* of the autonomous

hybrid automaton, which is a *hybrid trajectory* obeying to restrictions imposed by the system, i.e. *accepted* by the system. This can be formalized as follows.

Definition 2. A hybrid time set is a finite or infinite sequence of intervals $\tau = \{I_k\}_{k=0}^N$, s.t. $I_k = [\tau_k, \tau'_k]$ for all k , $I_N = [\tau_N, \tau'_N]$, if $N < \infty$ or $I_N = [\tau_N, \tau'_N)$ if $N = \infty$, and $\tau_k \leq \tau'_k = \tau_{k+1}$, for all k .

Hence τ'_k and τ_{k+1} , $k < N$, correspond to the time instants just before and just after some instantaneous discrete transition of the hybrid system takes place, respectively. If $\tau_k = \tau'_k$, then multiple transitions take place one after the other. Denote $\langle \tau \rangle = \{1, 2, \dots, N\}$, if $N < \infty$, and $\langle \tau \rangle = \{1, 2, \dots\}$, if $N = \infty$. The integer variable $k \in \langle \tau \rangle$ plays the role of *the logical time variable*, which orders the transitions.

Definition 3. A hybrid trajectory $(\tau, \mathbf{q}, \mathbf{x})$ consists of a hybrid time set $\tau = \{I_k\}_{k=0}^N$ and two sequences of functions $\mathbf{q} = \{q_k(\cdot)\}_{k=0}^N$ and $\mathbf{x} = \{x_k(\cdot)\}_{k=0}^N$, $q_k(\cdot) : I_k \rightarrow \mathcal{Q}$ and $x_k(\cdot) : I_k \rightarrow X$.

Definition 4. An execution of a hybrid automaton H is a hybrid trajectory $(\tau, \mathbf{q}, \mathbf{x})$ satisfying the conditions:

1. initial state: $(q_0(\tau_0), x_0(\tau_0)) \in \text{Init}$,
2. discrete evolution: for all $k \in \langle \tau \rangle \setminus N$, $e_{k+1} = (q_k(\tau'_k), q_{k+1}(\tau_{k+1})) \in E$,
 $x_k(\tau'_k) \in G(e_{k+1})$ and $x_{k+1}(\tau_{k+1}) \in R(e_{k+1}, x_k(\tau'_k))$ and
3. continuous evolution: for all $k \in \langle \tau \rangle$ with $\tau_k < \tau'_k$,
 - $q_k(\cdot) : I_k \rightarrow \mathcal{Q}$ is constant over I_k , i.e. $q_k(t) = q_k(\tau_k)$, for all $t \in I_k$,
 - $x_k(\cdot) : I_k \rightarrow X$ is the solution on I_k of the differential equation $\dot{x}_k(t) = f(q_k^\tau(\tau_k), x_k(t))$, with initial condition $x_k(\tau_k)$, and
 - for all $t \in [\tau_k, \tau'_k)$, $\mathbf{x}_k(t) \in D(q_k(t))$.

For simplicity, it can be assumed that $\tau_0 = 0$. In context, it is convenient to think the guard $G(e)$ as *enabling* a discrete transition $e \in E$ - the execution *may* take a discrete transition $e \in E$ from a state x as long as $x \in G(e)$ - and one may think $D(q)$ as *forcing* discrete transitions - the execution *must* take a transition if the state is about to leave the domain [4]. Also, unlike continuous systems, an autonomous hybrid automaton can accept *multiple* executions from some initial state $(q_0(\tau_0), x_0(\tau_0))$, which is a feature of hybrid systems, related to nondeterminism and uncertainty.

Given an execution (τ, q, x) of a hybrid automaton H , the discrete trajectory of H is $\omega = q_0(\tau_0), q_1(\tau_1), \dots, q_k(\tau_k), \dots$ and, depending on N and on the structure of H , it may be finite, infinite, periodic. Also, given any $I_k \in \tau$ with $\tau_k < \tau'_k$, $x_k(t) = x_k(\tau_k) + \int_{\tau_k}^t f(q(\tau_k), x_k(\theta))d\theta$, $\forall t \in I_k$. Hence, it is possible that $x_k(\tau_k) \neq x_k(\tau_{k+1})$, and the model captures also the case of *continuous state jumps*, as defined in the Branicky classification: for any $k \in \tau >$, the continuous state of H evolves as $x_k(\cdot)$, for all $t \in (\tau_k, \tau'_k)$ and at $t = \tau'_k = \tau_{k+1}$ it instantly switches according to $x_k(\tau'_k) \mapsto x_{k+1}(\tau_{k+1})$.

It is convenient to represent hybrid automata as directed graphs (Q, E) , with the vertices of the graph corresponding to the discrete states and the set of edges E . Each vertex $q \in Q$ has an associated set of continuous initial states $\{x \in X \mid (q, x) \in \text{Init}\}$, a vector field $f(q, \cdot): X \rightarrow TX$ and a domain $D(q) \subseteq X$. An edge $(q, q') \in E$ starts at $q \in Q$ and ends at $q' \in Q$. With each edge $(q, q') \in E$ is associated a guard, $G(q, q') \subseteq X$ and a reset function $R(q, q', \cdot): X \rightarrow 2^X$.

The example below, classic in the literature [25], [27], is simple but relevant.

Example 1. The two tank system in Fig.1a consists of two tanks containing water. Both tanks are leaking at a constant rate. Water is added to the system at a constant rate through a hose, which at any time moment is dedicated either to one tank or to the other. It is assumed that the hose can switch between tanks instantaneously. x_1 and x_2 denote the water level and v_1 and v_2 are the (constant) water flows out of tank 1 and 2, respectively. w is the (constant) input flow. Assume that $x_1(0) \geq r_1$, $x_2(0) > r_2$. The control task is to keep the water levels x_1 and x_2 above the limits r_1 and r_2 , respectively, by an adequate switching policy. A solution is to switch the inflow w to tank 1, whenever $x_1 \leq r_1$, and to tank 2, whenever $x_2 \leq r_2$. The hybrid autonomous automaton defining the control system is represented in Fig.1b and is defined as follows:

- $Q = \{q_1, q_2\}$ the set of discrete states, inflow going left or right, respectively;
- $X = \mathbf{R}^2$ and the continuous state vector is $x = [x_1 \ x_2]^T$;
- $f(q_1, x) = [w - v_1 \ -v_2]^T$, $f(q_2, x) = [-v_1 \ w - v_2]^T$;
- $\text{Init} = \{\{q_1, q_2\} \times \{x \in \mathbf{R}^2 \mid x_1 > r_1 \wedge x_2 > r_2\}\} \cup \{\{q_1\} \times \{x \in \mathbf{R}^2 \mid x_1 \geq r_1 \wedge x_2 > r_2\}\} \cup \{\{q_2\} \times \{x \in \mathbf{R}^2 \mid x_1 > r_1 \wedge x_2 \geq r_2\}\}$;
- $D(q_1) = \{x \in \mathbf{R}^2 \mid x_2 \geq r_2\}$, $D(q_2) = \{x \in \mathbf{R}^2 \mid x_1 \geq r_1\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;

- $G(q_1, q_2) = \{x \in \mathbf{R}^2 \mid x_2 \leq r_2\}$, $G(q_2, q_1) = \{x \in \mathbf{R}^2 \mid x_1 \leq r_1\}$;
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$, i.e. the continuous state is unchanged by a discrete transition.

A simulated evolution of the hybrid system is depicted in Fig.2. The hybrid time set is $\tau = \{[0,2],[2,3],[3,3.5]\}$ and the discrete evolution is $\omega^\tau = q_1 q_2 q_1$.

3 Switching Inputs and Oscillating Systems - A Hierarchy of Two Automata

The concept of hierarchy of two hybrid automata is introduced next through an example.

3.1 The Components of the Hierarchy

Consider a second order continuous system defined by

$$\ddot{y} + 2\xi\dot{y} + y = u, \quad \xi \in (0, 1), \tag{3}$$

where $u : I \rightarrow \mathbf{R}$, $I \subseteq \mathbf{R}$, is a control signal. Consider also a choice of state variables $x_1 = y$, $x_2 = \dot{y}$ and the state vector $x = [x_1 \ x_2]^T$. Denote $X_1 = \mathbf{R}^2$ the state space, $U_1 \subseteq \mathbf{R}$ the set of control values and $Y_1 \subseteq \mathbf{R}$ the set of output values. The model (3) can be written in the form

$$\dot{x} = f(x, u), \quad y = g(x), \tag{4}$$

with $f : X_1 \times U_1 \rightarrow TX_1$, $f(x, u) = [x_2; -x_1 - 2\xi x_2 + u]^T$ and $g : X_1 \rightarrow Y_1$, $g(x) = x_1$. For constant input values $u(t) = \bar{u} \neq x_1(0)$, for any $t \in I$ - for example generated as step functions - the output y in (3) and (4) oscillates around a stationary value $\lim_{t \rightarrow \infty} y(t) = \bar{u}$.

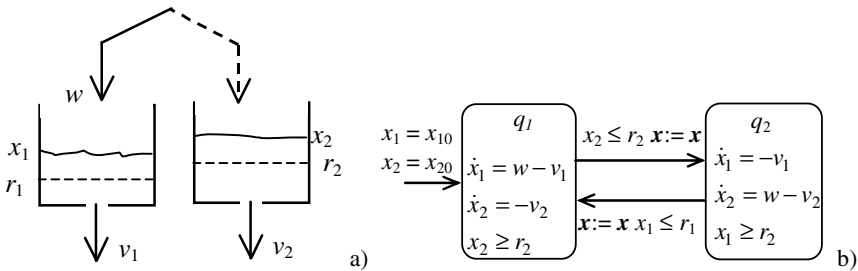


Fig. 1. a) The water tank system; b) the associated hybrid automaton

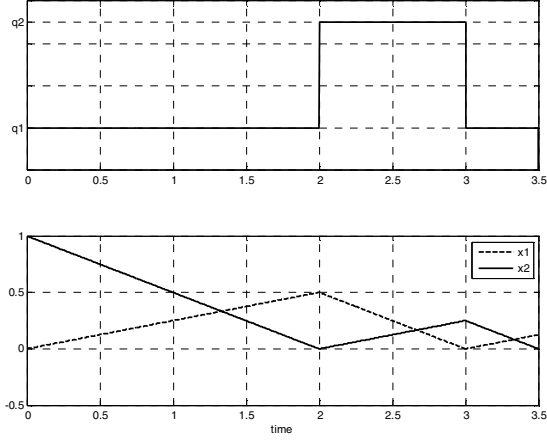


Fig. 2. Simulated evolution of the two water tanks hybrid system in Fig. 1: discrete trajectory (up) and continuous trajectory (down); $r_1 = r_2 = 0$, $v_1 = v_2 = 0.5$, $w = 0.75$, $q(0) = q_1$, $x(0) = [0 \ 1]^T$

This oscillating behaviour can be modelled by the hybrid automaton in Fig.3, denoted $H_1 = (Q_1, X_1, U_1, f_1, Init_1, D_1, E_1, G_1, R_1)$, with:

- $Q_1 = \{q_{11}, q_{12}\}$ the set of discrete states, x_1 below and above the stationary value \bar{u} , respectively;
- $X_1 = \mathbf{R}^2$ as above with $x = [x_1 \ x_2]^T \in X_1$;
- $U_1 = \{\bar{u} \in \mathbf{R}\}$, the set of constant input values;
- $f_1(\cdot, \cdot, \cdot): Q_1 \times X_1 \times U_1 \rightarrow TX$, $f_1(q_{11}, x, \bar{u}) = f(q_{12}, x, \bar{u}) = f(x, \bar{u})$;
- $Init_1 = \{q_{11}, q_{12}\} \times X_1 \times U_1$;
- $D_1(q_{11}) = \{x \in \mathbf{R}^2 \mid x_2 \leq \bar{u}\}$, $D_1(q_{12}) = \{x \in \mathbf{R}^2 \mid x_2 \geq \bar{u}\}$;
- $E_1 = \{(q_{11}, q_{12}), (q_{12}, q_{11})\}$;
- $R_1(q_{11}, q_{12}, x) = R_1(q_{12}, q_{11}, x) = \{x\}$, i.e. the continuous state is unchanged by a discrete transition and this is not represented in Fig. 3.

Consider a generator of a the piecewise constant signal $u: [0, \infty) \rightarrow \mathbf{R}$,

$$u(t) = \begin{cases} u_1, & 0 \leq t < t_1 \\ u_2, & t_1 \leq t < t_2, \\ u_3, & t \geq t_2 \end{cases} \quad (5)$$

and assume that $u_1 = 0$, $u_2 = 2$, $u_3 = 1$, $t_1 = 10$, $t_2 = 30$.

The signal generator can be modelled as a hybrid automaton denoted $H_0 = (Q_0, X_0, U_0, f_0, g_0, Init_0, D_0, E_0, G_0, R_0)$ (Fig. 4) with:

- $Q_0 = \{q_{01}, q_{02}, q_{03}\}$ the set of discrete states, each one associated to a time interval in (5);
- $X_0 = \mathbf{R}$, the state is $x_c \in X_0$;
- $U_0 = \{u_1, u_2, u_3\}$, the set of constant output values;
- $f_0(\cdot, \cdot): X_0 \rightarrow TX_0$, $f_0(q_{01}, x_c) = f_0(q_{02}, x_c) = f_0(q_{03}, x_c) = 1$;
- $g_0: Q_0 \rightarrow U_0$, $g_0(q_{0i}) = u_i$, $i = 1:3$, the output function ;
- $Init_0 = \{q_{01}\} \times X_0 \times \{u_1\}$;
- $D_0(q_{01}) = \{x_c \in \mathbf{R} \mid x_c \leq t_1\}$, $D_0(q_{02}) = \{x_c \in \mathbf{R} \mid x_c \leq t_2 - t_1\}$,
 $D_0(q_{03}) = \{x_c \in \mathbf{R} \mid x_c \geq t_2 - t_1\}$;
- $E_0 = \{(q_{01}, q_{02}), (q_{02}, q_{03})\}$;
- $G_0(q_{01}, q_{02}) = \{x_c \in \mathbf{R} \mid x_c > t_1\}$, $G_0(q_{02}, q_{03}) = \{x_c \in \mathbf{R} \mid x_c > t_2 - t_1\}$;
- $R_0(q_{01}, q_{02}, x_c) = R_0(q_{02}, q_{03}, x_c) = \{0\}$, i.e. after each transition the clock variable is reset.

The system (3)-(4) with the input signal (5) evolves, from the origin for $\xi = 0.5$ as in Fig.5 and for $\xi = 0.25$ as in Fig.6, respectively. Both simulation examples show that there are two distinct sequences of transitions:

1. one sequence corresponds to the switching input signal (5);
2. the other one corresponds to the internal oscillations of the second order system, with a frequency depending, between two consecutive switching inputs, not on the value of the input signal but on the damping parameter ξ .

However, in the controlled system evolution, the switching of the input is the primal cause of the system oscillations, so there is a causal hierarchy: the hybrid automaton H_0 resides at a ‘‘higher level’’ compared to H_1 . The problem is how to describe this hierarchy formally.

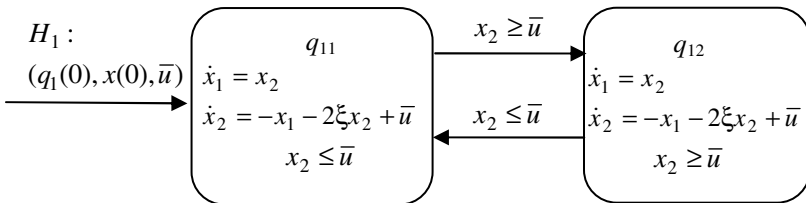


Fig. 3. The hybrid automaton H_1 , associated to the oscillating system (3)-(4)

$H_0 :$

$(q_0(0), x_c(0), u(0))$

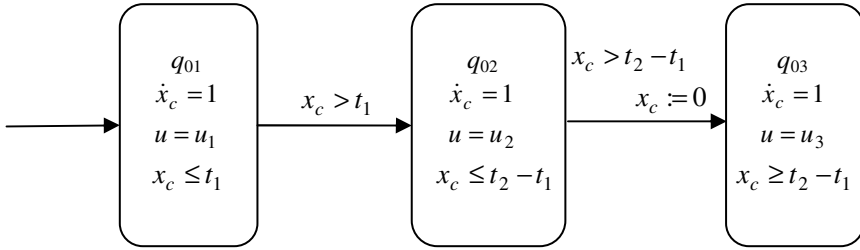


Fig. 4. The hybrid automaton modelling the time-driven signal generator (5); for simplicity, the notation of the output function $g_0(\cdot)$ replaced with the symbol u of the continuous time input signal in (3)-(4)

3.2 The Hierarchy

Two modelling sub-problems arise in the description of the hierarchy: 1) how to describe the communication between H_0 and H_1 ?; 2) how to specify the hybrid time set of the global hierarchical system? An approach to the first sub-problem is discussed below.

Denote $H_1(q_1(t), x(t), \bar{u})$ the hybrid automaton H_1 with a particular initialization $(q_1(t), x(t), \bar{u}) \in \text{Init}_1$ at some time instant $t \in \mathbf{R}$. It is obvious that, in open loop, the information flows in a single direction, from the higher priority automaton H_0 to H_1 , by a *change of initial conditions imposed to the control value u* , and this is the answer to the first modelling sub-problem.

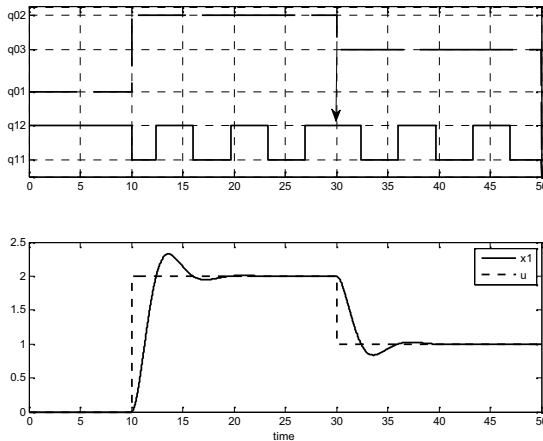


Fig. 5. Simulated execution with discrete evolution (up) and continuous evolution (down) of the hierarchical hybrid automaton $H_0 \times H_1$: in (3)-(4), $\xi = 0.5$ and the simulation time is $T = 50$

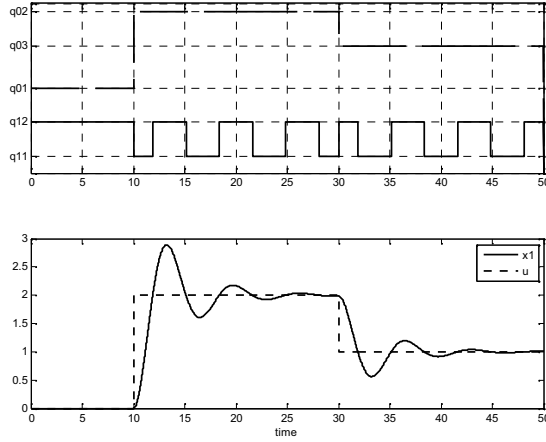


Fig. 6. Simulated execution with discrete evolution (up) and continuous evolution (down) of the hierarchical hybrid automaton $H_0 \times H_1$: in (3)-(4), $\xi = 0.25$ and the simulation time is $T = 50$

To formalize this, consider firstly *the hybrid time set* associated to the automaton H_0 ,

$$\tau^0 = \{I_0^0, I_1^0, I_2^0\}, \tag{6}$$

where

$$I_0^0 = [t_0, t_1] \equiv [0, 10], \quad I_1^0 = [t_1, t_2] \equiv [10, 30], \quad I_2^0 = [t_2, t_3] \equiv [30, \infty). \tag{7}$$

Consider the *hierarchy composed of H_0 and H_1* as an object $(H_0 \times H_1) = (H_0, \text{init}, H_1(\text{init}))$, where:

- H_0 is the hybrid automaton describing the signal generator (5),
- $\text{init}: Q_0 \rightarrow Q_1 \times X_1 \times U_0$, $q_{0i} \mapsto (q_1(t_i), \mathbf{x}(t_i), g_0(q_{0i}))$, $i = 1:3$, is the initialization function, associating the discrete higher level states to the initialization of the automaton H_1 and
- $H_1(\text{init})$ is the automaton H_1 with initial values specified by the function *init*.

The states of H_0 become macro-states and within each macro-state the lower level automation H_1 evolves with an associated initialization. Denote $(q_{0i}, q_{1j}) \in Q_0 \times Q_1$ a discrete state of the hierarchical model $(H_0 \times H_1)$. A graphical representation of $(H_0 \times H_1)$ is proposed in Fig. 7.

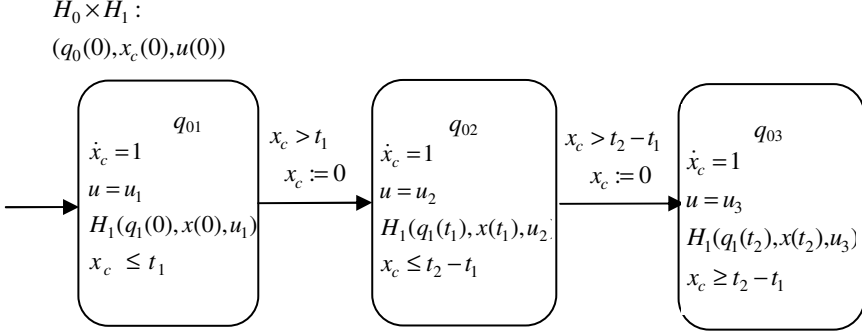


Fig. 7. The hierarchical hybrid automaton ($H_0 \times H_1$) modelling the system (3)-(4) with the time-driven input signal (5)

3.3 Hybrid Time Set of the Hierarchical System

In the sequel, as an introductory discussion concerning the second mentioned modeling sub-problem is proposed, based on intuitive simulation examples.

Consider the hybrid time set associated to the automaton H_1 ,

$$\tau^1 = \{I_k^1\}_{k=0}^N, \text{ with } I_k^1 = [\tau_k, \tau'_k], \quad (8)$$

and denote the hybrid time set of ($H_0 \times H_1$) by

$$\tau^{0 \times 1} = \{I_i^{0 \times 1}\}_{i=0}^{N_{0 \times 1}}, \text{ with } I_i^{0 \times 1} = [\tau_i, \tau'_i]. \quad (9)$$

To answer the second modelling sub-problem, compare firstly the evolutions depicted in Fig. 5 and in Fig. 6.

In Fig.5, at $t_2 = 30$, in H_0 occurs a transition $q_{02} \rightarrow q_{03}$, marked by an arrow, while H_1 is in the discrete state q_{12} . In consequence, the system ($H_0 \times H_1$) will transit to a new discrete state, $(q_{02}, q_{12}) \rightarrow (q_{03}, q_{12})$. Therefore, the current interval in (8), $I_1^m = [\tau_m, \tau'_m]$, $\tau_m < t_2 < \tau'_m$ has to be split into two new intervals, $[\tau_m, t_2]$ and $[t_2, \tau'_m]$, the logical time is incremented at t_2 , $m \mapsto m+1$, to keep track with the new transition and $N_{0 \times 1} = N+1$. Hence the hybrid time set of ($H_0 \times H_1$) is obtained by transforming τ^1 in (8) according to:

$$\tau^1 = \{I_k^1\}_{k=0}^N \mapsto \tau^{0 \times 1} = \{I_k^1\}_{k=0}^{m-1} \cup \{[\tau_m, t_2]\} \cup \{[t_2, \tau'_m]\} \cup \{I_k^1\}_{k=m+2}^N. \quad (10)$$

This situation doesn't occur in Fig. 6, where $\tau^{0 \times 1} = \tau^1$.

Summing up, if for any $k \in \{0:N\}$, there is no t_i , $i \in \{0:3\}$, such that $\tau_k < t_i < \tau_k'$, then $\tau^{0 \times 1} = \tau^1$. Else, the transformation similar with (10) is applied. In both situations, the hybrid time set τ^0 in (6), associated to the higher hierarchical level, *dominates* the hybrid time set τ^1 in (8), associated to the lower hierarchical level. This means that the transition moments in (8) depend on the damping factor ξ and, if the control input switches when H_1 resides within a location, it forces H_1 to leave the location and to restart its evolution with the new assigned control value, i.e. to execute a controlled transition. Moreover, the switching of the input signal at $t_1 = 10$ and $t_2 = 30$ are “enabling” the oscillations and this causality motivates the hierarchical structure.

4 Concluding Remarks

The hybrid model, called hierarchical hybrid automaton, proposed in this paper is based on classic autonomous hybrid automata and timed hybrid automata models.

The example of hierarchical hybrid automaton describes the behaviour of a second order system with piecewise constant input signal. The second order system has an intrinsic hybrid nature, due to the oscillations around the stationary output value and can be modelled as an autonomous hybrid automaton. The switching input signal “injects” exogenous hybrid behaviour, modelled as a timed automaton with outputs. The composition of the two behaviours is analysed in what concerns two special aspects: the communication within the hybrid hierarchy and the structure of the hybrid time set. A situation similar to the one considered in the paper occurs in the analysis of automotive systems behaviour. For example, power-train models including the nonlinearities due to the clutch dynamics receive piecewise constant active torques as test inputs [17].

It is interesting to compare the example in Section 3 with the classic autonomous hybrid automaton in Section 2, where the plant is not intrinsically hybrid, so the entire hybrid behaviour is due only to the switching control input. Another difference concerns the fact that, while the system in the example in Section 3 receives only open loop inputs, the example in Section 2 models a closed loop nonlinear control system. This emphasizes that the hierarchical model is different from the input-output automaton introduced in [13], centred on closed loop hybrid systems.

One can think a hierarchy of timed automata as modelling a scheme of inputs with associated priorities; the order of the priorities might be represented by the hierarchical order. A research direction is the study of hybrid automata models of continuous systems with intrinsic hybrid nature composed with a multi-level hierarchy of timed automata.

References

1. de Jong, H., Gouzé, J.-L., Hernandez, C., Page, M., Sari, T., Geiselman, J.: Hybrid Modeling and Simulation of Genetic Regulatory Networks: A Qualitative Approach. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 267–282. Springer, Heidelberg (2003)
2. Tomlin, C., Pappas, G., Sastry, S.S.: Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Trans. on AC* 43, 509–521 (1998)
3. Borgstede, M., Schicke, J.-W., Eggert, F., Golz, U.: Hybrid Automata as a Modeling Approach in the Behavioural Sciences. In: Proc. of the HAS Workshop HAS 2011, Saarbrücken (2011), <http://www.ips.cs.tu-bs.de/images/schicke/HAS2011.pdf>
4. Lygeros, J.: Lecture notes on hybrid systems. University of Patras, ENSIETA (2004), <http://citeseerx.ist.psu.edu>
5. Bemporad, A.: Hybrid Toolbox – User’s Guide (December 1, 2003), <http://www.dii.unisi.it/hybrid/toolbox>
6. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. *Automatica* 35, 407–427 (1999)
7. Stiver, J.A., Antsaklis, P.J., Lemmon, M.D.: A Logical DES Approach to the Design of Hybrid Control Systems. Technical Report of the ISIS Group at the University of Notre Dame, ISIS-94-011 (1994)
8. Henzinger, T.A., Ho, P.H., Wong, T.H.: A User Guide to HYTECH. In: Brinksma, E., Steffen, B., Cleaveland, W.R., Larsen, K.G., Margaria, T. (eds.) TACAS 1995. LNCS, vol. 1019, pp. 41–71. Springer, Heidelberg (1995)
9. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. *Theoretical Comp. Science* 138, 3–34 (1995)
10. Lygeros, J., Tomlin, C., Sastry, S.S.: Controllers for reachability specifications of hybrid systems. *Automatica* 35(3), 349–370 (1999)
11. Tomlin, C., Lygeros, J., Sastry, S.S.: A game theoretic approach to controller design for hybrid systems. *Proc. of the IEEE* 88(7), 949–970 (2000)
12. Lygeros, J., Johansson, K.H., Simić, S.N., Zhang, J., Sastry, S.S.: Dynamical Properties of hybrid automata. *IEEE Trans. on AC* 48(1), 2–17 (2003)
13. Lynch, N.A., Segala, R., Vaandrager, F.W.: Hybrid I/O Automata Revisited. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 403–417. Springer, Heidelberg (2001)
14. Kiencke, U., Nielsen, L.: Automotive control systems: for engine, driveline, and vehicle, 2nd edn. Springer (2005)
15. Sontag, E.: Nonlinear regulation: the piecewise linear approach. *IEEE Trans. on AC* 26(2), 346–358 (1981)
16. Johansson, M., Rantzer, A.: Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. on AC* 43(4), 555–559 (1998)
17. Bălău, A.E., Căruntu, C.F., Lazăr, C.: Driveline oscillations modeling and control. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, CSCS, vol. 18, pp. 134–139 (2011)
18. Alur, R., Henzinger, T., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. *Proc. of the IEEE* 88(7), 971–984 (2000)

19. Oltean, V.E., Dobrescu, R., Popescu, D.: On a hierarchy of hybrid automata models – an example. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, CSCS, vol. 18, pp. 905–913 (2011)
20. Di Caraiano, S., Bemporad, A.: Equivalent piecewise affine models of linear hybrid automata. *IEEE Trans. on AC* 55(2), 498–502 (2010)
21. Branicky, M.S., Borkar, V.S., Mitter, S.K.: A unified framework for hybrid control: background, model and theory. Technical report LIDS-P-2239, Laboratory for Information and Decision Systems, MIT (1994)
22. Branicky, M.S.: Introduction to hybrid systems. In: Hristu-Varsakelis, D., Levine, W.S. (eds.) *Handbook of Networked and Embedded Control Systems*, pp. 91–116. Springer, Birkhäuser (2005)
23. Arnold, V.I.: *Ecuatii diferențiale ordinare*. Editura științifică și Enciclopedică, București (1978)
24. Khalil, H.K.: *Nonlinear systems*, 2nd edn. Prentice Hall (1995)
25. Lygeros, J., Sastry, S., Tomlin, C.: *Hybrid systems: foundations, advanced topics and applications* (2010), <http://control.ee.ethz.ch/~ifaatic/book.pdf>
26. Tomlin, C.: *Hybrid systems: Modelling, analysis and control*. AA278A Lecture notes. Stanford University (2005), <http://www.stanford.edu/class/aa278a/>
27. Johansson, K.H., Egerstedt, M., Lygeros, J., Sastry, S.: On the regularization of Zeno hybrid automata. *Systems and Control Letters* 38, 141–150 (1999)

Improvement of Statistical and Fractal Features for Texture Classification

Dan Popescu¹, Radu Dobrescu¹, and Nicoleta Angelescu²

¹Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest, 313 Splaiul Independentei, Bucharest, Romania

²Faculty of Electrical Engineering, Valahia University of Targoviste, Targoviste, Romania
{dan_popescu_2002, rd_dobrescu, nicoletaangelescu}@yahoo.com

Abstract. Texture classification and segmentation have been studied using various approaches. The mean Grey-Level Co-occurrence Matrix, introduced by the authors, gives statistical features relatively insensitive to rotation and translation. On the other hand, texture analysis based on fractals is an approach that correlates texture coarseness and fractal dimension. By combining the two types of features, the discrimination power increases. The paper introduces the notion of effective fractal dimension which is an adapting fractal dimension to classification of texture and is calculated by elimination of a constant zone which appears in all textured images. In the case of colour images, we proposed a classification method based on minimum distance between the vectors of the effective fractal dimension of the fundamental colour components. The experimental results to classify real land textured images validate that effective fractal dimension offers a grater discrimination of classes than typical fractal distance based on complete box counting algorithm.

Keywords: texture, fractal dimension, box-counting algorithm, statistical features, image processing, texture classification.

1 Introduction

Textures in images, which are characterized by varying spatial intensity and colour of pixels, are useful in a variety of applications [1], [2], [3] like: classification of remotely sensed images, defect detection, medical image processing, robot/vehicle navigation, document processing, content-based image retrieval etc. Texture analysis has been an active research topic for more than four decades and has been studied by various approaches, most of which were used statistical methods [4], [5]: characteristics associable with grey level histogram, grey level image difference histogram, co-occurrence matrices and the features extracted from them, autocorrelation based features, power spectrum, edge density per unit of area etc. To develop precise techniques for analysis of textured images, both second order statistical features and fractal type features can be combined into a characteristic vector.

Many natural textured surfaces have a statistical quality of roughness and, sometimes, self-similarity at different scales. Pentland [6] demonstrated a correlation between texture coarseness and fractal dimension of a texture. For this reason, fractals were very useful in modelling texture properties in image processing and became popular for different applications of texture analysis. For example, the fractal dimension has been proven to be efficient in classifying natural textures [7].

The most common algorithm to evaluate the fractal dimension is the box-counting type [8]. It is one of the more widely used because it can be computed simply. For example, the classification method based on box counting algorithm implies a less calculus amount than the method based on the co-occurrence matrices. Box-counting analysis can be used to estimate the fractal dimensions of textured images with or without self-similarity.

Kaplan [9] evaluates the effectiveness of other fractal type characteristics named Hurst parameters as features for texture classification and segmentation. Thus, the segmentation accuracy using generalized and standard Hurst features is evaluated on images of texture mosaics. Reference [10] presents an algorithm to estimate the Hurst exponent of fractals with arbitrary dimension, based on the high-dimensional generalized variance. More recently, Li [11] presents an efficient box-counting based method for the improvement of estimation accuracy of fractal dimension. A new model is proposed to assign the smallest number of boxes to cover the entire image at each selected scale as required, thereby yielding more accurate estimates.

Because the fractal dimension is the most used fractal-type descriptor for texture analysis, the paper introduces and analyses some estimates of fractal dimensions for grey level and colour textures. Theoretical statements are validated by experimental results on real textured images.

In many practical applications, it is assumed implicitly that texture analysis of images captured is invariant to translations, rotations or scaling [12]. Therefore features for texture classification, proposed bellow, will have such invariant (for example, the features extracted from the average co-occurrence matrix [13]). This paper introduces the notion of effective fractal dimension which is an adapting fractal dimension to classification of texture and is calculated by elimination of a constant zone which appears in all textured images.

The work is organized as follows. Section 2 presents the most important features used in texture classification and thus segmentation: co-occurrence and fractal dimension type. Section 3 describes a more efficient method, based on box-counting algorithm, to estimate fractal dimension in textured image case. Section 4 reports some experimental results. Section 5 offers a brief conclusion.

2 Statistic and Fractal Features in Texture Classification and Segmentation

Texture segmentation and texture classification are two important problems that texture analysis research attempts to solve. Both require successive comparisons between the properties (features) of textured images. The classification process

can be used to segment images with different textured regions. Generally, the result is a coarse type of segmentation process. The segmentation fineness depends on the degree of partition of the initial images. If the degree of partition is too fine, then it is possible that the texture could disappear. Actually, for an application, a partition index is established taking into account the given image resolution and the texture fineness [13]. For example, the case study presented in [14] is related to a method of segmentation of the road image I_l , based on multiple comparisons of the textured regions. In order to follow the road, the asphalt texture is considered as the reference texture for a classification process. The application goal is to identify the asphalt regions and to produce an asphalt localization matrix by recognition techniques.

The most common statistical method for textured image analysis is based on features extracted from the Grey-Level Co-occurrence Matrix (GLCM), proposed by Haralick in 1973 [5]. Among these features, the most important are: contrast C , energy E , entropy Et , and homogeneity O . The contrast measures the coarseness of texture, because large values of contrast correspond to large local variation of the grey level. The entropy measures the degree of disorder or non-homogeneity. Large values of entropy correspond to uniform GLCM. The energy is a measure of homogeneity. The features implied in the classification process can differ from one application to another.

In order to obtain GLCM-based features, relatively insensitive to rotation and translation, in [14] the authors introduce the notion of *mean co-occurrence matrix*. So, for each pixel we can consider $(2d+1) \times (2d+1)$ symmetric neighbourhoods, $d = 1, 2, 3, \dots, n$. Inside each neighbourhood there are eight principal directions: 1, 2, 3, 4, 5, 6, 7, 8 (corresponding to $0^\circ, 45^\circ, \dots, 315^\circ$) and we evaluate the co-occurrence matrices $N_{d,k}$ corresponding to the displacement d determined by the central point and the neighbourhood edge point in the k direction ($k = 1, 2, \dots, 8$). For each neighbourhood type (d fixed), the *mean co-occurrence matrix* $CM_d(1)$ is calculated by averaging the eight co-occurrence matrices $N_{d,k}(1)$:

$$CM_d = (N_{d,1} + N_{d,2} + N_{d,3} + N_{d,4} + N_{d,5} + N_{d,6} + N_{d,7} + N_{d,8})/8, \quad d=1,2,\dots \quad (1)$$

Thus, for 3×3 neighbourhoods, $d = 1$, for 5×5 neighbourhoods, $d = 2$, and so on.

The same features as in the normal case (without averaging and therefore depending on rotation and translation) can be extracted from the mean co-occurrence matrix CM_d : C , E , Et and O . Evidently, they depend on the displacement d .

It can be easily observed that the addition of information about colour components increases the classification efficiency because the colour and statistical texture features have complementary roles. For colour texture classification, colour and texture features must be extracted separately and then combined in the Euclidian distance evaluation. The algorithms for the colour components (R , G , B or H , S , V) are the same as in the grey level case.

In different applications for classification or segmentation, not all features have suitable discriminatory properties. Because texture has many different dimensions and characteristics there is not a single method of texture representation and

classification that is everywhere adequate. Therefore statistic different methods including grey level histograms and spatial autocorrelation functions and also fractal analysis techniques are often used. Although textures and fractals refer to different things, methods for determining characteristics of fractal can be applied to texture classification.

Fractal description of textures is typically based on evaluation of fractal dimension and lacunarity to measure texture roughness and granularity. A multi-resolution feature vector, based on Hurst coefficients derived from pyramidal images, can describes both texture roughness and granularity [15].

Because it is a quite simply method of calculation, the box-counting algorithm is the most frequently used technique to estimate the fractal dimension (FD) of an image. For the box counting basic algorithm, the image must be binary. The method consists in dividing the image successively in 4, 16, 64 etc. If $(1/r)$ is the order of the dividing process on x and y axes (lattice of grid size r) and $N(r)$ is the number of the same size squares covered by the object image (containing one or more pixels with value 1), then the box counting dimension D (2) is related to the number $N(r)$ and the ratio r as follows [16]:

$$D = \lim_{r \rightarrow 0} \frac{\log N(r)}{\log (1/r)}. \quad (2)$$

Usually, the fractal dimension (FD) can be obtained by plotting $\log N(r)$ for different values of $\log(1/r)$ and calculating the coefficient a (4) of the corresponding regression line (3) [17]:

$$y = a x + b. \quad (3)$$

$$a = FD = \frac{n \sum_{i=1}^n x_i y_i - \left(\sum_{i=1}^n x_i \right) \left(\sum_{i=1}^n y_i \right)}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2}. \quad (4)$$

In the grey level (or monochrome) case, an average fractal dimension (AFD) is proposed [18]. The average is made on several fractal dimensions (5) calculated for binary edge-type images obtained from the original image for specified segmentation thresholds $T_j, j = 1, 2, \dots, k$:

$$AFD = \frac{1}{k} \sum_{j=1}^k FD_j. \quad (5)$$

Each threshold T_j corresponds to contour image CI_j . For each image CI_j , the fractal dimension FD_j is calculated using the box-counting algorithm. Choice of thresholds for binarization is a delicate issue because they influence the average fractal dimension. Note that the graphs in Fig.1 represent the dependence between FD and threshold level j . Several ways (W1, W2, and W3) of choosing the binarization thresholds for the calculation of AFD are presented below [17]:

W1. All levels of monochrome image representation are taken into account (j between i and k , Fig. 1a).

W2. Only grey levels with nonzero frequency in the histogram representation are taken into account (j between i and k , Fig. 1b). It is just an average size of the nonzero values of fractal dimensions. The resulting AFD is bigger than the case W1.

W3. Only levels of grey which hold mostly original texture appearance are taken into account (j between i and k , Fig. 1c). It is only average of values of the plateau-type region of the fractal dimension representation (almost constant values).

Figure 2 shows the influence of threshold on the appearance of the texture edges: a) original image, b) contour image of a properly chosen threshold (possible in case W3), c) contour image of an incorrectly chosen threshold (possible in cases W1 and W2).

The chain of primary image processing operations [13] that are necessary for evaluation of fractal dimension consists of:

- i) Edge detection by local median filter.

For the special cross neighborhood (Fig.3) the relation which describes the function's filter is the following (6):

$$f_{ij} = Me \{g_{i-1,j}, g_{i,j-1}, g_{i,j}, g_{i,j+1}, g_{i+1,j}\}, \tag{6}$$

where g_{ij} is the gray level of the point (i,j) in the original image and f_{ij} is the filter output. So, the median filter does not affect the image with less than 3×3 pixels details.

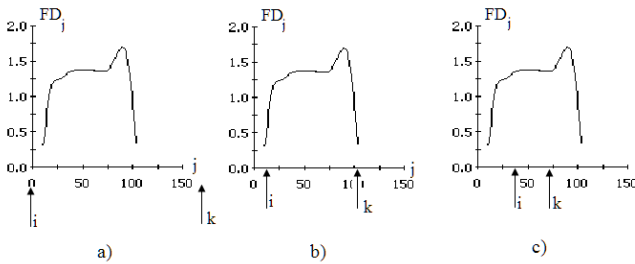


Fig. 1. Choice of threshold limits for j : i (minimum) and k (maximum)

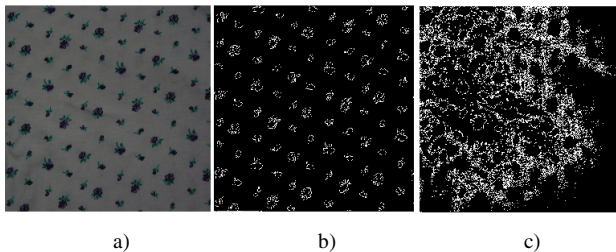


Fig. 2. Influence of the threshold on the appearance of texture

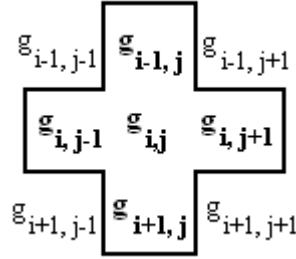


Fig. 3. Neighborhood for local median filter/ edge detection

ii) Conversion of monochromatic image (gray level) in binary image.

Towards edge extraction, we propose a logical function based algorithm. First, the image from the median filter is gone over into binary form by a threshold comparison (7). The threshold T is determined by one way $W1$, $W2$ or $W3$ and the pixels of the binary image are noted by $b_{i,j}$. Thus,

$$\begin{aligned} b_{i,j} &= 1, \text{ if } f_{i,j} \geq T, \text{ and} \\ b_{i,j} &= 0, \text{ if } f_{i,j} < T \end{aligned} \quad (7)$$

iii) Edge detection.

The matrix representation of the binary image is analyzed in 3×3 neighborhoods, like in the noise rejection case (Fig.3) in order to detect a “1” in the central position and at list a “0” in rest. The central point $b_{i,j}$ is replaced by resulting operator’s value $c_{i,j}$. Adequate to the neighborhood form, for the edge detection algorithm, two logical function expressions are possible (7), (8):

$$c_{i,j} = b_{i,j} \cdot (\bar{b}_{i-1,j-1} + \bar{b}_{i-1,j} + \bar{b}_{i-1,j+1} + \bar{b}_{i,j-1} + \bar{b}_{i,j+1} + \bar{b}_{i+1,j-1} + \bar{b}_{i+1,j} + \bar{b}_{i+1,j+1}), \quad (6)$$

$$c_{i,j} = b_{i,j} \cdot (\bar{b}_{i-1,j} + \bar{b}_{i,j-1} + \bar{b}_{i,j+1} + \bar{b}_{i+1,j}). \quad (8)$$

The algorithm for calculating AFD, which was implemented in MATLAB, consists of the following steps:

1. Reading and converting of the color image in 256 grey levels;
2. Converting of the 256 grey levels image to a binary level image using a fixed threshold T_j ;
3. Extraction of the image contour using 3×3 neighborhoods;
4. Computing of the fractal dimension FD_j , from the contour image, applying the box-counting algorithm;
5. Iteration of the steps 1-4, for $j = 1, \dots, k$;
6. Determination of ADF from equation (5).

For colour images, the process is to apply three times, for each colour component (R , G , B or H , S , V). The characteristics AFD_R , AFD_G and AFD_B (or AFD_H , AFD_S and AFD_V) are utilized as features in the texture classification process.

3 Effective Fractal Dimension for Texture Description

In the particular case of the fractal dimension evaluation by box counting algorithm applied to textured images, one can see that at the beginning of the algorithm, all the boxes contain points of the edges. This means that the corresponding values of the slope have the value 2. We propose to calculate the fractal dimension, named effective fractal dimension (EFD), similar to (4) by omitting the first points in the log-log representation (the points of the form $(x_b, 2x_b)$, $b = 1, 2, \dots, m$) which are present in all log-log representations of the classical box-counting algorithm, regardless of texture.

The result is a modified box-counting algorithm [17], with fewer points in the log-log representation ($n - m$ points). The coefficient a_E (effective fractal dimension) of the corresponding regression line is given by (9) and obviously involves a smaller volume of calculation. Because the slope portions of maximum value (equal 2) have been removed, EFD is less than FD . Figure 4 illustrates the simply case corresponding to five points (O, A, B, C and D) in the log-log representation. FD is the slope of the regression line for the points O, A, B, C, D (slope OA equal to 2 and slopes OA, OB, OC, OD smaller than 2) and EFD is the slope of the regression line determined by A, B, C, D . We can see by inspecting Fig. 4 that EFD is smaller than FD . However, being strongly dependent on texture content, EFD has a discriminatory power higher than FD (16).

$$a_E = EFD = \frac{(n-m) \sum_{i=m+1}^n x_i y_i - \left(\sum_{i=m+1}^n x_i \right) \left(\sum_{i=m+1}^n y_i \right)}{(n-m) \sum_{i=m+1}^n x_i^2 - \left(\sum_{i=m+1}^n x_i \right)^2} \quad (8)$$

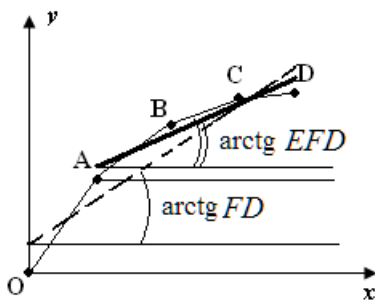


Fig. 4. Case illustrates EFD versus FD

In order to exemplify the difference between EFD and FD [17], we can consider the log-log representation for image $I4$ (Fig.8). Let v the division vector (values of $1/r$) along the horizontal and the vertical coordinates, and w the corresponding vector of $N(r)$ values (9):

$$\begin{aligned} v &= [2 \quad 4 \quad 8 \quad 16 \quad 32 \quad 64 \quad 128 \quad 256 \quad 512] \\ w &= [4 \quad 16 \quad 64 \quad 256 \quad 1024 \quad 4079 \quad 13621 \quad 30138 \quad 51816]. \end{aligned} \quad (9)$$

Italics represent *the effective points*, for which $w < x^2$ or $\log_2 w < 2\log_2 x$.

If:

$$x = \log_2 v, y = \log_2 w, \quad (10)$$

then:

$$\begin{aligned} x &= [0.301 \quad 0.602 \quad 0.903 \quad 1.202 \quad 1.505 \quad 1.806 \quad 2.107 \quad 2.408 \quad 2.709] \\ y &= [0.602 \quad 1.202 \quad 1.806 \quad 2.408 \quad 3.010 \quad 3.610 \quad 4.134 \quad 4.479 \quad 4.714]. \end{aligned} \quad (11)$$

Fractal dimension FD is calculated by equation (4), where $n = 9$, from x and y vectors, and the numerical result is $FD = 1.781$. The log-log diagram is presented in Fig. 5.a.

For $i = 1,2,3,4,5$, one can observe that $y(i) = 2x(i)$, i.e. the slope is 2. If we disregard these points, it is obtained two shorter set of points (12) x_I and y_I , from which it is calculated EFD . The new log-log diagram is presented in Fig.5.b. For EFD , the algorithm is (5), with $n = 9$ and $m = 5$.

$$\begin{aligned} x_I &= [1.806 \quad 2.107 \quad 2.408 \quad 2.709] \\ y_I &= [3.610 \quad 4.134 \quad 4.479 \quad 4.714]. \end{aligned} \quad (12)$$

The resulting distance EFD is less than FD : $EFD = 1.215$. Also we can see by Fig.5 that EFD is smaller than FD .

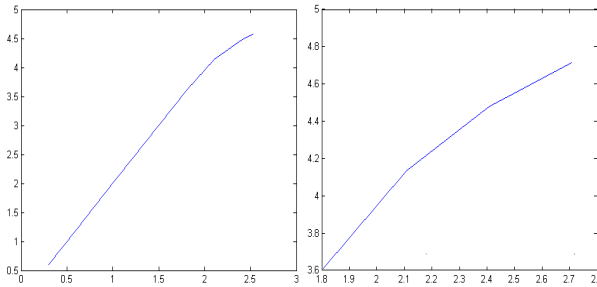


Fig. 5. a) Log-log diagram for FD ; b) Log-log diagram for EFD

4 Experimental Results

The experimental results were obtained using an original software system developed by the authors in two integrated development environments: Visual C++ and Matlab. The interface is composed of three application forms available from the *View menu*: *FrTex* (used to extract statistical type texture features for the classification of textured images), *Fractal* (used to evaluate FD , AFD and EFD for textured images) and *Fractal RGB* (use to extract fractal type features of the colour components R , G and B and to classify the textures). The screen capture of the application *Fractal RGB* is presented in Fig. 6.

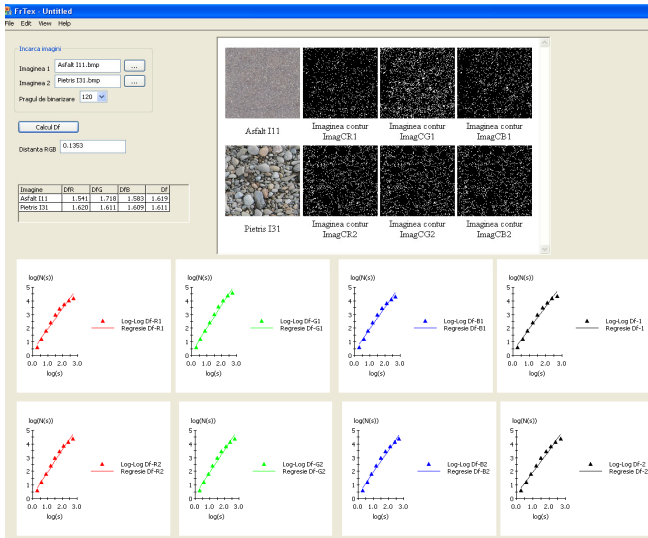


Fig. 6. Screen capture of the application (*Fractal RGB*)

First we studied the influence of the threshold limits (W_1 , W_2 , and W_3) on the calculation of fractal dimension. We considered three textured images I_1 , I_2 and I_3 (Fig. 7) and two ways of selecting binarization thresholds, W_2 and W_3 . The resulting average fractal dimensions, AFD_2 and AFD_3 are presented in Table 1.

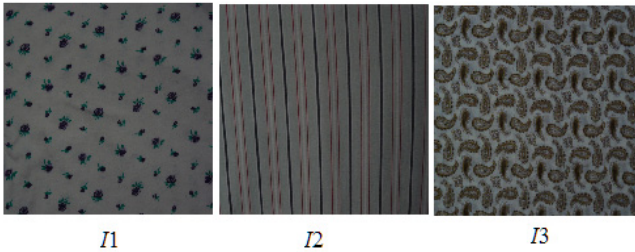


Fig. 7. Textured images to study the efficiency of the elements of average in AFD

Table 1. Results for AFD in cases W_2 (AFD_2) and W_3 (AFD_3)

Image	$T_i \div T_k$ Nonzero fractal dimension	AFD_2	$T_i \div T_k$ Plateau area	AFD_3
I_1	$7 \div 102$	1,29	$30 \div 70$	1,35
I_2	$14 \div 113$	1,39	$40 \div 60$	1,54
I_3	$11 \div 125$	1,45	$40 \div 80$	1,71

AFD efficiency of separating classes of textures is observed by calculating the relative differences of fractal dimensions for images I_1 , I_2 and I_3 as shown below (13) – (18).

$$\frac{AFD_2(I_2) - AFD_2(I_1)}{AFD_2(I_2)} = 7,19 \% \quad (13)$$

$$\frac{AFD_3(I_2) - AFD_3(I_1)}{AFD_3(I_2)} = 12,34 \% \quad (14)$$

$$\frac{AFD_2(I_3) - AFD_2(I_2)}{AFD_2(I_3)} = 4,14 \% \quad (15)$$

$$\frac{AFD_3(I_3) - AFD_3(I_2)}{AFD_3(I_3)} = 9,94 \% \quad (16)$$

$$\frac{AFD_2(I_3) - AFD_2(I_1)}{AFD_2(I_3)} = 11,03 \% \quad (17)$$

$$\frac{AFD_3(I_3) - AFD_3(I_1)}{AFD_3(I_3)} = 21,05 \% \quad (18)$$

It can be seen that W3 is a more efficient way than W2 for choosing which *FD* values are averaged (relative differences (14), (16) and (18) are greater than relative differences (13), (15) and (17), respectively).

With the purpose of showing the efficiency of *EFD* in colour texture classification, we considered the images and its contours in Fig.8: I_4 - asphalt, I_5 - grass, I_6 - stone.

Each analyzed image is decomposed in its fundamental colour: Red - *R*, Green - *G*, and Blue - *B*. From these components we calculated the fractal dimension vectors [*FD*] (19) and [*EFD*] (20), where:

$$[FD] = [FD_R, FD_G, FD_B], \quad (19)$$

$$[EFD] = [EFD_R, EFD_G, EFD_B]. \quad (20)$$

For each component, the algorithm is similar to the grey level case: (4) for *FD* and (8) for *EFD*. The experimental results are presented in Table 2.

To evaluate the efficiency in texture discrimination, we calculated the following Euclidian distances:

a) Between I_1 and I_2 for *FD* ($D_f(I_1, I_2)$) and for *EFD* ($D_{ef}(I_1, I_2)$) resulting:

$$D_f(I_1, I_2) = 0,190, \quad D_{ef}(I_1, I_2) = 0,295;$$

b) Between I_1 and I_3 for *FD* ($D_f(I_1, I_3)$) and for *EFD* ($D_{ef}(I_1, I_3)$), resulting:

$$D_f(I_1, I_3) = 0,280, \quad D_{ef}(I_1, I_3) = 0,414.$$

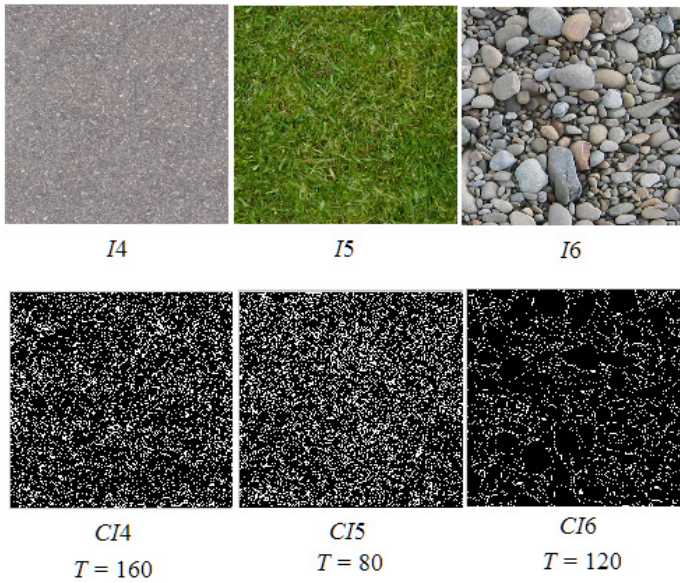


Fig. 8. Test images for evaluating the efficiency of *EFD* in colour case

Table 2. Fractal dimension and effective fractal dimension for colour components of the images I_1, I_2, I_3

Image	Threshold (T)	Fractal dimension FD	Effective fractal dimension EFD
$I_1 - R$	160	1.781	1.215
$I_1 - G$	160	1.680	0.967
$I_1 - B$	160	1.674	0.956
$I_2 - R$	80	1.804	1.295
$I_2 - G$	80	1.745	1.369
$I_2 - B$	80	1.403	1.010
$I_3 - R$	120	1.614	1.175
$I_3 - G$	120	1.610	1.169
$I_3 - B$	120	1.608	1.172

It can be observed that:

- *EFD* is less than the corresponding *FD*,
- The distances between two images with different textures are greater in the *EFD* case than in the *FD* case. Therefore, *EFD* offers a discriminatory power greater than *FD*.

5 Conclusions

Starting from two types of features widely used in texture classification namely features based on co-occurrence matrices and fractal dimension, we developed similar features which are primarily independent of rotation and secondly more efficient. Average of fractal dimensions for significant binary thresholds (AFD_3) and effective fractal dimension (EFD), which were introduced in this paper, both in the grey level case and also in the colour case, give good results in texture classification. We can observe that both improved fractal dimensions have a discriminatory power in texture classification greater than current fractal dimension, and they are easier to assess. The threshold assessment which is used for edge extraction constitutes a problem for the fractal dimension evaluation in the grey level image case. Our approach was primarily motivated by the requirement of simplicity in feature extraction and the underlying hardware. But, in spite of its computational efficiency, the regular partition scheme used by various box-counting methods intrinsically produces less accurate results than other methods. Therefore, we intend to extent the research to a novel multi-fractal estimation algorithm based on mathematical morphology to characterize the local scaling properties of textures.

References

1. Tuceryan, M., Jain, A.: Texture Analysis. In: Chen, C.H., Pau, L.F., Wang, P.S.P. (eds.) The Handbook of Pattern Recognition and Computer Vision, 2nd edn., pp. 207–248. World Scientific Publishing Co. (1998)
2. Pesaresi, M.: Texture Analysis for Urban Pattern Recognition Using Fine-resolution Panchromatic Satellite Imagery. *Geographical and Environmental Modelling* 4(1), 43–63 (2000)
3. Olujic, M., Milosevic, N., Oros, A., Jelinek, H.: Aggressive Posterior Retinopathy of Prematurity: Fractal Analysis of Images before and after Laser Surgery. In: Proc. of 18th Int. Conf. on Control Systems and Computer Science, pp. 877–882. Politehnica Press, Bucharest (2011)
4. Shapiro, L., Stockman, G.: *Computer Vision*. Prentice Hall (2001)
5. Haralick, R.M., Shanmugam, K., Dinstein, I.: Texture Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics* 3(6), 610–621 (1973)
6. Pentland, A.P.: Fractal based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, 661–674 (1984)
7. Keller, J.M., Chen, S., Crowner, R.M.: Texture Description and Segmentation through Fractal Geometry. *Computer Vision, Graphics and Image Processing* 45, 150–166 (1989)
8. Peitgen, H.O., Jurgens, H., Saupe, D.: *Chaos and Fractals: New Frontiers of Science*. Springer, New York (1992)
9. Kaplan, L.M.: Extended fractal analysis for texture classification and segmentation. *IEEE Transactions on Image Processing* 8(11), 1572–1585 (1999)

10. Carbone, A.: Algorithm to estimate the Hurst exponent of high-dimensional fractals. *Physical Review E* 76 056703/1 - 056703/7 (2007)
11. Li, J., Du, Q., Sun, C.: An improved box-counting method for image fractal dimension estimation. *Pattern Recognition* 42(11), 2460–2469 (2009)
12. Zhang, J., Tan, T.: Brief review of invariant texture analysis methods. *Pattern Recognition* 35, 735–747 (2002)
13. Dobrescu, R., Popescu, D.: Image processing applications based on texture and fractal analysis. In: Qahwaji, R., Green, R., Hines, E. (eds.) *Applied Signal and Image Processing: Multidisciplinary Advancements*, pp. 226–250. IGI Global Publishing (2011)
14. Popescu, D., Dobrescu, R.: Carriage road pursuit based on statistical and fractal analysis of the texture. *International Journal of Education and Information Technologies* 2(11), 62–70 (2008)
15. Wu, C.M., Chen, Y.C., Hsieh, K.S.: Texture features for classification of ultra-sonic liver images. *IEEE Transactions on Medical Imaging* 11, 141–152 (1992)
16. Mandelbrot, B.B.: *Fractals: Form, Chance and Dimension*. W.H. Freeman and Company, San Francisco (1977)
17. Popescu, D., Dobrescu, R., Angelescu, N.: Fractal Analysis of Textures Based on Modified Box-Counting Algorithm. In: *Proc. of 18th Int. Conf. on Control Systems and Computer Science*, pp. 894–898. Ed. Politehnica Press, Bucharest (2011)
18. Popescu, D., Dobrescu, R., Angelescu, N.: Colour textures discrimination of land images by fractal techniques. In: *Proc. 4th Int. Conf. REMOTE 2008, Venice*, pp. 51–56 (2008)

Towards a PIO II Criterion: Improving the Pilot Modeling

Adrian Toader and Ioan Ursu

“Elie Carafoli” National Institute for Aerospace Research,
Blvd. Iuliu Maniu, 220, 061126, Bucharest, Romania
{atoader, iursu}@incas.ro

Abstract. In this paper, a study is performed from the perspective of giving a methodology to analyze and predict the emergence of PIO (Pilot-Induced Oscillation) phenomenon. More precisely, a proper procedure of human pilot mathematical model synthesis in order to analyze PIO II type susceptibility of a VTOL-type aircraft, related with the presence of position and rate-limited actuator, is considered. The mathematical tools are those of LQG control synthesis and semi-global stability theory developed in recent works.

Keywords: Pilot-Induced-Oscillation, Hess pilot mathematical model, position and rate-limited actuator, limit cycles, semi-global stabilization.

1 Introduction

“Pilot-Induced-Oscillation” (PIO) is a phenomenon usually due to adverse aircraft-pilot coupling during some tasks in which “tight closed loop control of the aircraft is required from the pilot, with the aircraft not responding to pilot commands as expected by the pilot himself” [1]. Predicting PIO is difficult and becomes even more difficult with the advent of new technologies such as active control and fly-by-wire flight control systems. According to common references (see, for example, [2]), PIOs are categorized depending essentially on the degree of nonlinearity in the event. In the category PIO II, quasi-linear oscillations result mainly from rate and/or position saturation of the actuator.

Undoubtedly, to have at hand a mathematical model of pilot behavior is very important for deriving a PIO prognostic theory. A recent work [3] highlighted the main steps of deriving a complex model of human pilot, as developed by Davidson and Schmidt [4]. A numerical validation of the obtained model in terms of PIO I prognostic was performed upon the concrete case of a hovering VTOL-type aircraft analyzed in the classical reports [5], [6], [7], [8].

The present paper retakes the problem of deriving the human pilot mathematical model from the perspective of approaching a PIO II prevention criterion. It is known (see [9]) that mathematical methodologies are more “rigid” than mathematical models. So, our intention, herein and in future works, is to treat

realistic presence of the position and rate saturations in the plant model, by adapting the semi-global stabilization theory for systems subject to input saturation [10]. The involved algorithm is easier to apply in the case of Hess's LQG pilot mathematical model [4], [11], than in the case of Modified Optimal Control Model (MOCM) considered in [17], [11]. Indeed, in the approach of Hess, the pilot's control task is simply the minimization of a standard quadratic performance index negotiating the pilot's observations and the pilot's commanded control, unlike MOCM in which a supplementary component – the pilot's commanded control-rate – is added.

This paper is organized as follows. Firstly, in Section 2, a proper model of the human pilot is presented, nearly following [4]. Then, in Section 3, the semi-global stabilization concept and a main result are shortly presented and, respectively, suited in order to evaluate a positive influence on the prevention of aircraft PIO II phenomenon in the presence of the realistic position and rate actuator saturations. In Section 4, some numerical simulations are presented. A conclusive Section 5 underlines the interest of the proposed research orientation in the prominence of PIOs.

2 Description of the Hess's LQG Pilot Model Synthesis

The aircraft dynamics is written in the form of well know invariant linear system, see [3]

$$\dot{x} = Ax + B\delta + Ew, y_o = Cx + D\delta + v_y, := y + v_y \quad (1)$$

There are some usual pilot models: optimal control model (OCM) [5], modified optimal control model (MOCM) [4], Hess's LQG model, described in [17], [4], [11]. Herein will be used Hess's LQG model, initially introduced as a structural model [12], composed of two blocks: the central nervous block and the neuromuscular block (Fig. 1). Both are of delay type: first is modeled by the transfer function

$$\frac{u_p}{u_c} = \frac{(s - 4/\tau)^2}{(s + 4/\tau)^2} \quad (2)$$

(τ is the delay, u_p is the pilot's delayed control input, u_c is the pilot's commanded control) and the second is modeled by the lag block

$$\frac{\delta}{u} = \frac{1}{\tau s + 1}, u = u + v \quad (3)$$

Both the blocks are placed in Fig. 1 at the pilot's output and methodologically will be considered as part of the plant dynamics. The two blocks, in state space form, are written so

$$\dot{x}_d = A_d x_d + B_d u_c + E_d v_u, \delta = C_d x_d \quad (4)$$

The dynamics (1), (4) are then concatenated as extended plant dynamics

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{x}_d \end{bmatrix} &= \begin{bmatrix} A & BC_d \\ 0 & A_d \end{bmatrix} \begin{bmatrix} x \\ x_d \end{bmatrix} + \begin{bmatrix} 0 \\ B_d \end{bmatrix} u_c + \begin{bmatrix} E & 0 \\ 0 & E_d \end{bmatrix} \begin{bmatrix} w \\ v_u \end{bmatrix} \\ y &= \begin{bmatrix} C & DC_d \end{bmatrix} \begin{bmatrix} x \\ x_d \end{bmatrix} \end{aligned} \quad (5)$$

or, in matrix form

$$\dot{x}_s = A_s x_s + B_s u_c + E_s w_1, y = C_s x_s, y_0 = C_s x_s + v_y \quad (5')$$

The approach of pilot modeling is based on the hypothesis that the pilot behaves “optimally” [5], more exactly, in the terms of the LQG paradigm, aims to minimize the index

$$J_p = E_\infty \left\{ y^T Q_y y + (u_c)^T r u_c \right\} \quad (6)$$

subject to pilot observations y_o , with cost function weights $Q_y > 0$ and $r > 0$ [13]. The minimizing of the control law is obtained by application of LQG solution techniques to the augmented system. This leads to the full-state feedback relation

$$u_c = -g_p \hat{x}_s = -r^{-1} B_s^T K \hat{x}_s \quad (7)$$

where \hat{x}_s is the estimate of the state x_s and K is the unique positive definite solution of the matrix Riccati equation

$$0 = A_s^T K + K A_s + Q - K B_s r^{-1} B_s^T K \quad (8)$$

with $Q = C_s^T Q_y C_s$. The current estimate of the state is given by a Kalman filter

$$\begin{aligned} \dot{\hat{x}}_s &= A_s \hat{x}_s + B_s u_c + F(y_o - \hat{y}) = \\ & (A_s - F C_s) \hat{x}_s + F C_s x_s + B_s u_c + F v_y, F = S C_s^T V_y^{-1} \end{aligned} \quad (9)$$

The covariance matrix of the estimation error S is the unique positive definite solution of the matrix Riccati equation

$$0 = A_s S + S A_s^T + W_1 - S C_s^T V_y^{-1} C_s S \quad (10)$$

where the covariance matrix W_1 is a diagonal of covariance matrices $W_1 = \text{diag}(W, V_u)$. Consequently, the state space representation of the closed-loop system is given by

$$\begin{aligned} \begin{bmatrix} \dot{\hat{x}}_s \\ \dot{\hat{x}}_s \end{bmatrix} &= \begin{bmatrix} A_s & -B_s g_p \\ FC_s & A_s - B_s g_p - FC_s \end{bmatrix} \begin{bmatrix} x_s \\ \hat{x}_s \end{bmatrix} + \begin{bmatrix} E_s & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} w_1 \\ v_y \end{bmatrix} \\ \begin{bmatrix} y_0 \\ u_c \end{bmatrix} &= \begin{bmatrix} C_s & 0 \\ 0 & -g_p \end{bmatrix} \begin{bmatrix} x_s \\ \hat{x}_s \end{bmatrix} + \begin{bmatrix} v_y \\ 0 \end{bmatrix} \end{aligned} \quad (11)$$

respectively

$$\begin{aligned} \dot{x}_{cl} &= A_{cl}x_{cl} + E_{cl}w, \quad Y = C_{cl}\hat{x}_{cl} + \tilde{v}_y \\ A_{cl} &:= \begin{bmatrix} A_s & -B_s g_p \\ FC_s & A_s - B_s g_p - FC_s \end{bmatrix} \\ E_{cl} &:= \begin{bmatrix} E_s & 0 \\ 0 & F \end{bmatrix}, \quad C_{cl} := \begin{bmatrix} C_s & 0 \\ 0 & -g_p \end{bmatrix} \end{aligned} \quad (11')$$

The pilot's dynamics is represented by

$$\begin{aligned} \begin{bmatrix} \dot{\hat{x}}_s \\ \dot{x}_d \end{bmatrix} &= \begin{bmatrix} A_s - B_s g_p - FC_s & 0 \\ -B_d g_p & A_d \end{bmatrix} \begin{bmatrix} \hat{x}_s \\ x_d \end{bmatrix} + \begin{bmatrix} F \\ 0 \end{bmatrix} y + \\ &\begin{bmatrix} F & 0 \\ 0 & E_d \end{bmatrix} \begin{bmatrix} v_y \\ v_u \end{bmatrix}, \quad \delta = \begin{bmatrix} 0 & C_d \end{bmatrix} \begin{bmatrix} \hat{x}_s \\ x_d \end{bmatrix} \end{aligned} \quad (12)$$

or, in matrix form

$$\dot{x}_p = A_p x_p + B_p y + E_p v_p, \quad \delta = C_p x_p \quad (12')$$

The next step of the synthesis, represented by the explicit determination of the matrices in (11), (12), supposes an optimization procedure for the selection of the tuning parameters: the noises covariance matrix Q_n (see below) and r , in order to obtain the signal noise ratios $V_u/\sigma_{v_u}^2 = \pi \times 0.003$ and $V_{y_i}/\sigma_{y_i}^2 = \pi \times 0.01$, which correspond to normalized control noise and normalized observation noise ratios assumed to be -25 dB and -20 dB, respectively [5]. It is worthy to note that covariance value P of the state vector in (11), given by the Lyapunov equation

$$A_{cl}P + PA_{cl}^T + E_{cl}Q_n E_{cl}^T = 0 \quad Q_n = \text{diag}(\pi \times \sigma_w^2, \pi \times \sigma_u^2, \pi \times \sigma_{y_i}^2) \quad (13)$$

and the covariance of the vector $Y^T = [y_0 \quad u_c]^T$, given by the Lyapunov equation

$$E[YY^T] = C_{cl}PC_{cl}^T + Q_{\tilde{v}_y}, \quad Q_{\tilde{v}_y} = \text{diag}(\pi \times \sigma_{y_i}^2 \quad 0) \quad (13')$$

will be used in the scheme of fitting the Hess's LQG type pilot model (see Section 4).

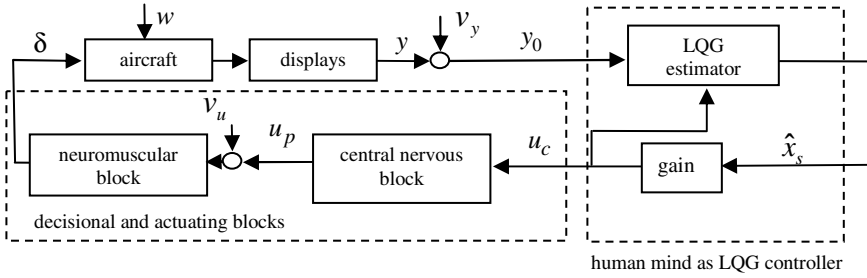


Fig. 1. Conceptual block diagram of the Hess human pilot dynamic model

3 Adapting Semi-global Stabilization Theory for the System with Both Position and Rate Actuator Saturations

A typical block diagram for the study of category PIO I-II is shown in Fig. 2. There, two basic nonlinearities, usual in flight control, are present: the position saturation σ_p , related to control stick displacement limits (corresponding to flight control surface rotation limits) and the rate saturation σ_r , mainly related to flow rate limits of the hydraulic servoactuator). In figure, specifically to the auto-oscillation searching, $r=0$ is the null reference. $K_p(s)$ is the mathematical model of the pilot, $y(s)$ is the vector of displayed variables and δ is the “ideal” signal elaborated by pilot, which will be realistically subjected to rate and position saturations

$$\begin{aligned} \delta(s) &= C_p (sI - A_p)^{-1} B_p y + C_p (sI - A_p)^{-1} E_p v_p \\ &:= K_p(s) (B_p y(s) + E_p v_p) \end{aligned} \tag{14}$$

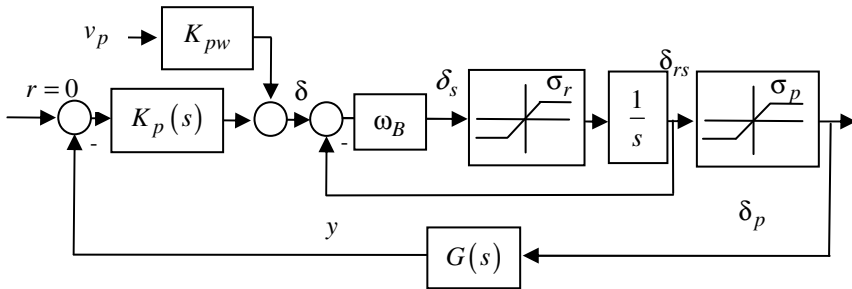


Fig. 2. Block diagram of the realistic system with position and rate saturation

see (12'). Thus, $\delta_p = \sigma_p(\delta_{rs})$ is the effective applied control after being also exposed to the effect of rate limit σ_r . The angular frequency ω_B is in connection with the time constant of the actuator, $\tau_e = 1/\omega_B$ -

$$\dot{\delta}_s + \omega_B \delta_s = \omega_B \delta \quad (15)$$

Actuator dynamics will be implicitly considered only if nonlinear synthesis. $G(s)$ is the model of aircraft as stated in (1) $G(s) = C(sI - A)^{-1}B + D$. In accordance with block diagram in Fig. 2, the system including the two saturation functions is

$$\begin{aligned} \dot{\delta}_{rs} &= \sigma_r(\omega_B(\delta - \delta_{rs})) \\ \dot{x} &= Ax + B\sigma_p(\delta_{rs}) + Ew \\ y_o &= Cx + D\delta + v_y := y + v_y \end{aligned} \quad (16)$$

Starting from [10], is easy to prove that the following family of *controllers* solves the *problem of semi-global stabilization* [14] as applied to the system (16)

$$\begin{aligned} \delta &= C_d(sI - A_d)^{-1}(B_d u_c + E_d v_u) \\ u_c &= -r^{-1} B_s^T K \hat{x}_s \omega_B / \varepsilon^2 - (\omega_B / \varepsilon^2 - 1) \delta_{rs} \\ 0 &= A_s^T K + K A_s + Q(\varepsilon) - K B_s r^{-1} B_s^T K, Q := C_s^T Q_y(\varepsilon) C_s \\ \dot{\hat{x}}_s &= A_s \hat{x}_s + F(y_o - C_s \hat{x}_s), F = S C_s^T V_y^{-1} \\ 0 &= A_s S + S A_s^T + W_1 - S C_s^T V_y^{-1} C_s S \end{aligned} \quad (17)$$

Taking into account the presence of the noises in system, these controllers ensure only local stability. For conformity, usual hypotheses must work: 1) the stabilizability of the pair (A_s, B_s) ; 2) all eigenvalues of A_s are located in the closed left half plane; 3) the detectability of the pair (C_s, A_s) . First two conditions define the so-called null controllability [14]. For the nominal case $\varepsilon^2 = \omega_B$, the saturation corrections are missing, the control (3), (2), (7) is recovered, but will result a pilot model depending on ε , therefore other than standard one.

The solution (17) starts from the parameterization of the state weight matrix Q_y by a single parameter $\varepsilon \in (0, \gamma \omega_B]$, with a suitable $\gamma > 0$

$$A_s^T K + K A_s - K B_s r^{-1} B_s^T K = -Q(\varepsilon) \quad (18)$$

where $Q(\varepsilon) = C_s^T Q_y(\varepsilon) C_s$. The properties of the solution $K(\varepsilon)$ are: 1) $\lim_{\varepsilon \rightarrow 0} K := K(\varepsilon) = 0$; 2) there exists a constant $\alpha > 0$, such that $\|K^{1/2} A_s K^{1/2}\| \leq \alpha$, with α independent of ε . The property 2 can be easily proven, based on usual formulation in the literature $Q(\varepsilon) := \varepsilon I$.

4 Numerical Simulations

Let's now consider the case of human pilot performing the hovering control of a VTOL-type aircraft [5], [6], [7], [8]. Briefly, the pilot's task is to minimize longitudinal position errors while hovering in turbulent air. The approach in the cited references is that of ignoring in estimation/measurement of any information about control. In other words, the matrix D is taken as a null matrix. Our approach will be different. Specifically, the aircraft model and the displayed outputs will be modified so as to consider the state δ_s , $D \neq 0$ and active displays [15], [18].

$$\begin{bmatrix} \dot{u}_g \\ \dot{u} \\ \dot{x}_h \\ \dot{q} \\ \dot{\theta} \\ \dot{\delta}_s \end{bmatrix} = \begin{bmatrix} -\omega_{u_g} & 0 & 0 & 0 & 0 & 0 \\ X_u & X_u & 0 & 0 & -g & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ M_u & M_u & 0 & M_q & 0 & M_\delta \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/\tau_e \end{bmatrix} \begin{bmatrix} u_g \\ u \\ x_h \\ q \\ \theta \\ \delta_s \end{bmatrix} + \quad (19)$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/\tau_e \end{bmatrix} \delta + w := Ax + B\delta + Ew$$

$$A = \begin{bmatrix} -0.314 & 0 & 0 & 0 & 0 & 0 \\ -0.1 & -0.1 & 0 & 0 & -9.81 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0.068 & 0.068 & 0 & -3 & 0 & 16.968 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 \end{bmatrix} \quad (19)$$

$$B = [0 \ 0 \ 0 \ 0 \ 0 \ 20]^T$$

$$\begin{bmatrix} u \\ x_h \\ q \\ \theta \\ u_c \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_g \\ u \\ x_h \\ q \\ \theta \\ \delta_s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_c, \text{ i.e., } y = Cx + Du_c \quad (20)$$

The notations concern: u_g – longitudinal component of the gust velocity [m/s]; u – velocity perturbation \dot{x}_h along the x axis [m/s]; θ – pitch attitude [rad]; $q = \dot{\theta}$ – pitch rate, rad/s; δ – control stick input [m]; δ_s – actuator state variable [m]; M_u – speed stability parameter [rad/m/s]; M_q – pitch rate damping [1/sec]; M_δ – control sensitivity [rad/sec²/m]; X_u – longitudinal drag parameter [1/sec]; g – gravitational constant, 9.81 [m/s²]; τ_e – actuator time constant (0.05 sec); ω_{u_g} – white noise filter pole [rad/s]. The index (6) will be so recalculated

$$\begin{aligned} J_p &= E_\infty \left\{ y^T Q_y y + u_c^2 r \right\} = \\ E_\infty \left\{ x^T C^T Q_y C x + 2u_c D^T Q_y C x + (D^T Q_y D + r) u_c^2 \right\} \\ Q_y &:= \text{diag}(0 \quad 1 \quad 400 \quad 0 \quad \rho_{u_c}) \\ Q_y &:= \text{diag}(0 \quad 1 \quad 400 \quad 0) - \text{nominal case} \end{aligned} \quad (6)$$

Q_y is in accordance with [5], but with a new tuning parameter ρ_{u_c} . Other synthesis parameters are: $r = 1$; $\tau_\eta = 0.1$ s; $\tau = 0.15$ s. Values for the rate saturation σ_r and position saturation of the actuator are $-3 \text{ cm/s} \leq \sigma_r \leq 3 \text{ cm/s}$ and $-16.8 \text{ cm} \leq \sigma_p \leq 16.8 \text{ cm}$, respectively.

Table 1. Stability margins (dB, °) and vector margins (VM) versus the relaxation of the constraints in normalized signals ratios

$V_y / \sigma_y^2, V_u / \sigma_u^2$	dB	°	VM
-20dB, -25dB nominal	5.77	27.6	0.4267
-15dB, -25dB	-6.7	25.6	0.4427
-25dB, -25dB	3.91	22.1	0.3279
-20dB, -20dB	6.35	31.1	0.4670
-20dB, -30dB	5.52	25.7	0.4056

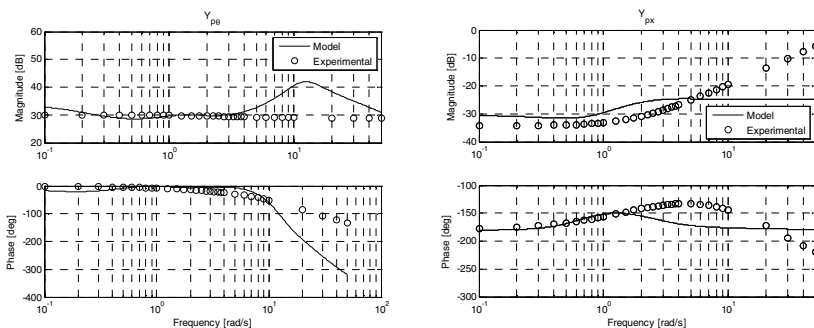


Fig. 3. Comparison between experimental and theoretical results, nominal case (11), $D = 0$

Table 2. Vector margin versus the tuning parameter ρ_{u_c}

	VM
$\rho_{u_c} = 0.1$	0.4357
$\rho_{u_c} = 1$	0.4407
$\rho_{u_c} = 5$	0.4597

The scenario of numerical experiments consisted of the following steps: 1) the determination of Hess’s pilot model and comparison with results of the experimental programs described in [5], [6], [7], [8], in the “nominal” case $D = 0$; 2) simulations of models in accordance with nonlinear blocks in Fig. 2; 3) the analysis of the open loop pilot-aircraft based on Robust Stability Analysis Criterion [16], which evaluates the minimum distance – the so-called vector margin (VM) – from the critical point $(-1,0)$ to the Nyquist plot of the open loop pilot-aircraft transfer function.

The analysis summarized in Tab 1 shows that a value of the normalized control noise of -20 dB, equal with that of the normalized observation noise ratio, gives better robustness than the reference values $(-25$ dB, -20 dB) [5].

Another approach to provide an increased robustness to the system is related to the insertion of the pilot’s commanded control between the displayed variables ((20) and (6’)), see Table 2 and Fig. 4. Table 2 summarizes three cases in which the weighted control signal (flight control position) is displayed to the pilot.

In this paper, the stability of the closed loop pilot-aircraft system in the presence of rate and position limited actuator was approached as following: a) deriving, first, a mathematical model for human pilot using as guide the lines from [4]; b) secondly, adding an antisaturation compensation based on semi-global stabilization theory [10].

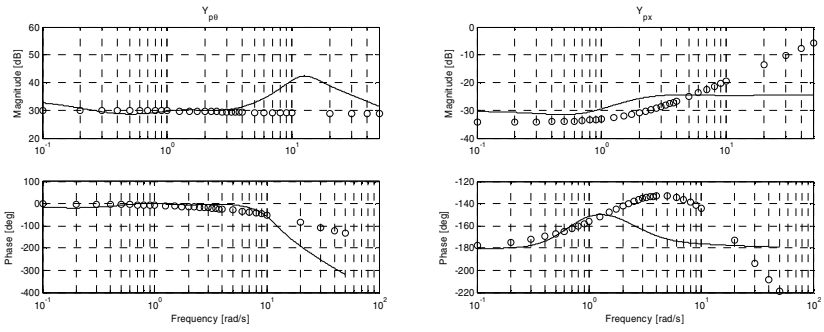


Fig. 4. Comparison between experimental and theoretical results $D \neq 0$ case, $r = 1$, first row in Table 2

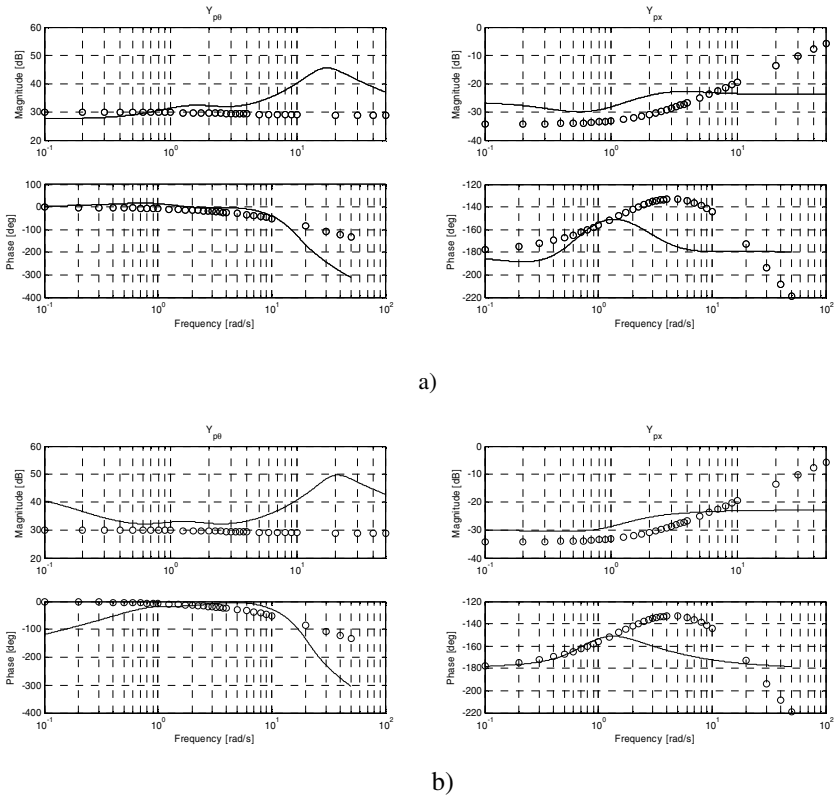


Fig. 5. Comparison between experimental and theoretical results: a) $\epsilon = \sqrt{\omega_B}$ b) $\epsilon = 20$

Simulation results presented in Figures 6 a) and b) represent the response of the closed loop pilot-aircraft system to an initial condition. Figure 6 a) shows the response of the system without a correction component in the control, while in Figure 6 b) the correction is included in the LQG control law. This outcome demonstrates the effectiveness of the algorithm (17) in suppressing the oscillation. Thus, the algorithm (17) can be considered as a method to minimize the risk of the PIO advent.

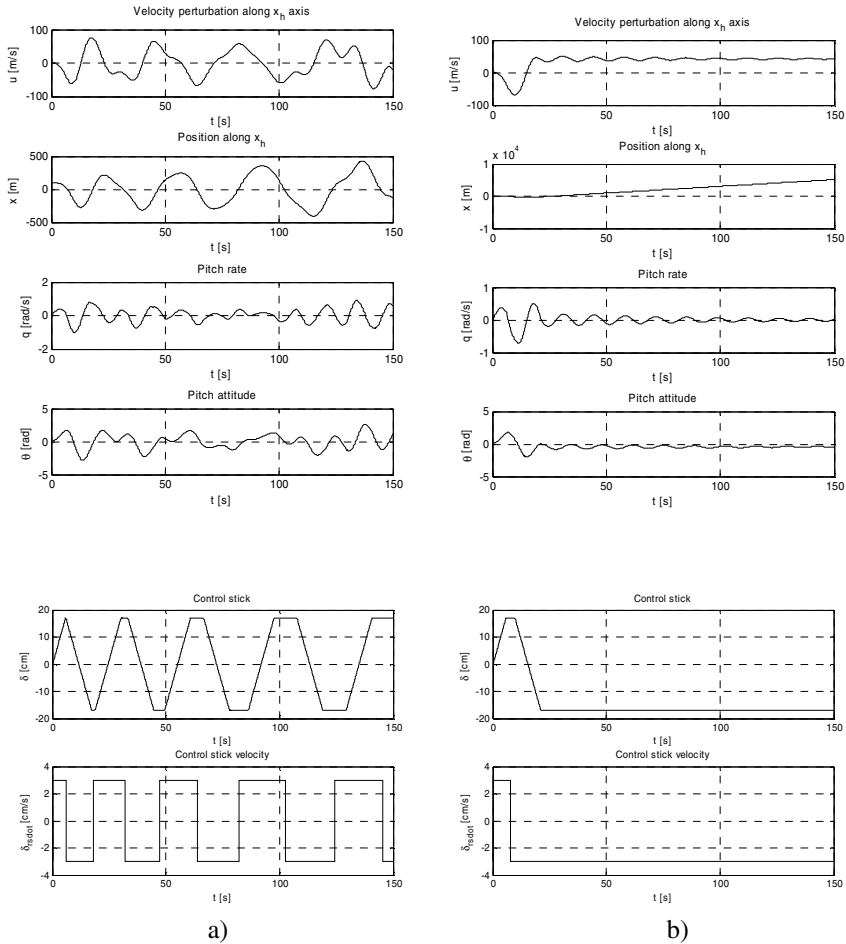


Fig. 6. Time histories of the measured variables, position and rate of the control stick, $x_0 = [0 \text{ m/s} \quad 2.1336 \text{ m/s} \quad 94.488 \text{ m} \quad 0 \text{ rad/s} \quad 0.0663 \text{ rad}]$ a) $\varepsilon = \sqrt{\omega_B}$ a); b) $\varepsilon = 20$

5 Concluding Remark

Generally speaking, many Regulations and criteria concerning the flying qualities specifications for conventional aircraft dynamics are available. In contrast, much less instructions, specifications and criteria are available in the issue of analyzing and predicting the emergence of PIO phenomenon. There is a notable research direction about human pilot modeling, but not on modeling pilot's compensation dynamics. Such a dynamic compensation of linear systems with both position and rate-limited actuators can be provided by the control law analyzed in this paper. In fact, the approach can be connected to some works [15], [18] proposing augmented information on process displayed to the human operator as an aid in the closed-loop control of high performance aircrafts.

Acknowledgements. The authors gratefully acknowledge the financial support of the National Authority for Scientific Research-ANCS, UEFISCSU, through PN-II Research Project code ID 1391/2008.

References

1. Amato, F., Iervolino, R., Scala, S., Verde, L.: Actuator design for aircraft robustness versus category PIO. In: Proc. of the 7th Mediterranean Conf. on Control and Automation, MED 1999, Haifa, Israel, June 28-30 (1999)
2. Klyde, D.M., Mitchell, D.G.: A PIO Case study – Lessons learned through analysis. In: AIAA Atmosphere Flight Mechanics Conference and Exhibit, AIAA 2005-5813, San Francisco, California, August 15-18 (2005)
3. Toader, A., Ursu, I.: PIO I-II tendencies case study. Part I. Mathematical modeling. INCAS Bulletin 2(1), 91–102 (2010a)
4. Davidson, J.B., Schmidt, D.K.: Modified optimal control pilot model for computer-aided design and analysis, NASA-TM-4384 (1992)
5. Kleinman, D.L., Baron, S.: Manned vehicle systems analysis by means of modern control theory. NASA CR-1753, Washington, D.C. (1971) (declassified)
6. Vinje, E.W., Miller, D.P.: Interpretation of pilot opinion by application of multiloop models to a VTOL flight simulator task. NASA SP-144 (March 1967)
7. Vinje, E.W., Miller, D.P.: An analysis of pilot adaptation in a simulated multiloop VTOL hovering task. NASA SP-192 (March 1968)
8. Miller, D.P., Vinje, E.W.: Fixed base flight simulator studies of VTOL aircraft handling qualities in hovering and low-speed flight. United Aircraft Research Laboratories Report F910482-12 (1968)
9. Ursu, I., Ursu, F., Popescu, F.: Backstepping design for controlling electrohydraulic servos. Journal of The Franklin Institute 343, 94–110 (2006)
10. Lin, Z.: Semi-global stabilization of linear systems with position and rate-limited actuators. Systems & Control Letters 30, 1–11 (1997)
11. Toader, A., Ursu, I.: From limits of human pilot mathematical modeling to actuator rate limits. A PIO II tendencies case study. In: Mathematical Methods in Engineering International Symposium, Instituto Politecnico de Coimbra, Portugal, October 21-24. CD published (2010b)

12. Hess, R.A.: Theory for aircraft handling qualities based upon a structural pilot model. *Journal of Guidance, Control, and Dynamics* 12(6), 792–797 (1989)
13. Kwakernaak, H., Sivan, R.: *Linear optimal control systems*. Wiley Interscience, New York (1972)
14. Lin, Z.A., Saberi, A.: Semi-global exponential stabilization of linear systems subject to input saturation via linear feedback. *Systems & Control Letters* 21(3), 225–239 (1993)
15. Davidson, J.B., Schmidt, D.K.: *Extended cooperative control synthesis*, NASA-TM-5r61 (1994)
16. *** *Flight Control Design – Best Practices*, NATO Research and Technology Organization, TR 029 (2000)
17. Hess, R.A.: *A method for generating numerical pilot opinion ratings using the optimal pilot model*, NASA TM X-73,101 (1976)
18. Gary, S., Schmidt, D.K.: *Optimal cooperative control synthesis of active displays*, NASA Report 4058 (1987)

Human Skin Detection Using Texture Information and Vector Processing Techniques by Neural Networks

C.M. Dumitrescu and Ioan Dumitrache

PhD. Student at “POLITEHNICA” University of Bucharest, Professor at “POLITEHNICA” University of Bucharest, Department of Automatic Control and Systems Engineering

Abstract. Starting with algorithms for face detection / recognition, algorithms for hand gesture analysis and ending with expert systems used in the medical field (dermatology), one of the most important stages is represented by the detection of the human skin. In this work a human skin recognition system is developed and tested. To obtain a high accuracy, our system uses information regarding the texture (obtained from GLCM) and color features obtained by using vector processing techniques and classic techniques. Our goal is to develop a system that has a detection rate greater than 98%.

Keywords: skin recognition, skin detection, neural networks, vector processing, texture analysis.

1 Introduction

Human skin detection is an important step for the computer vision and graphic field, but also for other domains as the medical field. An accurate recognition scheme can greatly enhance the performance of algorithms such as facial detection / recognition and facial features localization / tracking. In the medical field, such as dermatology, human skin recognition is utilized in methods for computer assisted diagnosis of skin disorders.

Many skin segmentation methods depend only on skin color information. Even if these methods have a high processing speed, they all present a big drawback: low accuracy. To use only the skin color in segmenting an image into “*skin*” / “*non-skin*” regions represents a vulnerable technique. The human skin color depends on many variables such as subject parameters (age / race / sex), body part, image parameters (lighting, camera position). All this variables lead to a high error rate. Although the different lighting conditions can be partially avoided by using the $YCbCr$ color space, there are still many unanimated objects in the real world that have a chrominance in the range of the human skin. Thereby, those objects

will be wrongly classified as human skin. To avoid cases like the ones presented earlier, there were developed methods that combine skin color features with texture features, this way a higher accuracy was obtained.

All of the existing methods don't take into account the correlation that exists between the color components of an RGB color image. To extract the color features they treat each individual channel of the color image as a monochrome image. This way, the correlation that exists between the color components of natural images represented in a correlated color space, is disrupted.

The purpose of this work is to use the vector processing techniques for extracting some of the color features of an image. Thus, a color image is treated as a vector field. Assuming a color image in the RGB space $p: \mathbb{N}^2 \rightarrow \mathbb{N}^3$, each pixel $p_{ij} = [p_{ij}^R \ p_{ij}^G \ p_{ij}^B]^T$ represents a three-component vector in a color space.

1.1 State of the Art

Most of the existing skin segmentation techniques classify each pixel of a color image into "skin" / "non-skin" categories only on the basis of pixel color information.

K. Bhojar and O. Kakde propose, in [1], the usage of a "feedforward" neural network. The designed neural network had 3 neurons in the input layer, 5 neurons in the hidden layer and 2 neurons in the output layer. The inputs of the neural network are the three color components of each pixel (RGB). The first neuron of the output layer represents the "skin" class and the second neuron the "non-skin" class. The best performance obtained by this classifier was a 95% detection rate and a 3% false positive rate.

Table 1. Performance of different skin detectors

Method	Detection False	
	Rate	Positive Rate
Bayes SPM (RGB)	80%	8,5%
[Jones and Rehg 1999]	90%	14,2%
Bayes SPM (RGB) [Brand and Mason 2000]	93,4%	19,8%
Maximum Entropy Model in RGB [Jedynak et al. 2002]	80%	8%
Gaussian Mixture models in RGB	80%	9,5%
[Jones and Rehg 1999]	90%	15,5%
Self Organizing Map in TS [Brown et al. 2001]	78%	32%
Single Gaussian in CbCr [Lee and Yoo 2002]	90%	33,3%
Elliptical boundary model in CIE-xy	90%	20,9%
[Lee and Yoo 2002]		
Thresholding of I axis in YIQ [Brand and Mason 2000]	94.7%	30.2%
Gaussian Mixture in IQ [Lee and Yoo 2002]	90%	30%

A survey on pixel color based skin segmentation techniques is presented in [2]. The presented methods for image segmentation into “skin” / “non-skin” regions are: Bayes classifier, Kohinen neural networks (Self Organizing Map), single Gaussian skin distribution model, multiple Gaussian clusters. The detection rate of these methods is between 78% and 94.7% and the false positive rate ranging between 8.5% and 30.2%, as shown in (Table 1.).

N.K. Al Abbadi, N.S. Dahir and Z.A. Alkareem propose the usage of both skin color features and texture features for the image segmentation into “skin” / “non-skin” regions. The system designed by them has a detection rate of 96%.

2 Feature Extraction

The features vector is extracted by scanning the picture (pixel by pixel) with a square processing window (see Fig. 1). The scanning process is started from the top right corner. For each position of the processing window, we extract a features vector which contains information regarding the color and texture.

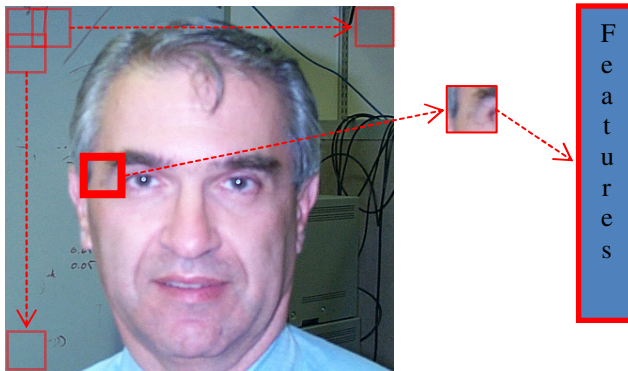


Fig. 1. Image scanning process (features extraction using a sliding window)

2.1 Color Features Extracted Using Vector Processing Techniques

Assuming that the texture is a random process ($f(x) \in \mathbb{R}^3$), for a region R , containing N pixels, several k order statistical moments can be defined:

$$m_k = \frac{1}{N} \sum_{x \in R} f(x)^k \tag{2.1}$$

$$\overline{m_k} = \frac{1}{N} \sum_{x \in R} (f(x) - m_1)^k \quad (2.2)$$

The main statistical moments used to characterize the color distribution are: the first-order moment (m mean color), the second-order moment (σ standard deviation), and the third-order moment (θ skewness of color). In this work we will be using only two statistical moments to characterize the color distribution: m mean color and σ standard deviation.

$$m = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N p_{ij} \quad (2.3)$$

$$\sigma = \left[\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (p_{ij} - m)^2 \right]^{1/2} \quad (2.4)$$

Where M and N are the image dimensions, p_{ij} is the color pixel in the i^{th} row and j^{th} column of the image.

A color image, described in the RGB color space is composed of three color components (R – red, G – green, B – blue). Each color component can be regarded as a monochrome image. Traditional methods of extracting the k order statistical moments often involve the application of the formulas previously defined on each color channel separately. Thus, resulting a separate value for each color component (R – red, G – green, B – blue):

$$m_c = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N p_{ij}^c \quad (2.5)$$

$$\sigma_c = \left[\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (p_{ij}^c - m_c)^2 \right]^{1/2} \quad (2.6)$$

Where M and N are the image dimensions, p_{ij}^c is the value of the c^{th} color component of the color pixel in the i^{th} row and j^{th} column of the image.

However, such an approach, treating each color channel separately, disrupts the correlation that exists between the color components of natural images represented in a correlated color space, such as RGB. Each processing step is usually accompanied by certain inaccuracy, inaccuracy that leads to color artifacts. To diminish the probability of color artifacts, the two statistical moments used in this work to characterize the color distribution (m mean color and σ standard deviation); will be determined by using vector processing techniques. Vector processing techniques treat the color image as a vector field. Assuming a color image in the RGB space $p: N^2 \rightarrow N^3$, each pixel $p_{ij} = [p_{ij}^R p_{ij}^G p_{ij}^B]^T$ represents a three-component vector in a color space. The color image p is a two-dimensional matrix of three components vectors.

Because each pixel is treated as three dimensional vectors, a distance between two vectors must be defined. The most commonly used distance between two color vectors, $p_1 = [p_1^R p_1^G p_1^B]^T$ and $p_2 = [p_2^R p_2^G p_2^B]^T$, is the generalized weighted Minkowski metric:

$$d(p_1, p_2) = |p_1 - p_2| = c \left(\sum_{k=1}^3 \xi_k |p_{1k} - p_{2k}|^L \right)^{\frac{1}{L}} \quad (2.7)$$

The non-negative scaling parameter c is a measure of the overall discrimination power. The exponent L defines the nature of the distance metric. The parameter ξ_k measures the proportion of attention allocated to the dimensional component k and, therefore $\sum_{k=1}^3 \xi_k = 1$. The most popular metrics are obtained when $L = 1$ (city-block distance), $L = 2$ (Euclidian distance) and $L = \infty$ (chess-board distance).

To determine the mean color of an image region, we will not be using the mathematical equation (2.5), but a three-dimensional median filter. Each output pixel of the filter represents the median value in the M by N by 3 neighborhood centered on the corresponding pixel.

To determine the standard deviation, we are using the L_2 metric (Euclidian distance), as the distance between two vectors:

$$p_{ij} - m = \left[\sum_{c=1}^3 (p_{ij}^c - m_c)^2 \right]^{\frac{1}{2}} \quad (2.8)$$

Substituting (2.8) in (2.4), yields:

$$\sigma = \left[\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \left(\left[\sum_{c=1}^3 (p_{ij}^c - m_c)^2 \right]^{\frac{1}{2}} \right)^2 \right]^{\frac{1}{2}} \quad (2.9)$$

$$\sigma = \left[\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \sum_{c=1}^3 (p_{ij}^c - m_c)^2 \right]^{\frac{1}{2}} \quad (2.10)$$

2.2 Color Features Extracted Using Classic Processing Techniques

To characterize the color distribution in a non-vector way, we use the Shannon entropy for each color channel of the RGB triplet. Each color component will be treated separately as a monochrome image. For each of the three color planes (R G B), we are calculating the value of the Shannon entropy:

$$H_c = - \sum_{i=0}^{255} p_i \log_2(p_i) \quad (2.11)$$

Where p_i represents the probability of a pixel whose color component c has the value $i \in [0 \dots 255]$.

2.3 Texture Features

In the 1970's Haralick et al. propose the usage of the gray level co-occurrence matrix (GLCM) as a method of texture characterization.

For a defined region R of the studied texture and for a given spatial displacement vector t , the components of the gray-level co-occurrence matrix are defined for all possible pairs of gray levels (a, b) as:

$$M_t(a, b) = \text{Card} \left\{ \begin{array}{l} (x, x+t) \in R \times R \\ f(x) = a \text{ and } f(x+t) = b \end{array} \right\} \quad (2.12)$$

Where $M_t(a, b)$ represents the number of pixel pairs in the defined region R , separated by the spatial displacement vector t , who's gray level values are equal to a and respectively b .

Because co-occurrence matrices are typically large and sparse, various metrics of the matrix are often taken to get a more useful set of features. In this work the following texture features are computed using the gray level co-occurrence matrix:

Homogeneity:

$$O = \frac{1}{N_{nz}} \sum_a \sum_b \frac{1}{1 + |a - b|} M_t(a, b) \quad (2.13)$$

Contrast:

$$C = \frac{1}{N_{nz}} \sum_{a, b} |a - b|^2 M_t(a, b) \quad (2.14)$$

Correlation:

$$B = \frac{1}{N_{nz}} \sum_a \sum_b \frac{(a - m_a)(b - m_b) M_t(a, b)}{\sigma_a \sigma_b} \quad (2.15)$$

Where N_{nz} represents the number of non-zero elements in the co-occurrence matrix, m_a and m_b are the mean values along the lines respectively the columns, σ_a and σ_b represents the corresponding dispersions.

In this work the spatial displacement vector t will have two values: $t = \{(0, 1), (-1, 0)\}$. As a result we will extract six parameters (three for each value of the displacement vector t).

3 The Proposed Algorithm

The main stages of our proposed skin recognition algorithm are:

- skin samples database creation;
- neural network training;
- image segmentation into “*skin*” / “*non-skin*” regions using the neural network.

Segmentation of an image into two classes “*skin*” / “*non-skin*” requires the coverage of several steps. The first step is to extract the features vector for each pixel. Using the previously determined characteristics vectors for each pixel we determine, by using the neural network, the degree of belonging to the two classes “*skin*” and “*non-skin*”. Pixels with the degree of belonging to “*skin*” category over a preset threshold (in this work set at 0.5) are considered to be human skin, all others are considered to be non-skin pixels.

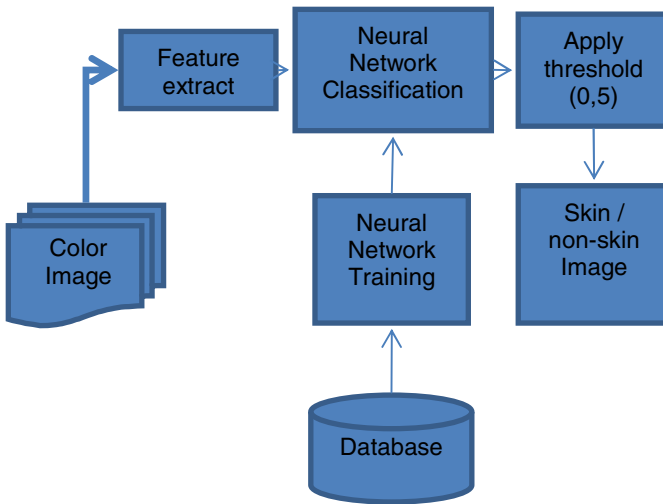


Fig. 2. Proposed algorithm

3.1 Skin Samples Database

The database is composed by 942 human skin samples and by 1348 non-skin samples, with the minimum dimensions of 51 by 51 pixels. Resulting in more than two million pixels of “*skin*” and over three million pixels of “*non-skin*”. The samples were extracted manually from more than 400 color images. Samples of the database are shown in Fig. 3 and Fig. 4.



Fig. 3. Human skin samples

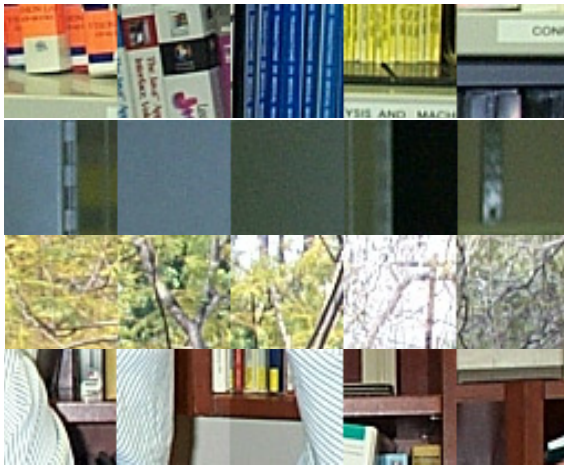


Fig. 4. Non-human skin samples

3.2 *The Structure of the Neural Network*

To classify the pixels into the “*skin*” / “*non-skin*” classes, we use a “*feed-forward*” multilayer perceptron, neural network. The proposed architecture of the neural network used in this work has three layers, as follows:

- Input layer: consisting of 16 neurons;
- Hidden layer: consisting of 60 neurons;
- Output layer: consisting of 2 neurons.

The activation function used is the tan sigmoid function for both the hidden layer and the output layer. This function satisfies the differentiability and monotonicity requirements imposed by the “*back-propagation*” training algorithm.

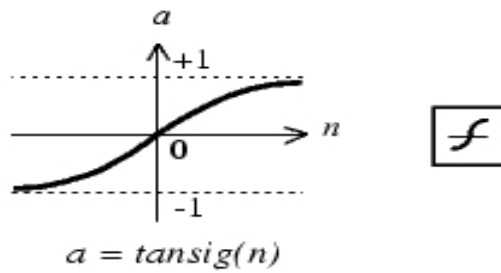


Fig. 5. Neuron activation function

The input of the neural network is the feature vector containing 16 components for each image pixel. The feature vector is composed of: the pixel color in the YC_bC_r color space (3 values), the texture features defined in (2.13), (2.14) and (2.15) (6 values), Shannon entropy (2.11) of each color component in the RGB color space (3 values), standard deviation (2.10) (1 value) and mean color (3 values).

The neural network classifier has two outputs, one for each of the two classes (“skin” / “non-skin”). The output of the neural network represents the degree of membership of the input vector to one of the two classes “skin” / “non-skin”. The value is between 0 (0%) and 1 (100%).

The network is trained using the “back-propagation” algorithm, *Levenberg - Marquardt* method. This algorithm minimizes the error between the desired output and the actual output. As performance criteria we use the mean square error (MSE).

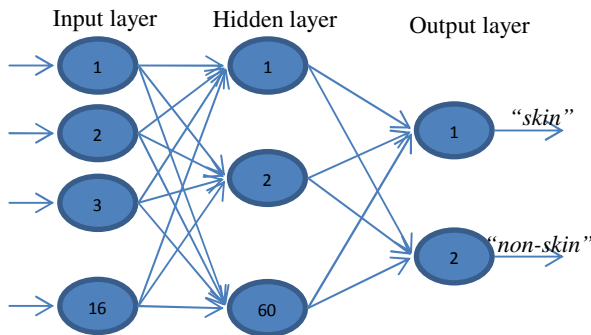


Fig. 6. Neural network structure

4 Results

The neural network was trained using “skin” and “non-skin” samples from the database. In total, 477.516 of “skin” pixels and more than 500.000 of “non-skin” pixels were used to train the neural network. The first output of the neural network is assumed to be 1 for “skin” input and the second output to be 1 for “non-skin” input.

The training data was divided into three subsets, as follows: 70% (the training set) were used for computing the gradient and updating the network weights and biases, 15% validation set and 15% test set. The error on the validation set is monitored during the training process. The validation error normally decreases during the training process, but when the neural network begins to over fit the data it begins to rise.

Following the neural network training, we have obtained the confusion matrixes presented Fig. 7, Fig. 8, Fig. 9 and Fig. 10. A confusion matrix contains information about known class labels and predicted class labels. The (i, j) element in the confusion matrix is the number of samples whose known class label is class i and whose predicted class is j . The diagonal elements represent correctly classified inputs.

Output Class	1	330296 48.3%	4628 0.7%	98.6% 1.4%
	2	4071 0.6%	345311 50.5%	98.8% 1.2%
		98.8% 1.2%	98.7% 1.3%	98.7% 1.3%
		1	2	
		Target Class		

Fig. 7. Training confusion matrix

Output Class	1	70880 48.3%	965 0.7%	98.7% 1.3%
	2	920 0.6%	73843 50.4%	98.8% 1.2%
		98.7% 1.3%	98.7% 1.3%	98.7% 1.3%
		1	2	
		Target Class		

Fig. 8. Validation confusion matrix

Output Class	1	70480 48.1%	1022 0.7%	98.6% 1.4%
	2	869 0.6%	74246 50.6%	98.8% 1.2%
		98.8% 1.2%	98.6% 1.4%	98.7% 1.3%
		1	2	
		Target Class		

Fig. 9. Test confusion matrix

Output Class	1	471656 48.2%	6615 0.7%	98.6% 1.4%
	2	5860 0.6%	493400 50.5%	98.8% 1.2%
		98.8% 1.2%	98.7% 1.3%	98.7% 1.3%
		1	2	
		Target Class		

Fig. 10. All confusion matrix

Analyzing the all confusion matrix (Fig. 10), we can observe that the following performances were achieved:

- “skin” detection rate (output class number 1) 98.8%;
- false negative rate 1.2%, this means that 1.2% of the “skin” pixels were misclassified as “non-skin”;
- “non-skin” detection rate (output class number 2) 98.7%;
- false positive rate 1.3%
- overall performance:
 - detection rate 98,7%;
 - error rate 1,3%.

Fig. 11 shows the neural networks error histogram. We can note that over 95% of the pixels have a membership error of less than or equal to 5%.

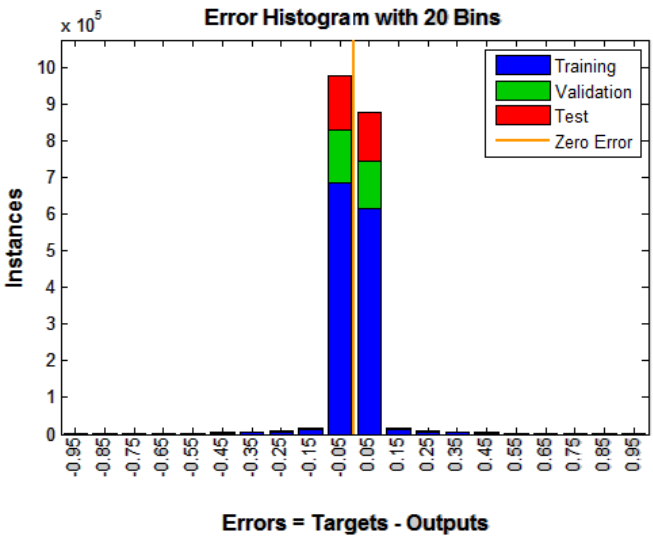


Fig. 11. Training error histogram

Fig. 12 shows the true positive rate (sensitivity) versus false positive rate (1 - specificity), also known as “Receiver Operating Characteristic” or ROC curve. We can note that curve is near the upper left corner or coordinate (0,1) of the ROC space, also known as the perfect classification point. This point represents 100% sensitivity (no false negatives) and 100% specificity (no false positives).

To test our system after training, we used 100 human “skin” samples and 100 “non-skin” samples. The images were randomly chosen from the database and were not used in the training phase. We have obtained the following results.

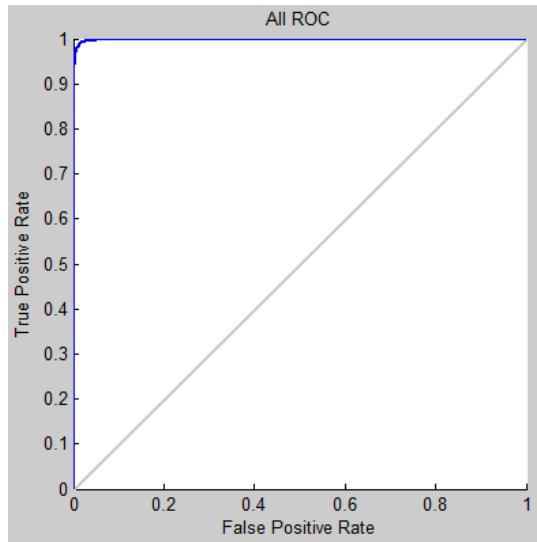


Fig. 12. Receiver Operating Characteristic

We assume that a sample is correctly classified if the average classification error of all the pixels that belong to the sample is less than or equal to 0.5 (50%). A pixel is correctly classified if the error to its membership class is less than or equal to 50%.

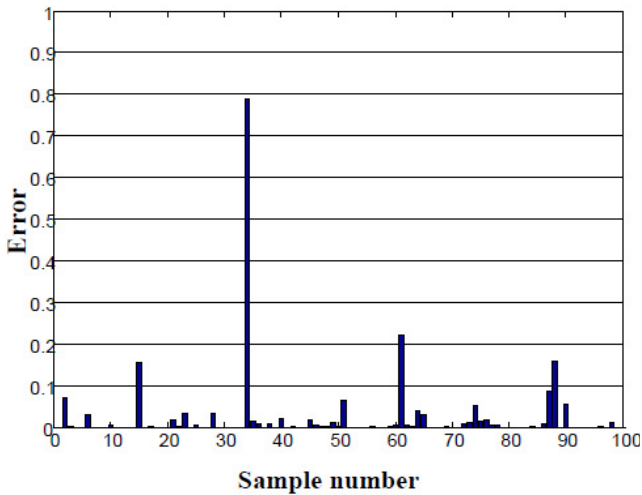


Fig. 13. Error classification: "skin" samples

Fig. 13 shows the classification error of 100 "skin" samples randomly chosen. We can note that from 100 samples, 99 are correctly classified.

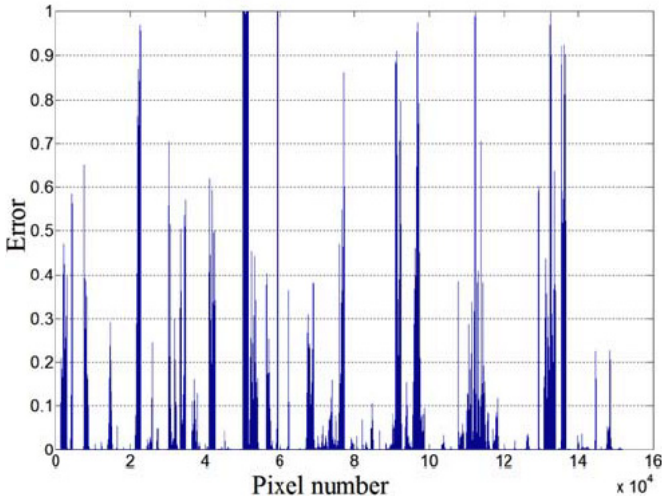


Fig. 14. Error classification at pixel level: “*skin*” samples

Fig. 14 represents the classification error at pixel level for pixels that belong to the “*skin*” class. There are a total of 152.061 “*skin*” pixels, of who 1.975 have a classification error higher than 0.5 (50%). This leads to a classification error (false negative rate), at pixel level, of 1.2%, which is similar to the one obtained in the training stage.

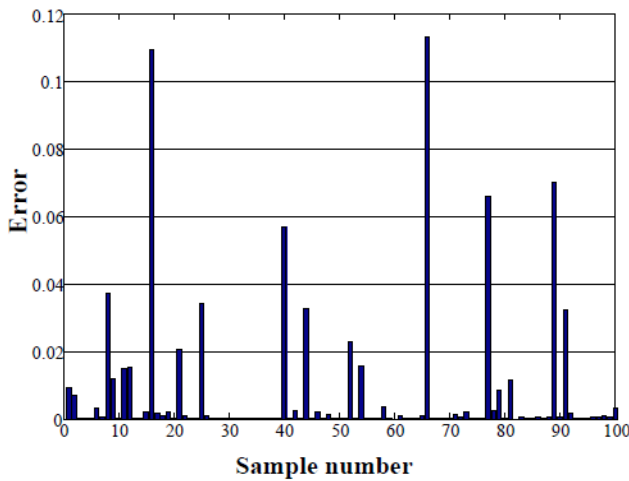


Fig. 15. Error classification: “*non-skin*” samples

Fig. 15 shows the classification error of 100 randomly chosen “*non-skin*” samples. We can note that all of the samples were correctly classified. Their main error is less than 0.5 (50%).

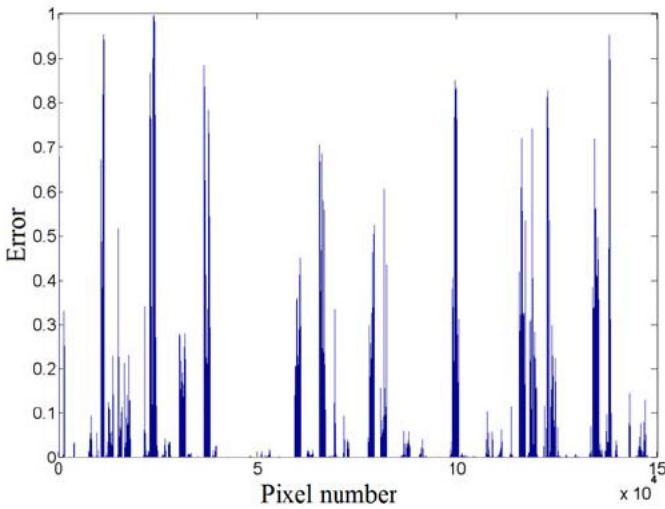


Fig. 16. Error classification at pixel level: “*non-skin*” samples

The classification error at pixel level for “*non-skin*” samples is shown in Fig. 16. Thus, from 147.573 “*non-skin*” samples, only 403 pixels have a membership error greater than 50%. Resulting in a classification error (false positive rate), at pixel level, of 0.273%.

Fig. 17 shows 3 “*skin*” samples that are misclassified, as “*non-skin*”, by the neural network.

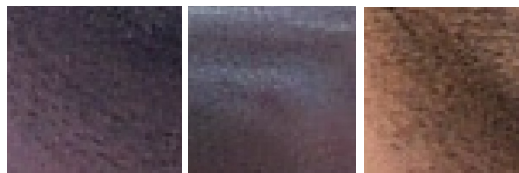


Fig. 17. “*Skin*” samples classified as “*non-skin*”

A real example on a full image is shown in the figures below:

- Fig. 18 represents the input color image;
- Fig. 19 represents the output of the neural network;
- Fig. 20 represents the image segmented into “*skin*” / “*non-skin*” with the threshold set at 0.5 (50%).



Fig. 18. Input image



Fig. 19. Neural network output image



Fig. 20. Segmented image

In Fig. 21, Fig. 22 and Fig. 23 a worst case scenario is presented:

- Fig. 21 contains a piece of furniture made of ecological skin;
- Fig. 22 represents the output of the neural network;
- Fig. 23 represents the image segmented into “*skin*” / “*non-skin*” with the threshold set at 0.5 (50%).



Fig. 21. Input image

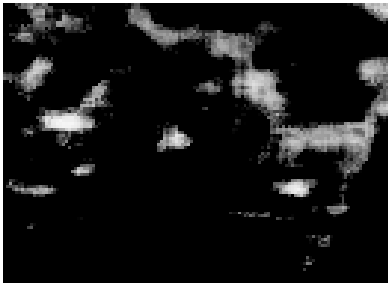


Fig. 22. Neural network output

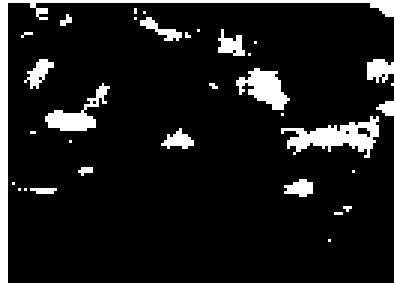


Fig. 23. Segmented image

5 Conclusions

Analyzing the previously presented results (Fig. 13, Fig. 14, Fig. 15 and Fig. 16) we can concur that the system presents a high rate of classification into “*skin*” / “*non-skin*” pixels classes.

The “*skin*” detection rate obtained both during neural network training stage and during the random samples tests was above 98%. Thus, of 100 samples of human skin, only one was misclassified, all the other samples presented a membership class error below 25%. For the “*non-skin*” samples the membership class error was even lower, under 12%.

We can observe that the “*skin*” samples that were falsely classified as “*non-skin*”, shown in Fig. 17, are either taken under poor lighting conditions or they contain facial hair (beard, mustache).

In the worst case scenario, when the system was tested on a piece of ecological skin (Fig. 21, Fig. 22 and Fig. 23), the false positive rate was 7.16%, 882 pixels out of 12.314 were misclassified as “*skin*”.

The presented algorithm and system was designed for high accuracy, not for speed, so in the current form it cannot be used in real time applications.

The feature vector extraction time for Fig. 18, whose size is 700 by 400 pixels, was 230 second; and the time needed for the neural network to process the feature vector was 0.9 seconds.

A performance enhancement regarding the processing speed, without reducing the accuracy of the system, can be achieved by using parallel computing on *CUDA* architectures.

References

- [1] Bhojar, K., Kakde, O.: Skin color detection model using neural networks and its performance evaluation. *Journal of Computer Science* 6 (2010)
- [2] Vezhnevets, V., Sazonov, V., Andreeva, A.: A survey on pixel-based skin color detection techniques. In: *GRAPHICON 2003* (2003)
- [3] Al Abbad, N., Dahir, N., Alkareem, Z.: Skin texture recognition using neural networks. In: *International Arab. Conference on Information Technology* (2008)
- [4] Phung, S., Bouzerdoum, A., Chai, D.: Skin segmentation using color pixel classification: analysis and comparison. *IEEE Trans. Pattern Analysis and Machine Intelligence* 27(1) (January 2005)
- [5] Hassanpour, R., Shahbahrami, A., Wong, S.: Adaptive gaussian mixture model for skin color segmentation. In: *World Academy of Science, Engineering and Technology* (2008)
- [6] Haralick, R.M., Shanmugam, J., Dinstein, I.: Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics SMC-3*, 610–621 (1973)
- [7] The Caltech Frontal Face Dataset, collected by Markus Weber. California Institute of Technology, USA [Interactiv], <http://www.vision.caltech.edu/html-files/archive.html>
- [8] Dumitrescu, C.M., Dumitrache, I.: Human skin detection using texture information and vector processing techniques. In: *The 18th International Conference on Control Systems and Computer Science* (2011)
- [9] Dumitrache, I.: Neural networks

MDL Based Structure Selection of Union of Ellipse Models for Scaled and Smoothed Histological Images

Jenni Hukkanen¹, Edmond Sabo², and Ioan Tabus¹

¹ Department of Signal Processing, Tampere University of Technology, PO Box 553 Tampere, Finland

² Department of Pathology, Rappaport Faculty of Medicine, Technion, Haifa, Israel

Abstract. In this chapter, we investigate refinements to the structure selection method used in our recently developed minimum description length (MDL) method for interpreting clumps of nuclei in histological images. We start from the SNEF method, which fits elliptical shapes to the clump image based on the extracted contours and on the image gradient information. Introducing some variability in the parameters of the algorithm, we obtain a number of competing interpretations and we select the least redundant interpretation based on the MDL principle, where the description codelengths are evaluated by a simple implementable coding scheme. We investigate in this paper two ways for allowing additional variability in the basic SNEF method: first by utilizing a pre-processing stage of smoothing the original image using various degrees of smoothing and second by using re-scaling of the original image at various downsizing scales. Both transformations have the potential to hide artifacts and features of the original image that prevented the proper interpretation of the nuclei shapes, and we show experimentally that the set of candidate segmentations obtained will contain variants with better MDL values than the MDL of the initial SNEF segmentations. We compare the results of the automatic interpretation algorithm against the ground truth defined by annotations of human subjects.

1 Introduction

The analysis of hematoxylin and eosin (H&E) stained histological images is one of the most used basis for the diagnosis and staging of many types of cancers and it is of vital importance for a correct diagnosis and treatment. It is difficult to predict when entirely automatic diagnosis will be possible, but important steps were taken towards assisted diagnosis, where the pathologist is offered various options for enhancing, pre-processing, segmenting, and extracting features of interests from the H&E images [4].

The final goal is to be able to create an automatic tool for tissue diagnosis (i.e. to automatically distinguish cancer versus benign tissues), or maybe a semiautomatic diagnosis where an expert guided system will suggest the diagnosis to the pathologist who will still keep the final word in deciding the diagnosis. This may reduce heavy workload that pathologist are exposed to, will increase the diagnostic accuracy and will reduce the interobserver or intraobserver variability when performing the diagnosis or the prediction of the patient prognosis.

One of the most laborious and difficult tasks when analyzing H&E images is the segmentation of cell nuclei, which are normally seen as elongated shapes having bluish colour, while the surrounding parts of the tissue have various shades of pink. Separation of nuclei from H&E images based only on colour information is not reliable due to non-uniformity and non-homogeneity of the practical staining process, noise in the images, imperfect illumination conditions. For this reason, after thresholding based on colour and intensity information and after applying basic morphological operations to remove small objects, one gets contours of groups of nuclei, which are in general noisy and non-smooth and may enclose more than one nucleus. One of the major problems is that in the plane of the image we get projections of the objects from within a thin slice of tissue, which is however three-dimensional. Quite often, if the nuclei located in the three-dimensional slide are dense enough, the image obtained by taking the projection to a plane will contain overlapping nuclei. In addition, malignant cells usually have nuclei of large size, and for adjacent cells, we will see in the H&E images nuclei that are touching each other. Consequently, apart from well-separated and delineated nuclei, we see in H&E images many clumps of nuclei.

Among the features used by pathologists to describe nuclei in H&E images are the orientation of nuclei and the eccentricity of nuclei, under the implicit assumption that nuclei shape should be first approximated by an ellipse, whose major axis will establish the orientation, and the ratio of major axis over minor axis will represent the eccentricity. Therefore, interpretation of a clump of nuclei as a superposition of a few elliptic shapes will help providing the features necessary to pathologists for diagnosis. In case when the contours of the clumps are not noisy, the interpretation of the clump based on its contour is possible and was presented in [1, 8].

We presented in a recent paper [5] ways to interpret clumps of nuclei as superposition of ellipses based not only on the contour information, but also on additional information from the luminance image. We introduced in [6] a principled way to decide about the best interpretation among a number of possible interpretations, provided by human subjects or by an algorithm, based on the minimum description length (MDL) principle. The MDL principle is widely used for deciding the best structure of a model, when many such competing structures are available. For image segmentation this principle was used to successfully segment natural images in [7,9,10], while more recent papers discuss the usage of this principle in connection to sparse modeling, in the context of audio coding in [3], and more generally, in its asymptotic equivalent form, Bayesian information criterion (BIC), in [11].

We present in this paper improvements to our previous tool for interpreting clumps of overlapping nuclei in H&E images and further analyze the effects of spatial image transforms on the precision of the segmentation.

2 Evaluating the Performance of a Segmentation by Using Minimum Description Length

The segmentation algorithm produces a number of n_E ellipses, each ellipse \mathcal{E}_i being described by its five parameters: center coordinates (x_{0_i}, y_{0_i}) , major semiaxis a_i , minor semiaxis b_i , and angle θ_i between the major axis and the horizontal coordinate axis, resulting in a parameter vector $\boldsymbol{\beta}_i = [x_{0_i}, y_{0_i}, a_i, b_i, \theta_i]$. For one ellipse, we denote \mathcal{E}_i the set of points of the image grid contained inside the contour of the ellipse. The union of all points of the ellipses $\Omega_F = \bigcup_{i=1}^{n_E} \mathcal{E}_i$ defines the clump interior points, or foreground, and we simply define the background as the complementary set $\Omega_B = \Omega \setminus \Omega_F$, where Ω is the set of pixels of the whole image.

We propose an implementable (non-asymptotic) two-part code MDL, where the $n \times m$ luminance image I is encoded by transmitting first the information about ellipses, and then the luminance of all the pixels is transmitted, using different encoding parameters for the foreground and the background.

The process of encoding in [6] starts with transmitting the number of ellipses, n_E , followed by the n_E sets of quantized parameters $Q(\boldsymbol{\beta}_1), \dots, Q(\boldsymbol{\beta}_{n_E})$. The quantization of the parameter vector $\boldsymbol{\beta}_i$ is done elementwise, with uniform quantization using the same number of bits, q , for each element of the parameter vector $\boldsymbol{\beta}_i$ (in our experiments $q = 7$). The upper and lower bounds for each parameter are known to both encoder and decoder. This stage requires $n_E(1 + 5q)$ bits. Then the average luminance value of the foreground \hat{z}_F (the average value of I over Ω_F), is transmitted using $q_1 = 8$ bits, followed by the average luminance value of the background, \hat{z}_B , transmitted with q_1 bits.

Finally, for all pixels (x, y) of the image we transmit the residuals $I(x, y) - \hat{z}_F$, if $(x, y) \in \Omega_F$, and $I(x, y) - \hat{z}_B$ if $(x, y) \in \Omega_B$. The residuals are integer values with sign, for which first a mapping to non-negative integers is performed, after which the non-negative integers are encoded using Golomb-Rice codes [13], with different parameter l_F and l_B for the foreground and background, respectively. The optimal parameters are transmitted as side information before starting the transmission of the residuals.

We note that the largest part of the description length is formed by the residuals, and the prior transmission of ellipses has the potential of improving significantly the residual codelength since the ellipses are splitting the residuals into two distinct classes, foreground and background, having essentially different properties, each class being encoded using different Golomb-Rice code parameters. Hence, segmentation could be evaluated using its associated MDL

value, obtained by adding the costs of transmitting the parameters of the ellipses, the residuals, and the coding parameters. If another segmentation has a better MDL value, it means that the second segmentation makes the split into foreground and background in a better way, which translates into a more efficient representation of the image.

Similar ideas for scoring segmentations based on MDL were used in [7,9,10] but their approach is different, first by being based on chain codes for contours of the regions (which will allow only representing outer contours and are not well suited to represent overlapping objects) and second by using asymptotic expressions for the parameter codelength.

3 MDL Variability under Spatial Transformation

3.1 Experimental Set-Up

The experiments are realized using H&E images from a database of images collected from patients suspected of esophageal cancer of various grades (some of the patients were finally diagnosed as healthy based on the pathological analysis of the images). The type of tissue is Barrett's epithelium from the esophagus. The same database was used in [6]. We noticed that apart of the isolated nuclei, which can be segmented very reliably, there are many clumps of nuclei, which are difficult to interpret.

3.2 Down-Scaling

One desirable feature of the segmentation algorithm is that it should work consistently and reasonably well over a wide range of sizes of images and we investigate here the sensitivity of the solution with respect to the size of the image, when the image is down-scaled by various scaling factors.

Due to the changes introduced by the scaling transformation, we also expect some variability in the results of the segmentation. At some scales some features can be seen better than at other scales, and the segmentation algorithm will take different decisions and produce different segmentations when analyzing the different scaled versions of the same image. These different segmentations can be utilized either for selecting an winner, selected according to the best achieved MDL value, or they can be combined so that the consistent parts of the segmentations will be reinforced and the regions that are different are either combined into a new region, or they are evaluated as insignificant and discarded as spurious contours.

A number of various strategies can be devised along these lines, but the goal of this paper is mainly to evaluate the type of variability present in the segmentation of the same image at various scales and to illustrate the results for some typical images.

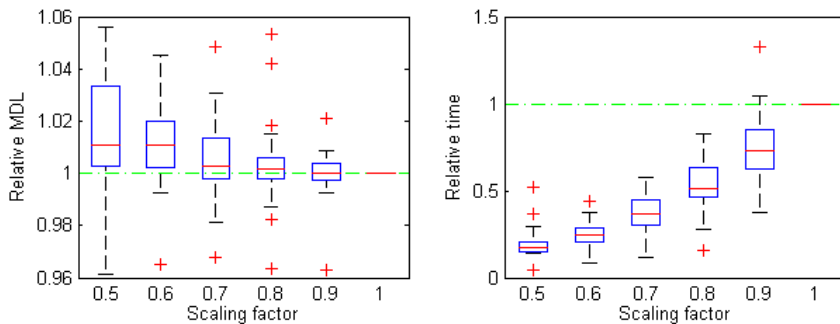


Fig. 1. Scaling experiment with scales $s_i \in \{0.9; 0.8; 0.7; 0.6; 0.5\}$: (Left) Boxplot of the relative MDL values $r_{MDL}(s_0, s_i, I_j) = \frac{MDL(s_i, I_j)}{MDL(s_0, I_j)}$. (Right) Boxplot for the relative time values $\frac{t(s_i, I_j)}{t(s_0, I_j)}$, where $t(s_i, I_j)$ is the execution time of SNEF for image I_j and scale s_i . The reference scale is $s_0 = 1$.

We make an experiment where we scale the original images by five scaling factors, $s_i \in \{0.9; 0.8; 0.7; 0.6; 0.5\}$, so that the image dimensions are decreased up to 0.5 of the original dimensions. The interpolation filter used here is a bicubic spline filter, which uses 16 neighboring pixels in the original grid for finding the gray level in each new grid point [12].

As a side effect, the interpolation filter is also smoothing the image, so that part of the unwanted patterns in the original image are blurred, e.g. the chromatin texture is smoothed, and also some noise in the image is removed. On the other hand, the interpolation filtering may damage the true gradients relevant for the identification of nuclei edges.

After the ellipse coordinates are found in the downscaled image we map the ellipse geometrical parameters back to the original size of the image, and compute the MDL for describing the original image based on the found ellipses. That will give a fair basis for comparing MDL values.

We first define the relative MDL value for the segmentation of a given image I_j at a scale s_i as the ratio

$$r_{MDL}(s_0, s_i, I_j) \triangleq \frac{MDL(s_i, I_j)}{MDL(s_0, I_j)}$$

between the MDL values for the segmentations obtained at scale s_i and at scale $s_0 = 1$. Figure 1 shows boxplot representations of the relative MDL at various scales. In the right panel of Figure 1 are shown the relative execution times at various scales. At each scale value the distribution of the relative values obtained for the 24 images I_j is illustrated as follows: the median is a red line, the quartiles above and below the median are enclosed as full boxes, and black whiskers delimitate the range of the values not considered outliers, while outliers (when they exist) are shown as red crosses.

The median value of the relative MDL increases with the reduction of the image sizes, showing that a stronger scaling in average increases the MDL values. However, for some of the images the relative MDL values are subunitary, which correspond to better image representations and better segmentations.

The execution times of the SNEF algorithm are decreasing with the size of the image faster than quadratically. In the panels of column three of Figure 3 (marked as column c), we show the results of SNEF algorithm for three scaling factors. For the first image the downscaling by $s = 0.8$ produced the correct results, while for the image in second row scaling by $s = 1$ or by $s = 0.8$ gave correct results.

3.3 Smoothing with Various Support Sizes

A second type of transformation of the original image in the spatial domain is smoothing, which has two main goals: reduce some of the non-desired patterns in the image (like the effects of chromatin non-homogeneity, which may lead to fake edges) and reduce the level of noise in the image. However smoothing may affect negatively the ability to detect the correct edges, which may become blurred.

We use in our experiments Gaussian filtering, in which the image is convolved with the Gaussian kernel

$$K(x, y) = C^{-1} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

having nonzero values only on the support (x, y) with $x, y \in \{-1, 0, 1\}$. The normalization constant is

$$C = \sum_{i \in \{-1, 0, 1\}} \sum_{j \in \{-1, 0, 1\}} e^{-\frac{i^2+j^2}{2\sigma^2}}.$$

We experiment with five standard deviations: $\sigma_i \in \{0.2; 0.4; 0.6; 0.8; 1.0\}$, which will have an increasing effect of smoothing.

Figure 2 presents similar boxplots as for the scaling experiment, but now for the segmentations obtained with the SNEF algorithm starting from the smoothed images having various smoothing strength σ . The relative values for the MDL and the times are reported relative to the MDL and time obtained for the original image, for which we formally have $\sigma_0 = 0$. The benefits of smoothing are much better than those obtained by scaling. Almost in half of the cases the MDL of the smoothed images were better than the MDL of the original image, hinting that if time allows, one would better execute the segmentation algorithm twice, once starting from the original image and second time from a smoothed image with a smoothing factor $\sigma \approx 0.5$, and then one can choose between the two results based on the MDL value of the two segmentations. In the panels of column four of Figure 3 we show the results of SNEF algorithm for three smoothing parameters. For the image in the first row the initial segmentation, of the image without smoothing, produced erroneously joining two nuclei (ellipses in yellow), while a too large smoothing factor produced again erroneous results (ellipses in cyan).

However, the moderate smoothing factor $\sigma=0.4$ produced correct results (ellipses in green). For the image in the second row, all smoothing parameters led to correct segmentations.

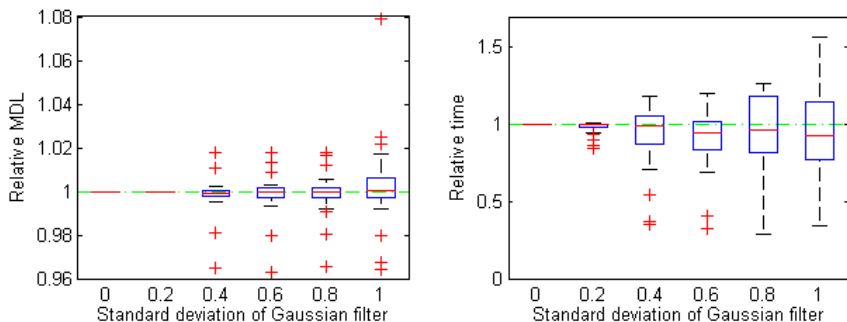


Fig. 2. Smoothing experiment with smoothing parameters $\sigma_i \in \{0.2; 0.4; 0.6; 0.8; 1.0\}$: (Left) Boxplot of the relative MDL values $\frac{MDL(\sigma_i, I_j)}{MDL(\sigma_0, I_j)}$ (Right) Boxplot for the relative time values $\frac{t(\sigma_i, I_j)}{t(\sigma_0, I_j)}$, where $t(\sigma_i, I_j)$ is the execution time of SNEF for image I_j and smoothing parameter σ_i .

3.4 Constructing a Set of Ground Truth Ellipses

We extracted a number of 24 images of clumps, which represent overlapping nuclei. In order to obtain ground truth ellipses for these images we asked five human subjects to draw ellipses overlaid over the colour H&E image, as much in line with the visual clues present in the image, so that each ellipse will correspond to one nucleus (only nuclei fully contained in the image were considered). We constructed a graphical interface in matlab, where each subject could draw ellipses and move them until they reach the convenient position. Each subject could specify number of different interpretations for each image, expressed as various collections of ellipses, and associate to each collection a degree of confidence. In the panels of column two of Figure 3 we show with thin red lines the ellipses drawn by all five subjects for the two original images. We processed these ellipses so that we group in clusters the ellipses corresponding to the same nucleus, and then averaged them to get the ellipses represented with thick cyan lines, which we consider in the rest of the paper as ground truth ellipses. During this process, we had to discard some of the collections (usually low confidence alternative interpretations given by some subjects), if a collection of ellipses cumulated only a degree of confidence smaller than a threshold. The ground truth images were validated by the expert pathologist.

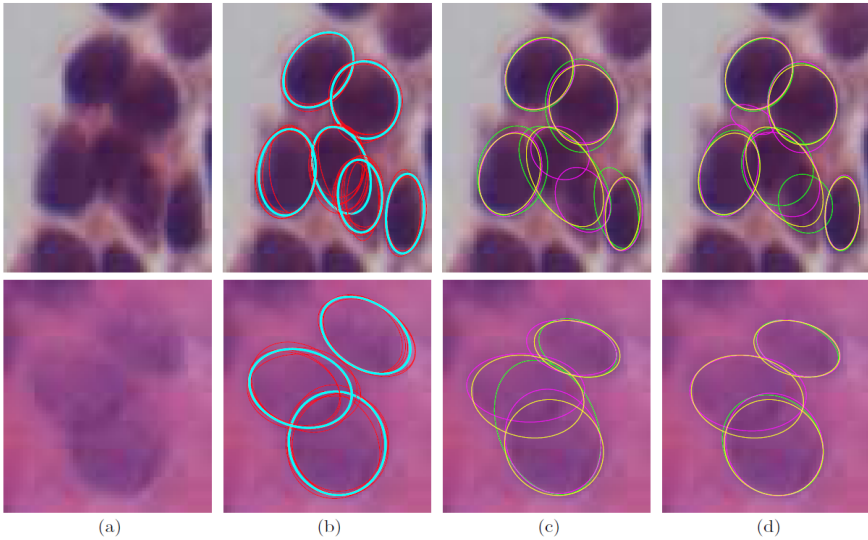


Fig. 3. (a) Original Image. (b) Ellipse drawn by human subjects (thin red) and their averaged ellipses (bold cyan). (c) SNEF results after scaling by values of 1 (yellow), 0.8 (magenta) and 0.6 (green). (d) SNEF results after lowpass Gaussian filtering with values of standard deviations of 0 (yellow), 0.4 (magenta), and 0.8 (green).

3.5 The Conventional Dice Coefficients for Scoring a Segmentation against the Ground Truth Ellipses

The SNEF algorithm was used to produce automatic segmentations. We measure degrees of similarity of the automatic segmentation with the ground truth produced by averaging the subject segmentation by two Dice coefficient [2] based similarity indexes.

Consider two segmentations, one with the set $\{\mathcal{E}_{1i}\}_{i=1,\dots,n_{E_1}}$ of n_{E_1} ellipses and the second with the set $\{\mathcal{E}_{2i}\}_{i=1,\dots,n_{E_2}}$ of n_{E_2} ellipses. The first similarity index scores the matching of foreground (union of ellipses) between the two sets of ellipses, and we refer to this index as area similarity:

$$S_A = \frac{2|(\cup_{i=1}^{n_{E_1}} \mathcal{E}_{1i}) \cap (\cup_{i=1}^{n_{E_2}} \mathcal{E}_{2i})|}{|(\cup_{i=1}^{n_{E_1}} \mathcal{E}_{1i})| + |(\cup_{i=1}^{n_{E_2}} \mathcal{E}_{2i})|}$$

where $|\Phi|$ denotes the cardinality of a set Φ . We notice that by this index, we evaluate the agreement between the overall foregrounds in the clumps, but we do not measure in any ways the particular split of the clump into ellipses.

The second similarity index, S_E , is more refined, by accounting for the pairwise agreement between corresponding ellipses, in two different interpretations. This similarity index is asymmetrical, depending on which collection of ellipses of the

two is chosen to be the basis of fitting, and we chose always as the first collection the one having the highest number of ellipses. To compute the index we first allocate each ellipse \mathcal{E}_{1j} from the first set to an ellipse from the second set $\{\mathcal{E}_{2i}\}_{i=1,\dots,n_{E_2}}$ by choosing the ellipse $\mathcal{E}_{2i(j)}$ having the largest intersection with \mathcal{E}_{1j} . We compute pairwise similarities

$$S_{\mathcal{E}_{1j}, \mathcal{E}_{2i(j)}} = \frac{2 |\mathcal{E}_{1j} \cap \mathcal{E}_{2i(j)}|}{|\mathcal{E}_{1j}| + |\mathcal{E}_{2i(j)}|},$$

and denote their average

$$S_{\mathcal{E}} = \frac{1}{n_{E_1}} \sum_{j=1}^{n_{E_1}} S_{\mathcal{E}_{1j}, \mathcal{E}_{2i(j)}}.$$

In Tables 1 and 2, we show the values of MDL and the two similarity indexes, S_A and $S_{\mathcal{E}}$, for seven images and in the last row the average over 24 images. The results for the two images shown in Figure 3 are presented in the first two rows of the table. The most desirable situation will be when all considered performance criteria do agree, i.e. the supervised DICE area, S_A , similarity indices will indicate the winner among the six segmentations of a particular image in agreement with the DICE ellipse, $S_{\mathcal{E}}$, and also in agreement with the lowest MDL criterion for the six segmentations. In Tables 1 and 2 one can see that this desirable agreement holds for most of the images, the best MDL value corresponds to the highest DICE similarity. For those images where the full agreement does not hold, the segmentation chosen by the smallest MDL value will be the second best (or third best) in the ranking of the supervised DICE indices.

The agreement between the three performance indices is visualized in Figures 4-5, where we present for all the 24 images the supervised similarity indices (represented as blue points) between the ground truth segmentation and the given segmentations (SNEF or human subjects). Hence, in Figures 4-5 there are four different plots, based on which similarity index (S_A or $S_{\mathcal{E}}$) is used, and on who has proposed the segmentation (the SNEF algorithm or the human subjects). For SNEF algorithm, we have altogether 11 different segmentations induced by different scaling and smoothing experiments. Human subjects provided us 5 different segmentations. In figures 4 and 5, we have marked by red circles the segmentation with the lowest obtained MDL value and correspondingly by black square the segmentation with the worst (highest) MDL value. Ideally, the segmentation chosen by the unsupervised MDL (which does not use a ground truth for its computation) should agree with the ranking given by the supervised S_A , which uses the ground truth for its evaluation.

In Figure 4 we have presented the results using the similarity index S_A . On left plot in the Figure 4 are subject segmentations and on the right subjects' segmentations. It can be seen (Figure 4, left) that the points with lowest MDL values are the same or close to the points having the highest area-wise similarity index S_A , especially for SNEF segmentations (Figure 4, left). This shows that SNEF combined with MDL criterion is finding the best area-wise segmentation among the proposed SNEF segmentations. In addition, the S_A values obtained by SNEF (Figure 4, left) are in most cases close to subjects' ones (Figure 4, right).

Table 1. The three performance criteria for evaluating various segmentations: *relative* MDL (rMDL) (smaller is better), similarity index S_A (larger is better), and similarity index S_ε (larger is better), in the image scaling experiment with scales $s_i \in \{0.9; 0.8; 0.7; 0.6; 0.5\}$. Column 3 shows the average of the performance criteria over the five segmentations provided by the five different subjects. Columns 4 to 9 show the performance index of the segmentation obtained by SNEF algorithm over the scaled image. The images 1, 2, 4, and 6 show a complete agreement of the criteria in choosing the best scaling parameter.

Image	Criterion	Subjects			Scaling			
		Average	0.5	0.6	0.7	0.8	0.9	1
1	rMDL	1.023	1.021	1.016	1.013	1.005	1.000	1.000
	S_ε	0.926	0.584	0.793	0.822	0.878	0.907	0.857
	S_A	0.960	0.882	0.929	0.918	0.938	0.946	0.945
2	rMDL	1.034	1.056	1.018	0.997	0.999	0.995	1.000
	S_ε	0.940	0.688	0.668	0.835	0.831	0.842	0.820
	S_A	0.951	0.860	0.828	0.880	0.880	0.882	0.878
3	rMDL	1.019	1.011	1.009	1.010	1.007	0.999	1.000
	S_ε	0.709	0.687	0.698	0.693	0.715	0.872	0.884
	S_A	0.867	0.897	0.882	0.885	0.887	0.917	0.932
4	rMDL	1.021	0.997	0.993	0.983	0.987	0.993	1.000
	S_ε	0.851	0.547	0.531	0.723	0.705	0.506	0.484
	S_A	0.910	0.782	0.768	0.864	0.835	0.691	0.653
5	rMDL	1.018	1.007	1.008	1.003	1.005	1.003	1.000
	S_ε	0.915	0.778	0.869	0.892	0.886	0.814	0.917
	S_A	0.943	0.924	0.922	0.928	0.928	0.932	0.942
6	rMDL	1.018	0.997	1.011	0.995	0.999	0.997	1.000
	S_ε	0.939	0.884	0.779	0.917	0.870	0.761	0.749
	S_A	0.947	0.924	0.822	0.929	0.893	0.925	0.924
7	rMDL	1.024	1.002	1.021	1.014	1.018	0.999	1.000
	S_ε	0.923	0.865	0.819	0.834	0.830	0.909	0.919
	S_A	0.930	0.908	0.826	0.842	0.836	0.915	0.937
Averages over 24 images								
	rMDL	1.017	1.011	1.011	1.003	1.002	1.000	1.000
	S_ε	0.901	0.688	0.733	0.834	0.830	0.866	0.839
	S_A	0.939	0.880	0.860	0.905	0.896	0.915	0.906

Table 2. The three performance criteria for the same set of images as in Table 1, for evaluating the segmentations in the image smoothing experiment. Columns 4 to 9 show the performance index of the segmentation obtained by SNEF algorithm over the smoothed image, for six smoothing parameters, $\sigma_i \in \{0; 0.2; 0.4; 0.6; 0.8; 1.0\}$.

Image	Criterion	Subjects	Gaussian filtering					
		Average	0	0.2	0.4	0.6	0.8	1
1	rMDL	1.023	1.000	1.000	1.018	1.018	0.998	0.999
	S_ε	0.926	0.857	0.857	0.819	0.841	0.901	0.892
	S_λ	0.960	0.945	0.945	0.919	0.913	0.944	0.944
2	rMDL	1.034	1.000	1.000	1.002	0.998	1.000	1.079
	S_ε	0.940	0.820	0.820	0.816	0.845	0.825	0.619
	S_λ	0.951	0.878	0.878	0.868	0.886	0.873	0.730
3	rMDL	1.019	1.000	1.000	0.998	0.997	0.999	1.005
	S_ε	0.709	0.884	0.884	0.884	0.872	0.850	0.724
	S_λ	0.867	0.932	0.932	0.924	0.921	0.911	0.891
4	rMDL	1.021	1.000	1.000	0.995	0.993	0.992	0.968
	S_ε	0.851	0.484	0.484	0.497	0.520	0.507	0.842
	S_λ	0.910	0.653	0.653	0.675	0.687	0.696	0.905
5	rMDL	1.018	1.000	1.000	1.002	1.001	1.012	0.999
	S_ε	0.915	0.917	0.917	0.921	0.908	0.769	0.904
	S_λ	0.943	0.942	0.942	0.940	0.936	0.887	0.941
6	rMDL	1.018	1.000	1.000	1.000	0.998	0.997	1.010
	S_ε	0.939	0.749	0.749	0.752	0.874	0.702	0.618
	S_λ	0.947	0.924	0.924	0.924	0.905	0.919	0.794
7	rMDL	1.024	1.000	1.000	1.011	1.009	1.001	1.001
	S_ε	0.923	0.919	0.919	0.855	0.865	0.908	0.896
	S_λ	0.930	0.937	0.937	0.861	0.868	0.922	0.917
Averages over 24 images								
	rMDL	1.017	1.000	1.000	0.999	0.999	0.999	1.000
	S_ε	0.901	0.839	0.839	0.858	0.869	0.839	0.849
	S_λ	0.939	0.906	0.906	0.916	0.911	0.910	0.910

The corresponding results for the other similarity index, the ellipse-wise similarity index S_ε , are presented in Figures 5, where larger differences are observed between the highest S_ε (highest blue point) and the S_ε corresponding to best MDL (red circles). This might imply that occasionally ellipse set ranked the best by the MDL does not agree with the segmentation ranked the best by S_ε .

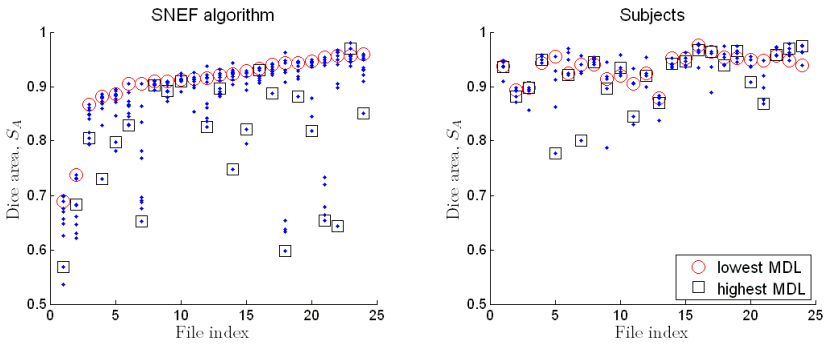


Fig. 4. (Left) Supervised similarity index S_A (represented as blue points) between the ground truth segmentation and the eleven segmentations given by SNEF algorithm for different smoothing and scaling parameters, for each of the 24 files. By red circle we mark the similarity index S_A of the segmentation having the lowest MDL value out of the 11 segmentations for each file and by black square we mark the segmentation with the worst (highest) value. Ideally, the segmentation chosen by the unsupervised MDL (which does not use a ground truth for its computation) should agree with the ranking given by the supervised S_A , which uses the ground truth for its evaluation. (Right) Similarly, the right plot also shows the ranking by MDL and by supervised similarity index S_A , except that for each file, instead of 11 SNEF segmentations we use the 5 different segmentations provided by the 5 subjects. The plot illustrates the variability among the segmentations provided by the subjects.

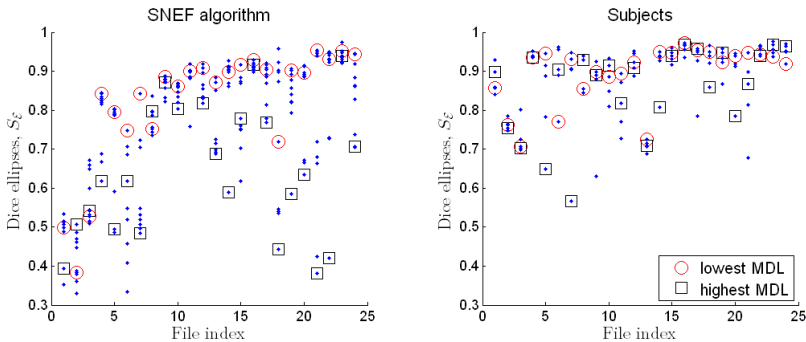


Fig. 5. Plots comparing the best segmentation according to MDL with the ranking of the segmentations using the similarity index S_E . The left and right plots correspond to the ones in Figure 4, where the similarity index S_A was used instead of S_E .

4 Conclusions

We compared the segmentations obtained by the SNEF algorithm under various preprocessing applied to the original image, by using three criteria: MDL, similarity index S_A , and similarity index S_E . The first criterion, MDL, can be

computed in an unsupervised way, while the last two can be computed only in a supervised way, based on ground truth defined by human subjects. We noticed a good correlation between the best MDL and the best similarity indexes, showing that MDL can be integrated in the segmentation routines for performing a decision among various segmentations, produced e.g. by the same algorithm on various preprocessed versions of the image to be analyzed.

References

- [1] Bai, X., Sun, C., Zhou, F.: Splitting touching cells based on concave points and ellipse fitting. *Pattern Recognition* 42, 2434–2446 (2009)
- [2] Dice, L.: Measures of the amount of ecologic association between species. *Ecology* 26, 297–302 (1945)
- [3] Ghido, F., Tabus, I.: Performance of sparse modeling algorithms for predictive coding. In: *Proc. of the 18th International Conference on Control Systems and Computer Science (CSCS-18)*, Bucharest, Romania, May 24-27 (2011)
- [4] Gurcan, M.N., Boucheron, L.E., Can, A., Madabhushi, A., Rajpoot, N.M., Yener, B.: Histopathological image analysis: a review. *IEEE Reviews in Biomedical Engineering* 2, 147–171 (2009)
- [5] Hukkanen, J., Hategan, A., Sabo, E., Tabus, I.: Segmentation of cell nuclei from histological images by ellipse fitting. In: *Proc. of the European Signal Processing Conference*, Aalborg, Denmark, pp. 1219–1223 (2010)
- [6] Hukkanen, J., Sabo, E., Tabus, I.: Representing clumps of cell nuclei as unions of elliptic shapes by using the MDL principle. In: *Proc. of the European Signal Processing Conference*, Barcelona, Spain, pp. 1010–1014 (2011)
- [7] Kanungo, T., Dom, B., Niblack, W., Steele, D.: A fast algorithm for MDL-based multi-band image segmentation. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 609–616 (1994)
- [8] Kumar, S., Ong, S., Ranganath, S., Ong, T., Chew, F.: A rule-based approach for robust clump splitting. *Pattern Recognition* 39, 1088–1098 (2006)
- [9] Leclerc, Y.: Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision* 3, 73–102 (1989)
- [10] Luo, Q., Khoshgoftaar, T.M.: Unsupervised multiscale color image segmentation based on MDL principle. *IEEE Transactions on Image Processing* 15(9), 2755–2761 (2006)
- [11] Onose, A., Dumitrescu, B.: Sparsity estimation for greedy RLS filters via information theoretic criteria. In: *Proc. of the 18th International Conference on Control Systems and Computer Science (CSCS-18)*, Bucharest, Romania, May 24-27 (2011)
- [12] Ward, J., Cok, D.: Resampling algorithms for image resizing and rotation. In: *Proc. SPIE Digital Image Processing Applications*, vol. 1075, pp. 260–269 (1989)
- [13] Weinberger, M.J., Seroussi, G., Sapiro, G.: LOCO-I: A low complexity, context-based, lossless image compression algorithm. In: *Proc. of the IEEE Data Compression Conference*, Snowbird, Utah (1996)

Retinopathy of Prematurity: Fractal Analysis of Images in Different Stages of the Disease

Nebojša T. Milošević¹, Maja Olujić², Ana Oros², and Herbert F. Jelinek³

¹Department of Biophysics, Medical Faculty, University of Belgrade,
KCS Institute of Biophysics pp. 129, 11129 Belgrade 102, Serbia
{mtn@med.bg.ac.rs}

²Department of Ophthalmology, Medical Faculty, University of Novi Sad, Hajduk Veljkova
3, 21000 Novi Sad, Serbia
{molujic; annaoros03}@yahoo.com

³School of Community Health, Charles Sturt University, Albury, 2640, Australia
{hjelinek@csu.edu.au}

Abstract. Retinopathy of prematurity is a potentially blinding proliferative disease of the retinal vasculature that affects premature infants with low birth weights. This paper explores the possibility of applying fractal analysis to images of blood vessels in the human retinae with three types of retinopathy. Although the fractal analysis suggests that the retinal images with different grade of retinopathy are more complex than those without retinopathy, this technique cannot completely distinguish retinal vessels in a patient with inactive and active retinopathy. Furthermore, the fractal dimension of retinal images in children with diagnosed aggressive retinopathy is higher than the fractal dimensions of the same images after laser surgery. Our results lead to the conclusion that the blood vessels in the retinae with aggressive retinopathy are more complex, irregular in shape and have a higher degree of vessel aberration than those vessels with ‘typical’ retinopathy.

Keywords: Blood vessel, Box-counting method, Fractal analysis, Human retina; Medical imaging, Retinal vasculature.

1 Introduction

A number of human disorders are associated with obliteration of preexisting blood vessels. Microvessel rarefaction often takes place in the hypertensive lung, in the myocardium of patients with chronic renal failure, and in the elderly [1]. Conversely, a failure to eliminate transient embryonic vasculature destined for regression may also lead to disease, as exemplified by persistent hyperplastic primary vitreous, a common congenital developmental anomaly of the eye in which hyaloid vessels fail to regress [2]. A striking example of a disease caused by vessel regression is retinopathy of prematurity (ROP).

ROP is a blindness-causing neovascularizing disease that affects premature infants treated with high concentrations of oxygen. ROP develops in two distinct stages [1]. First, the hyperoxic insult leads to obliteration of immature retinal vessels, thereby compromising retina perfusion. The second phase, initiated upon resumption of the breathing of normal air, is an adverse compensatory neovascularization response, mediated by ischemia-induced vascular endothelial growth factor (VEGF), in which formation of new vessels is excessive, new vessels are leaky, and the inner limiting membrane of the retina is breached, allowing vessel growth into the vitreous. The latter event may ultimately lead to retinal detachment and vision loss [3].

The landmark multicenter trial entitled Cryotherapy for ROP (CRYO-ROP) was critical for describing the natural progression of ROP. The study defined the "threshold" of the disease when ablation to the peripheral avascular retina should be performed [3,4]. The Early Treatment of Retinopathy study showed that treatment was beneficial for high-risk eyes that did not yet meet the CRYOROP criteria for threshold disease [5].

The trend toward early treatment of high-risk patients further emphasizes the importance for early detection of ROP, particularly of aggressive ROP. The condition was highlighted in the 2005 updated classification of ROP and is considered the most virulent form of the disorder [3,5]. Aggressive ROP occurs among the smallest of low-birth-weight babies. It is characterized by its posterior location, prominence of plus disease, and the ill-defined nature of the retinopathy [3].

Aggressive ROP progresses rapidly into more severe forms of the disease and may not follow the classic stages of typical ROP [5]. The junction of vascular and avascular retina can be subtle and more easily overlooked during examinations. In addition, the neovascularization of aggressive ROP is less obvious due to its growth along the retinal surface, rather than into the vitreous cavity.

This study explores the possibility of applying fractal analysis to images of blood vessels in the human retinae with different stages of the ROP. The aim of this study was to investigate the use of the fractal dimension as a parameter, which quantifies the complexity, shape and straightness of retinal blood vessels, and to explore whether these parameters were capable to distinguish retinal images with different ROP. In addition this study aimed to quantify the difference among retinal blood vessels in patients with aggressive ROP before and after laser surgery.

2 Materials and Methods

2.1 Patients and Image Acquisition

On the basis of ocular findings, 43 infants were identified from a cohort of 71 infants and with ROP in zone 1 (posterior pole) and included in this study. The characteristics recorded for each infant in this study included birth weight, gestational age at birth, gestational age at laser treatment and during the post laser treatment follow-up at one or two weeks [3]. Ocular fundus photography was performed as a part of ROP screening in all infants who had been hospitalized at

the Institute for Neonatology, Belgrade, Serbia during the period of March-June 2010. The screening criteria included neonates with a gestational age of 36 weeks or younger or those weighing less than 2000 grams.

During the screening examination, the RetCam3 (Clarity Medical Systems, CA, USA), equipped with 130° lens, was used to obtain colour images of the retina of both eyes. Pupillary dilation was achieved with two cycles of 0.5% cyclopentolate and 2.5% phenylephrine at 30 min intervals of instillation. RetCam 3 images were taken by the ophthalmologist. The goal of each imaging session was to obtain clearly focused images of all parts of the ocular fundus. The images were stored on the RetCam3 computer hard drive (Fig. 1A).

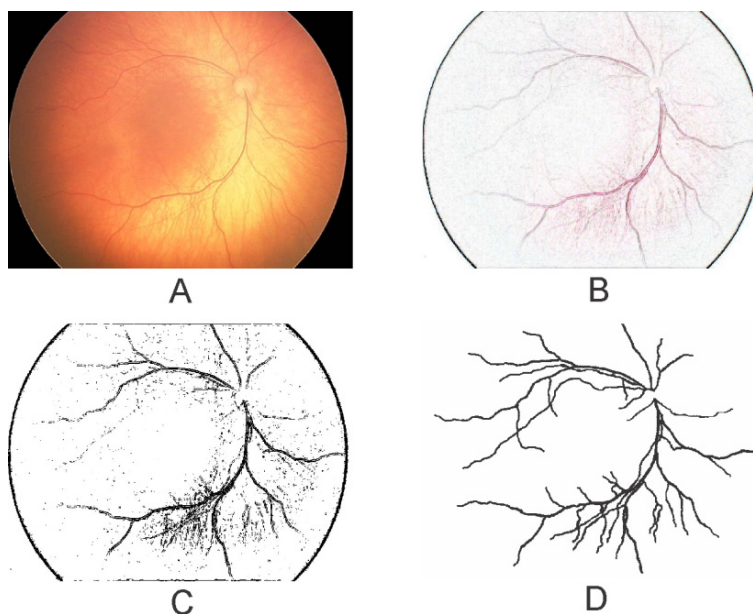


Fig. 1. An example of image acquisition: RGB image recorded with RetCam 3 (A), half-tone inverted image (B), binary image with (C) and without (D) artifacts

Analyzing 106 RGB images, we have collected 84 images and initially set them into 2 major groups, according to the presence of the ROP. Furthermore, we have subdivided the second major group into 3 smaller subgroups, due to the stage of ROP development [3]. Our sample of images, classified according to the previously described scheme, consisted of 24 images in group I (control), 13 images in group II (inactive ROP), 17 images in group III (active ROP) and 30 images in group IV (aggressive ROP). Among 15 patients (i.e. 30 images) with aggressive ROP, in 8 patients laser treatment was performed [6]. For these patients the fundus images were taken during the laser treatment session but before surgery and one or two weeks post-treatment.

Digital RGB images of each retinae were further imported in Image J (www.rsb.info.nih.gov/ij). With three subroutines ‘Filter:Stylize:Halftone’ (Fig. 1B), ‘Image:Adjustment:Threshold’ (Fig. 1C) and ‘Image:Mode:Binary’ (Fig. 1D), all images were converted into binary images, where all remaining artifacts were digitally removed in previous step.

2.2 Fractal Analysis

All binary images were saved in three different formats, as binary, outline and skeletonized images [7], and subjected to fractal analysis, particularly to the box-counting procedure [8,9]. When the box-counting method is applied to 2D images, the fractal dimension (i.e. the box dimension) depends on image presentation. While the box dimension of binary images presents their space-filling property, the same parameter calculated from outline and skeletonized images defines the irregularity in the shape of the image and the degree of vessel aberrations from straight lines [10,11], respectively.

Box-counting method was incorporated into Image J, and for each image the box-counting dimension (D) was obtained. Briefly, this procedure “covers” the image with sets of squares [12,13]. Each set was described by the size of the square edge. The corresponding number of squares necessary to cover the image was presented as a function of the size of square edge. The box dimension was obtained as an absolute value of the (negative) slope of the log-log relationship between number of squares and the size of the square edge [14,15].

The box sizes for the box-counting method were taken from 2^1 to 2^k pixel, where k is the value for which N is equal to unity. As for our material, the box sizes were from 2 (i.e. $0.6 \mu\text{m}$) to 256 pixels ($76.8 \mu\text{m}$), 512 pixels ($153.6 \mu\text{m}$) or 1024 pixels ($307.2 \mu\text{m}$), depending on the image size. In all cases (Fig. 2) the correlation coefficient (R) of the straight line, fitted through 9 (or 10) data points, was higher than 0.95, proving the existence of the linear relationship between the logarithm of the number of squares and the logarithm of the size of the square edge [14].

2.3 Statistical Analysis

Statistical analysis of the calculated box dimension of the retinal images depends on whether the box dimension distribution is normal or not. In the case when the calculated box dimension follows a normal distribution, each group is described by the corresponding mean value and standard deviation (or standard error). Further analysis is performed according to the methods and tests of parametric statistics. Alternatively, when the calculated values do not follow a normal distribution, the groups are described by the medians, and value range, and further analysis proceeds according to non-parametric statistics laws.

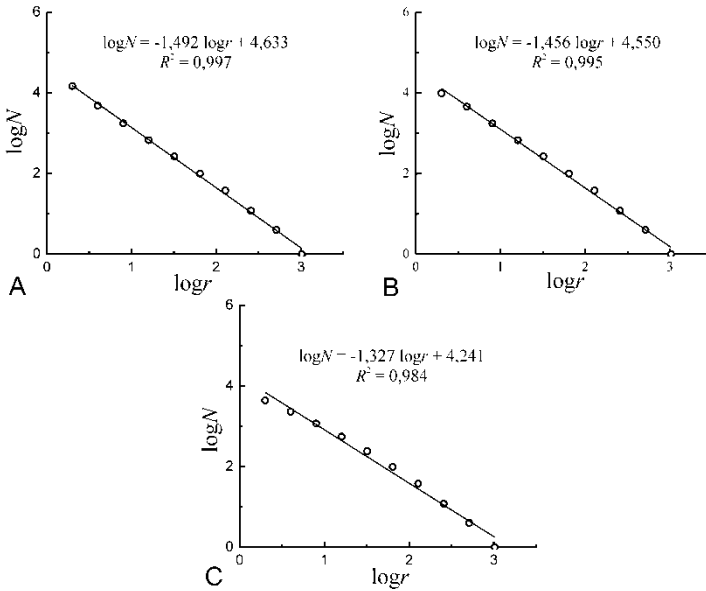


Fig. 2. Log-log plot of relationship between N and r for the binary (A), outline (B) and skeletonized (C) image. The equation of $\log N$ against $\log r$ (where the box dimension is the negative value of the slope of the fitted line) along the determination coefficient (R^2) is presented in upper part of each graph.

It is known that testing whether the data is distributed normally is performed by a chi-square test if the studied population consists of a large number of samples (more than 50). If the sample is small then testing of the distribution character is based on the calculation of two statistical parameters: skewness (a_3) and excess (e) of distribution. The interval of skewness and excess values, which define the normal distribution, is estimated when these two parameter are divided by the corresponding mean square errors (σ_3 and σ_4). If the absolute value of the quotients a_3/σ_3 and e/σ_4 is less than or equal to 2, then the data distribution is normal. Otherwise, the distribution in a sample (or population) has no characteristics of a normal distribution. For further explanation of this procedure, the reader is referred to [15].

3 Results

Table 1 shows the values of the parameters used to assess the nature of the distribution of three box dimensions of retinal images in four groups of patients, as well as in patients before and after surgery. As the values of the quotients $|a_3/\sigma_3|$ and $|e/\sigma_4|$ do not exceed the critical value, i.e., 2, the box dimension values in all cases are normally distributed. Therefore further statistical analysis of the results of the fractal analysis was performed according to parametric statistics laws.

Table 1. The values of the skewness (a_3), excess (e) and corresponding mean square errors (σ_3 and σ_4) in distribution of three box dimensions (D_{bin} , D_{out} and D_{skel}) of retinal images for four group of patients, along with group of patients before (a) and after (b) the surgery

Patients	Parameters	D_{bin}	D_{out}	D_{skel}	
I	a_3	-0.25	0.36	0.68	
	e	-0.74	-0.72	-0.49	
	σ_3		0.45		
	σ_4		0.80		
II	a_3	-0.32	0.28	0.30	
	e	-1.03	-1.61	-1.73	
	σ_3		0.57		
III	a_3	-0.33	-0.27	0.26	
	e	-0.66	-1.34	-1.48	
	σ_3		0.52		
	σ_4		0.87		
IV	a_3	-0.22	0.22	0.24	
	e	-1.10	-1.07	-0.89	
	σ_3		0.41		
a	σ_4		0.75		
	a_3	-0.69	-0.59	-0.56	
	e	-0.82	-0.80	-0.90	
	σ_3		0.53		
b	σ_4		0.88		
	a_3	0.29	0.38	0.32	
	e	-1.29	-1.34	-1.37	
	σ_3		0.53		
			σ_4	0.88	

3.1 Patients with Different Grade of the ROP

The central tendency and dispersion measure of the three box dimensions (D_{bin} , D_{out} and D_{skel}) in four groups of patients are shown in Tab. 2. As can be seen from this table, patients with aggressive retinopathy (group IV) have the highest means of all box dimensions. To investigate whether there are significant differences among these means, a one-way ANOVA was applied. The calculated F -value for all box dimensions was higher than the critical value for a level of confidence $p < 0.01$ (Tab. 2). As the next step in the analysis, we used a t -test in order to assess how many pairs of means D_{bin} , D_{out} and D_{skel} were significantly different.

This test points out that the means of the three box dimensions in group I were highly different than the same means of other groups ($p < 0.001$). As for the mean D_{bin} , all other pairs were significantly different: pair II-III ($p < 0.01$), pair II-IV ($p < 0.05$) and pair III-IV ($p < 0.001$). When it comes to the mean D_{out} and D_{skel} similar results were obtained, a significant difference was found in two pairs: II-III ($p < 0.01$ for both box dimensions) and III-IV ($p < 0.01$ for D_{out} and $p < 0.001$ for D_{skel}).

Table 2. The values of three box dimensions (D_{bin} , D_{out} and D_{skel}) in four groups of patients: control (I), inactive retinopathy (II), active retinopathy (III) and aggressive retinopathy (IV). Each value is presented as the mean \pm standard error. F is the tabulated value of the one-way ANOVA, while $F_{0.05}$ and $F_{0.01}$ are corresponding tabulated values.

Patients	D_{bin}	D_{out}	D_{skel}
I	1.28 \pm 0.01	1.25 \pm 0.01	1.13 \pm 0.01
II	1.41 \pm 0.01	1.371 \pm 0.007	1.250 \pm 0.007
III	1.369 \pm 0.007	1.337 \pm 0.006	1.221 \pm 0.007
IV	1.45 \pm 0.01	1.40 \pm 0.01	1.27 \pm 0.01
F	47.26	50.38	46.18
$F_{0.05}$		2.76	
$F_{0.01}$		4.13	

3.2 Aggressive ROP before and after the Surgery

Table 3 shows means and standard errors of the three box dimensions before and after laser treatment. Appropriate statistical estimations are also shown in the same table. The mean box dimensions before treatment was significantly higher than the same values after treatment. These results suggest a decrease in all box dimensions after laser treatment.

Table 3. The values of three box dimensions (D_{bin} , D_{out} and D_{skel}) in patients with aggressive retinopathy before (A) and after (B) the laser surgery. Each value is presented as the mean \pm standard error. p is the significance level

Patients	A	B	p
D_{bin}	1.47 \pm 0.02	1.39 \pm 0.02	< 0.001
D_{out}	1.39 \pm 0.02	1.32 \pm 0.02	< 0.001
D_{skel}	1.25 \pm 0.03	1.18 \pm 0.02	< 0.001

4 Discussion

4.1 Morphology of the ROP

The system used for describing the findings of active ROP is entitled *The International Classification of Retinopathy of Prematurity*, or simply ICROP [16]. ICROP uses a number of parameters to describe the disease. They are location of the disease into zones (1, 2, and 3), the circumferential extent of the disease based on the clock hours (1-12), the severity of the disease (stage 1-5) and the presence or absence of "Plus Disease". Each aspect of the classification has a technical definition. This classification was used for the major clinical trials and was revised in 2005[17].

The zones are centered on the optic nerve. Zone 1 is the posterior zone of the retina, defined as the circle with a radius extending from the optic nerve to double the distance to the macula. Zone 2 is an annulus with the inner border defined by zone 1 and the outer border defined by the radius determined as the distance from the optic nerve to the nasal ora serrata. Zone 3 is the residual temporal crescent of the retina [17].

The circumferential extent of the disease is described in segments as if the top of the eye were 12 on the face of a clock. For example one might report that there is stage 1 disease for zone 3 clock from 4 to 7 o'clock. The extent is a bit less important since the treatment indications from the Early Treatment for ROP [18].

The stages describe the ophthalmoscopic findings at the junction between the vascularized and avascular retina: stage 1 is a faint demarcation line, stage 2 is an elevated ridge, stage 3 is extraretinal fibrovascular tissue, stage 4 is sub-total retinal detachment and stage 5 is total retinal detachment. In addition, "Plus disease" may be present at any stage. It describes a significant level of vascular dilation and tortuosity observed at the posterior retinal vessels. This reflects the increase of blood flow through the retina [19].

4.2 Image Analysis of the Retinal Blood Vessel

Analysis of the retinal vessel structure is of immense interest for the investigation of diseases that involve structural or functional changes in the vasculature. Changes in retinal vasculature, such as hemorrhages, angiogenesis, increases in vessel tortuosity, blockages and arteriolar-venular diameter ratios are important indicators of, for example, diabetic retinopathy, retinopathy of prematurity and cardiovascular risk [21,22].

There is a growing body of literature on the segmentation of blood vessels from retinal images [23-27]. The vast majority of the literature is based on physically inspired models of vessel appearance, and somewhat dependent on the imaging modality [28]. For example, in fluorescent angiography the vessels appear bright (hyperfluorescent), whereas in red-free imaging the vessels appear dark compared to the retinal bed.

Vessel segmentation algorithms can be coarsely divided into two classes: those that are pixel operator based, and those that track vessels in a local region. The first class involve operators, such as morphological operators, that are applied globally to the whole image, whereas the second class, once it has a point on the vessel only needs to analyze the local region about the vessel to track the vessel through the image [29].

4.3 Fractal Analysis of Retinal Blood Vessels

Fractal analysis is a set of mathematical techniques used to quantify complex structures. Namely, the fractal dimension of a structure expresses numerically the degree of geometrical complexity of that structure [8]. Quantitative analysis of the retinal images can be successfully made by applying the concept of fractal geometry, as the retinal blood vessels is known to have a complex and irregular structure.

To the best of our knowledge, box-counting analysis of the retinal blood vessels has not completely attracted researchers' attention [6], contrary to other methods where the fractal analysis of retinal blood vessels has been studied extensively [276,298-321].

The global fractal dimension of the retinal vasculature has been studied for some time [332,343]. Several research groups have demonstrated that the normal retinal vasculature has fractal-like properties with a global dimension generally falling between 1.60 and 1.88 [20,24,298,354]. Our results suggest the opposite conclusion (see Tabs. 2 and 3). Reasons we identified for this include applying a different fractal method and/or an overall relative insensitivity to the finest level of branching in the segmenting method.

This study explored the possibility of applying of the standard box-counting analysis to the images of blood vessels in the human retinae with different degrees of the ROP. The fractal dimension of retinal images in children with diagnosed ROP is higher than the fractal dimension of same images in children without ROP (Tab. 2). Moreover, our results indicate that the blood vessels in the retinae with aggressive ROP are more complex, irregular in shape and have a higher degree of vessel aberration than those vessels with 'typical' ROP (Tab. 2). We have not been able to discriminate the shape and the degree of vessel aberration between inactive and active ROP retinal vessels (see Subsection 3.1). In addition, our results indicate a difference in the morphology of blood vessels of aggressive ROP images before and after the surgery (Tab. 3).

5 Conclusion

The results shown in this study are encouraging. The fractal dimension has proved to be a useful parameter for quantifying the complexity, shape and degree of vessel aberration in retinal images of ROP, along with images of aggressive ROP before and after laser treatment. We strongly believe that this study can be considered as a good start for further investigation of the quantification of retinal images in patients with ROP. Such analysis of the blood vessels can lead to a more reliable interpretation of the disease in the retinae, disease progression and treatment effectiveness.

References

1. Keshet, E.: Preventing pathological regression of blood vessels. *J. Clin. Invest.* 112, 27–29 (2003)
2. Upalakalin, J.N., Hemo, I., Dehio, C., Keshet, E., Benjamin, L.E.: Survival mechanisms during vascular remodeling. *Cold Spring Harb. Symp. Quant. Biol.* 67, 181–187 (2002)
3. Cryotherapy for Retinopathy of Prematurity Cooperative Group: Multicenter trial for cryotherapy for retinopathy of prematurity. Ophthalmological outcomes at 10 years. *Arch. Ophthalmol.* 119, 1110–1118 (2001)

4. Cryotherapy for Retinopathy of Prematurity Cooperative Group: Multicentre trial for cryotherapy for retinopathy of prematurity: preliminary results. *Arch. Ophthalmol.* 106, 471–479 (1988)
5. Azuma, N., Ishikawa, K., Hama, Y., Hiraoka, M., Suzuki, Y., Nishina, S.: Early vitreous surgery for aggressive posterior retinopathy of prematurity. *Am. J. of Ophthalmol.* 142, 636–643 (2006)
6. Olujčić, M., Milošević, N.T., Oros, A., Jelinek, H.F.: Aggressive posterior retinopathy of prematurity: fractal analysis of images before and after laser surgery. In: Dumitrache, I. (ed.) *Proceedings CSCS-18*, vol. 2, pp. 877–881. Editura Politehnica Press, Bucharest (2011)
7. Milošević, N.T., Ristanović, D., Rajković, K.: Mathematical model of box-counting analysis in the human dentate nucleus during development. In: *Book of Abstracts of Talks Presented at Mini-Symposia at the 8th European Conference on Mathematical and Theoretical Biology, and Annual Meeting of the Society for Mathematical Biology*, Kraków, Poland, p. 435 (2011)
8. Fernández, E., Jelinek, H.F.: Use of fractal theory in neuroscience: methods, advantages, and potential problems. *Methods* 24, 309–321 (2001)
9. Milošević, N.T., Krstonošić, B., Ristanović, D., Gudović, R.: Fractal analysis of neuronal dendritic branching pattern in the human neostriatum: a revised classification scheme. In: Dumitrache, I. (ed.) *Proceedings CSCS-18*, vol. 2, pp. 871–876. Editura Politehnica Press, Bucharest (2011)
10. Ristanović, D., Milošević, N.T., Stefanović, B.D., Marić, D.L., Rajković, K.: Morphology and classification of large neurons in the adult human dentate nucleus: a qualitative and quantitative analysis of 2D images. *Neurosci. Res.* 67, 1–7 (2010)
11. Jelinek, H.F., Fernández, E.: Neurons and fractals: how reliable and useful are calculations of fractal dimensions? *J. Neurosci. Meth.* 81, 9–18 (1998)
12. Smith Jr., T.G., Lange, G.D., Marks, W.B.: Fractal methods and results in cellular morphology: dimensions, lacunarity and multifractals. *J. Neurosci. Meth.* 69, 123–136 (1996)
13. Milošević, N.T., Ristanović, D., Jelinek, H.F., Gudović, R., Marić, D.: The Morphology and Cell Classification in The Human Dentate Nucleus: a Fractal Analysis Study. In: Dumitrache, I. (ed.) *Proceedings CSCS-17*, vol. 3, pp. 54–57. Editura Politehnica Press, Bucharest (2009)
14. Jelinek, H.F., Milošević, N.T., Ristanović, D.: The Morphology of Alpha Ganglion Cells in Mammalian Species: a Fractal Analysis Study. *J. CEAI* 12, 3–9 (2010)
15. Milošević, N.T., Ristanović, D., Stanković, J.B., Gudović, R.: Fractal analysis of dendritic arborisation patterns of stalked and islet neurons in substantia gelatinosa of different species. *Fractals* 15, 1–7 (2007)
16. Committee for the Classification of Retinopathy of Prematurity: An international classification of retinopathy of prematurity. *Arch. Ophthalmol.* 102, 1130–1134 (1984)
17. Committee for the Classification of Retinopathy of Prematurity: The International Classification of Retinopathy of Prematurity revisited. *Arch. Ophthalmol.* 123, 991–999 (2005)
18. Early Treatment for Retinopathy of Prematurity Cooperative Group: Revised indications for the treatment of retinopathy of prematurity: results of the early treatment for retinopathy of prematurity randomized trial. *Arch. Ophthalmol.* 121, 1684–1696 (2003)

19. Guyton, A., Hall, J.: Chapter 17: Local and Humoral Control of Blood Flow by the Tissues. In: Gruliow, R. (ed.) *Textbook of Medical Physiology*, 11th edn. Elsevier, Philadelphia (2006)
20. Jelinek, H.F., Depardieu, C., Lucas, C., Cornforth, D., Huang, W., Cree, M.J.: Towards vessel characterization in the vicinity of the optic disk in digital retinal images. In: *Proceeding of Image and Vision Computing New Zealand 2005*, Otago, New Zealand, pp. 351–357 (2005)
21. Nekovei, R., Sun, Y.: Back-propagation network and its configuration for blood vessel detection in angiograms. *IEEE Trans. Neur. Net.* 6, 64–72 (1995)
22. Soares, J.V.B., Leandro, J.J.G., Cesar Jr., R.M., Jelinek, H.F., Cree, M.J.: Retinal vessel segmentation using the 2-D Gabor wavelet and supervised classification. *IEEE Trans. Med. Imag.* 25, 1214–1222 (2006)
23. Gao, X.W., Bharath, A., Stanton, A., Hughes, A., Chapman, N., Thom, S.: Quantification and characterisation of arteries in retinal images. *Comp. Meth. Prog. Biomed.* 63, 133–146 (2000)
24. Cesar Jr., R.M., Jelinek, H.F.: Segmentation of retinal fundus vasculature in nonmydriatic camera images using wavelets. In: Suri, J.S., Laxminarayan, S. (eds.) *Angiography and Plaque Imaging*, pp. 193–224. CRC Press, Boca Raton (2003)
25. Staal, J., Abramoff, M.D., Niemeijer, M., Viergever, M.A., van Ginneken, B.: Ridgebased vessel segmentation in color images of the retina. *IEEE Trans. Med. Imag.* 23, 501–509 (2004)
26. Lowell, J., Hunter, A., Steel, D., Basu, A., Ryder, R., Kennedy, R.L.: Measurement of retinal vessel widths from fundus images based on 2-D modeling. *IEEE Trans. Med. Imag.* 23, 1196–1204 (2004)
27. Karperien, A., Jelinek, H.F., Leandro, J.J.G., Soares, J.V.B., Cesar Jr., R.M., Luckie, A.: Automated detection of proliferative retinopathy in clinical practice. *Clin. Ophthalmol.* 2, 109–122 (2008)
28. Landini, G.: Quantitative analysis of the epithelial lining architecture in radicular cysts and odontogenic keratocysts. *Head Face Med.* 2, 1–9 (2006)
29. Daxer, A.: Fractals and retinal vessels. *Lancet* 339, 618 (1992)
30. Cornforth, D., Jelinek, H.F., Peichl, L.: Fractop: a tool for automated biological image classification. In: *6th AI Australasia-Japan Joint Workshop*, pp. 141–148. Australian National University, Canberra (2002)
31. Masters, B.R.: Fractal analysis of the vascular tree in the human retina. *Annual Rev. Biomed. Eng.* 6, 427–452 (2004)
32. Iannaccone, P.M., Khokha, M.: *Fractal geometry in biological systems: an analytical approach*. CRC Press, Boca Raton (1995)
33. Family, F., Masters, B.R., Platt, D.E.: Fractal pattern formation in human retinal vessels. *Physica D* 38, 98–103 (1989)
34. Masters, B.R.: Fractal analysis of human retinal vessels. In: *SPIE Proc.*, vol. 1357, pp. 250–256 (1989)
35. Landini, G., Misson, G.P., Murray, P.I.: Fractal analysis of the normal human retinal fluorescein angiogram. *Curr. Eye Res.* 12, 23–27 (1993)

Time Series Analysis of Blood Flow in Kidney Graft: On the Emergence of Deterministic Chaos during the Phase of Acute Rejection

Przemyslaw Waliszewski^{1,2}

¹ Department of Urology, Andrology and Pediatric Urology,
Justus-Liebig-University Rudolf-Buchheim-Str. 7,
39352 Giessen, Germany

² Complexity Research Inc., Os. J III Sobieskiego 41/21,
60688 Poznan, Poland
complexityresearch@yahoo.com

Abstract. This study deals with time series analysis of blood flow velocity in the lobar branches of the renal artery of the kidney graft undergoing deterioration of function owing to acute rejection, acute tubular necrosis, acute pyelonephritis or chronic rejection. Both classical and Doppler sonography of kidney graft are not specific nor sensitive enough to distinguish between those clinical entities. Results of this pilot study indicate that the Hurst exponent of the time series of blood flow velocity changes significantly in the clinical course of acute cellular rejection or acute tubular necrosis. The Hurst exponent undergoes a unique temporal evolution indicating the emergence of deterministic chaos during the severe alteration of blood flow in kidney graft. This evolution can be described by the exponential function. Therefore, kidney graft with the underlying complex dynamics can be defined as a dynamic system of the first order.

Keywords: time series, fractal dimension, deterministic chaos, Hurst exponent, Doppler sonography, kidney, transplantation.

1 Introduction

Kidney transplantation is a method of choice in the treatment of patients with the end-stage renal disease. Indeed, survival after renal transplantation is significantly better than that of patients treated with dialysis [1]. Functionality of the graft may deteriorate, however, due to acute cellular rejection, acute tubular necrosis, pyelonephritis, urinary or vascular obstruction, immunosuppressive drug-induced nephrotoxicity, glucocorticoid-induced hyperglycemia or dehydration. It is important to differentiate between the causes of early graft dysfunction since some of them, e.g. cellular rejection can result in the loss of the organ. For example, kidney transplantation implies the necessity of suppression of the immune system.

Any infection may end-up under those circumstances in acute pyelonephritis in the graft or in life-threatening sepsis. Graft rejection, which is also accompanied by both fever and deterioration of renal function, may imitate acute pyelonephritis. Yet, treatment in both cases is different, i.e., antibiotics, antimycotic drugs, or antiviral drugs versus glucocorticoids or antibodies.

By definition, acute pyelonephritis denotes segmental, ascending inflammation of renal pelvis with formation of abscesses within the renal parenchyma due to infection. It occurs as the result of obstruction or reflux nephropathy. The activation of the biochemical inflammatory cascade results in the increased size of the graft due to swelling; a clinical hallmark of acute pyelonephritis. Yet, angiograms show normal anatomy of the renal vessels. They sometimes show stretching of interlobular arterial branches and the cortical striations [2]. Autoregulation of blood flow can however be preserved for a longer time. No change in the peak systolic velocity should occur unless vasoconstriction changes significantly the diameter of blood vessels and cuts blood flow off.

The clinical hallmark of acute rejection is an abrupt and significant decline in renal function. Graft rejection is driven by the circulating cytotoxic antibodies, activation of CD4+ lymphocytes and enhancement of IL-2 synthesis (reviewed in [3]). Although those events are different than the ones initiating pyelonephritis, they trigger the molecular response, which ends up in the activation of the same inflammatory cascade. The descending nature of the process results, however, in severe damage of both endothelium and interstitium. One of the earliest alterations is dilatation of the peritubular capillaries with mononuclear lymphoid cells. The endothelial cells swell and degenerate. This leads to renal ischemia. Infiltration of interstitium by lymphoid cells causes degeneration of tubular cells and results in swelling of the graft. It is worth to notice that glomerular changes in acute reversible rejection are not prominent. They are limited to swelling and hypertrophy of cells. The acute irreversible rejection is characterized by extensive fibrin thrombi in glomerular capillaries and their necrosis (reviewed in [4]). Due to the intraluminal thrombosis and necrosis blood flow can become turbulent. In addition, both the lower values of the peak systolic velocity and some significant changes in the autoregulation of blood flow can be expected.

The causes of early graft dysfunction may overlap. Of course, that makes any diagnostic approach particularly challenging. Although both clinical entities are determined by some distinct pathological mechanisms, there are no specific clinical, biochemical or biophysical criteria known which would allow to differentiate in a reliable manner between both clinical situations. A search for such the criterion continues.

Doppler sonography is one of the most frequently methods applied to appreciate both the kidney graft size and blood flow in and out of the graft. The superficial placement of a kidney graft in the iliac fossa makes sonographic imaging an attractive tool for the evaluation of this structure in a variety of pathological conditions. Most transplant patients with deteriorating renal function, as demonstrated by a rise in serum creatinine concentrations and/or decreased urine output, are referred for ultrasound evaluation as a first step towards a differential diagnosis.

This study has been undertaken to answer the following questions: 1. do time series obtained from a Doppler blood flow study possess fractal structure?, 2. how that structure changes during deterioration of renal function of the graft, 3. can one differentiate between the above-mentioned clinical entities on that basis?

2 Methods and Patients

2.1 Methods

Blood flow in the lobar branches of the renal artery of the graft was evaluated by means of the color-mode Doppler ultrasonography (Sonoline G50, the ultrasound C 5-2, 3.5 MHz, Siemens, Germany or Pro Focus UltraView, transducer 8830, 3-6 MHz, BK Medical, Denmark). The parameters, such as blood flow velocity (V), resistive index (RI), i.e., peak systolic velocity minus the end diastolic velocity divided by the peak systolic velocity, or pulsatile index (PI), i.e., a measure of the variability of blood velocity in a vessel, equal to the difference between the peak systolic and minimum diastolic velocities divided by the mean velocity during the cardiac cycle were recorded in the renal grafts of the patients with early graft dysfunction, chronic rejection or normal function. Blood flow was measured in a constant distance from the graft surface of about 2.5-3.0 cm. The time series obtained from those measurements have a total length of at least 100 impulses. Fractal analysis of the time series was performed using commercially available software (Benoit ver.1.3, TruSoft Int'l., Inc., USA). This software has been designed using the wavelet algorithm (see Appendix and [5] for the details).

Visual recurrence analysis and calculation of the correlation dimension was performed using VRA software ver 4.9 by Eugene Kononov. Correlation dimension can be applied to distinguish between chaotic and random behaviour of time series. The algorithm constructs a function $C(\epsilon)$ that represents a probability that two arbitrary points on the orbit of the attractor of the chaotic process are closer than ϵ . The correlation dimension is given by a formula $\log(C) / \log(\epsilon)$ in the limit $\epsilon \rightarrow 0$, and $N \rightarrow \infty$. The correlation dimension is defined as the slope of the curve $C(\epsilon)$ versus ϵ . In other words, $C(\epsilon)$ is the correlation of the data set, or the probability that any two points in the set are separated by a distance ϵ . A non-integer value of the correlation dimension indicates that the time series or dynamic process possesses most likely a fractal structure.

Both the mean value of the RI and the PI were calculated from the ten randomly chosen waveforms of each time series (Microsoft Exel 2000, Microsoft, USA).

Fitting of points and curves was performed using Sigma Plot version 10, Systat Software, USA.

In addition, morphology of the renal transplant arterial waveforms were recorded. The shape of the waveforms was categorized into six types according to criteria published elsewhere [6], [7] as shown in Figure 1.

The time interval between transplantation and sonography ranged from hours to years. Each patient was studied at least three times. The first measurement was

done on the day, in which acute symptoms or deterioration of renal function appeared. The additional measurements were performed during therapy to grasp any changes in blood flow. In particular, some additional blood flow measurements were performed every 24 hours in all cases of acute rejection in order to characterize dynamics of the process and its temporal evolution. Finally, blood flow was measured long after the end of treatment, when renal function had normalized and symptoms of early graft dysfunction had disappeared. The biochemical parameters such as leucocytes, creatinin, or CRP were measured by all patients at each Doppler sonography.

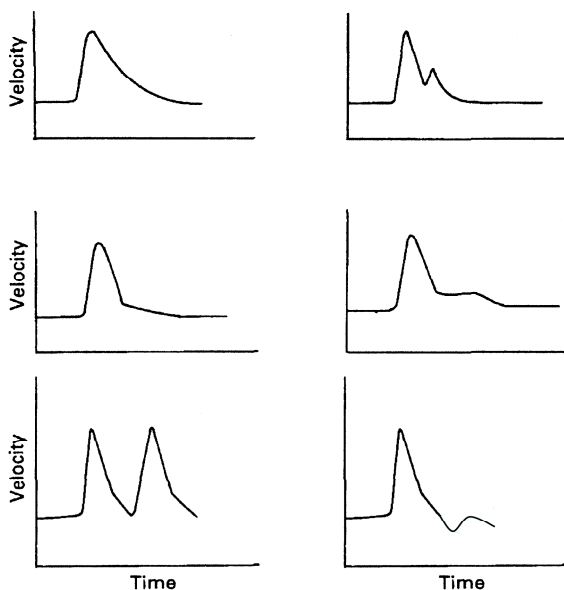


Fig. 1. Representation of the waveforms seen during the Doppler sonography of kidney graft according to [6], [7]. The left column from the top shows the waveform type 1, 3, and 5, respectively. The right column from the top shows the waveform type 2, 4, and 6. The waveform type 1 is typical of the normal kidney graft. The waveform 6 represents reversed diastolic flow, and is frequently seen in acute cellular rejection.

2.2 Patients

All patients participating in this study underwent kidney transplantation between 1996 to 2008 in the university transplantation centers in Halle, Erlangen or Munich and were treated in hospitals in Halle, Marktredwitz or Straubing, Germany.

Five of them had acute cellular rejection (ACR) at the moment of examination. Acute rejection was first diagnosed clinically on the ground of the following

criteria: 1. negative urine or blood cultures, 2. mild inflammation as determined by the lower initial values of C-reactive protein (CRP < 50 mg/l), 3. significant deterioration of graft's function, 4. rapid clinical response to treatment with immunosuppressive drugs, glucocorticoids or antibodies. The diagnosis was confirmed histologically by biopsy. All five cases were classified as acute cellular rejection of the grade II according to the Banff classification [8].

Five patients underwent chronic rejection (CR) with significant deterioration of graft's function as measured by the increased creatinine values greater than 350 $\mu\text{mol/l}$. Those patients responded relatively well to glucocorticoids with decline of creatinine concentrations to the average values about 150 $\mu\text{mol/l}$. In addition, the urine outcome improved. Neither had those patients acute symptoms, nor organ swelling typical of acute cellular rejection. No biopsy of the graft was performed.

Four patients were classified as acute pyelonephritis (APN) on the ground of the following criteria: 1. bacterial infection confirmed by urine culture, 2. large initial concentrations of C-reactive protein (CRP > 100 mg/l), 3. rapid clinical response to antibiotics and recovery of graft's function.

One patient underwent acute tubular necrosis (ATN) due to the prolonged ischemia time during transplantation, i.e., about 17 hours from the removal of the kidney from the cadaver donor to the reperfusion after transplantation. However, no biopsy was performed in that case.

One patient had high obstruction due to lymphocele, but no signs of rejection or pyelonephritis.

A control group contained 10 patients with good, stable function of kidney graft with creatinine concentration up to 60 $\mu\text{mol/l}$. The grafts worked well. Urine excretion was from 2 liter up to 7 liter per day. There were no signs of inflammation in the graft at the moment of examination.

3 Results

3.1 *Time Series Analysis of Blood Flow Velocity*

Table 1 summarizes results of time series analysis of blood flow in various clinical settings after kidney transplantation.

As expected, the increased values of RI and PI are associated with significant alterations of blood flow through the kidney vessels, such as those ones in the course of both acute cellular rejection and acute tubular necrosis. Those clinical entities are characterized by a significant decrease of the mean velocity of blood flow as compared with a group of patients with the normal grafts. The similar effect can be seen in the case of acute pyelonephritis. However, the mean velocity of blood flow is not so severely reduced as in the case of acute cellular rejection. In the case of ureter obstruction, RI and PI remain increased, but velocity of blood flow is significantly increased (compare Table 1). Owing to a low number of cases no statistical comparison between those groups was performed.

Table 1. Temporal evolution of the mean values of fractal dimension D , the Hurst exponent H , blood flow velocity V (cm/s) with the standard deviation σ , resistive index RI and pulsatile index PI in different clinical settings of kidney graft, where ACR stands for acute cellular rejection, ATN denotes acute tubular necrosis, CR denotes chronic rejection, and APN denotes acute pyelonephritis, and GNF stands for a graft with normal function.

Clinical entity	Day 0	Day 3	Day 5	Recovery
ACR n=5 patients	H=0.601	H=0.528	H=0.276	H=0.205
	D=1.399	D=1.472	D=1.724	D=1.795
	V=24.9	V=34.8	V=21.3	V=50.7
	σ =3.9	σ =7.7	σ =0.95	σ =9.5
	RI=0.95	RI=0.86	RI=1.02	RI=0.88
	PI=2.08	PI=1.88	PI=2.11	PI=1.71
ATN n=1 patient	H=0.378	H=0.423	H=0.521	H=0.219
	D=1.622	D=1.577	D=1.479	D=1.781
	V=27.4	V=14.1	V=19.4	V=32.2
	σ =3.5	σ =1.5	σ =3.2	σ =5.7
	RI=0.99	RI=0.87	RI=0.94	RI=0.81
	PI=1.91	PI=1.90	PI=1.79	PI=1.73
CR n=5 patients	H=0.176	H=0.244	H=0.277	H=0.273
	D=1.824	D=1.756	D=1.723	D=1.727
	V=23.8	V=22.5	V=28.3	V=29.7
	σ =2.5	σ =2.7	σ =3.0	σ =1.2
	RI=0.71	RI=0.73	RI=0.73	RI=0.72
	PI=1.81	PI=1.80	PI=1.78	PI=1.75
APN n=4 patients	H=0.404	H=0.273	H=0.278	H=0.232
	D=1.596	D=1.727	D=1.722	D=1.768
	V=36.1	V=35.7	V=36.9	V=42.1
	σ =5.8	σ =4.3	σ =5.1	σ =7.9
	RI=0.87	RI=0.92	RI=0.88	RI=0.81
	PI=2.10	PI=2.08	PI=1.91	PI=1.85
Obstruction n=1 patient	H=0.408	H=0.278	H=0.265	H=0.242
	D=1.592	D=1.722	D=1.735	D=1.758
	V=93.7	V=52.3	V=54.2	V=54.8
	σ =21.7	σ =10.2	σ =6.7	σ =4.5
	RI=0.85	RI=0.79	RI=0.72	RI=0.75
	PI=1.97	PI=1.78	PI=1.73	PI=1.71
GNF n=10 patients	H=0.224	H=0.213	H=0.218	H=0.216
	D=1.776	D=1.787	D=1.782	D=1.784
	V=55.4	V=57.3	V=56.1	V=56.7
	σ =3.9	σ =4.2	σ =4.8	σ =3.7
	RI=0.71	RI=0.69	RI=0.72	RI=0.73
	PI=1.58	PI=1.67	PI=1.60	PI=1.62

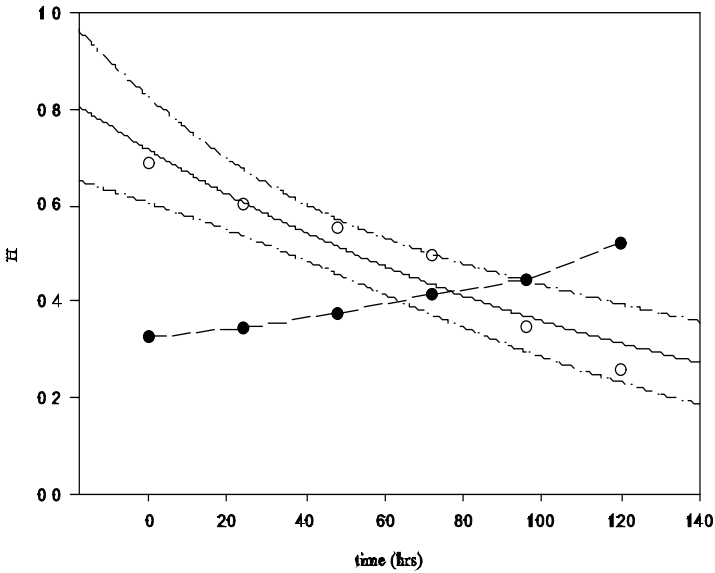


Fig. 2. Evolution of the Hurst exponent in time (hrs) representing time series obtained from a patient with acute tubular necrosis (black circles) or from a patient with acute cellular rejection (white circles). The dash-dot line indicates the 95% confidence band for the fitting with the function of exponential decay $f(t)=Ae^{-bt}$ ($R=0.963$, $A=0.714$, $b=0.007$; normality test, K-S test, and constant variance test passed, Durbin-Watson statistic failed). The process reflected by black circles was fitted with exponential function $f(t)=Ae^{bt}$ ($R=0.987$, $A=0.315$, $b=0.004$; normality test, Durbin-Watson statistic, K-S test, and constant variance test passed).

Either grafts with normal function or grafts in the course of chronic rejection possess much lower mean values of RI and PI. However, the mean value of blood flow velocity in those two groups varies about two-fold as compared with the cases of acute cellular rejection or acute tubular necrosis.

As shown in Table 1, a specific treatment of each clinical entity leads to some improvement of the mean values of the parameters eventually. The reaction to that treatment develops gradually as reflected by the temporal evolution of the fractal dimension or the Hurst exponent of the appropriate time series. In general, the mean values of the above parameters remain altered despite the successful treatment and recovery of the satisfactory kidney function as compared with the mean values of the same parameters in the grafts with the normal function (see Table 1).

Interestingly, the mean values of the Hurst exponent continue to increase during treatment in the case of acute tubular necrosis. On the contrary, the mean values of the Hurst exponent decrease quickly in the course of treatment in the case of acute rejection. Dynamics of both processes can be seen in the Figure 2. The appropriate points were fitted with a great accuracy with the exponential function reflecting exponential growth (black circles) and exponential decay (white circles) (see the details Fig. 2).

3.2 Temporal Evolution of the Hurst Exponent in Acute Tubular Necrosis or Acute Cellular Rejection

Temporal evolution of the Hurst exponent in acute cellular rejection is significantly different than in acute tubular necrosis. It can be easily seen from the Figure 2 that those processes possess different dynamics. More specifically, those processes are described well by a family of the exponential functions. The appropriate fitting was done with very large values of the coefficient of nonlinear regression $R > 0.95$. Therefore, both processes can be classified as the processes of the first order, i.e., the processes with dynamics described by the differential equations of the first order.

The correlation dimension is frequently used to distinguish between chaotic and random behaviour of time series. For the truly random signals, the correlation dimension graph will look like a 45-degree straight line, indicating that no matter how the noise is embedded, it will evenly fill the available space. Chaotic processes and the corresponding time series have a distinct spatial structure. Their correlation dimension will achieve a constant value at some point, as embedding dimension is increased. The values of the correlation dimension calculated for one of the patients with acute cellular rejection on day 0 and day 3 can be appreciated in the Figure 3.

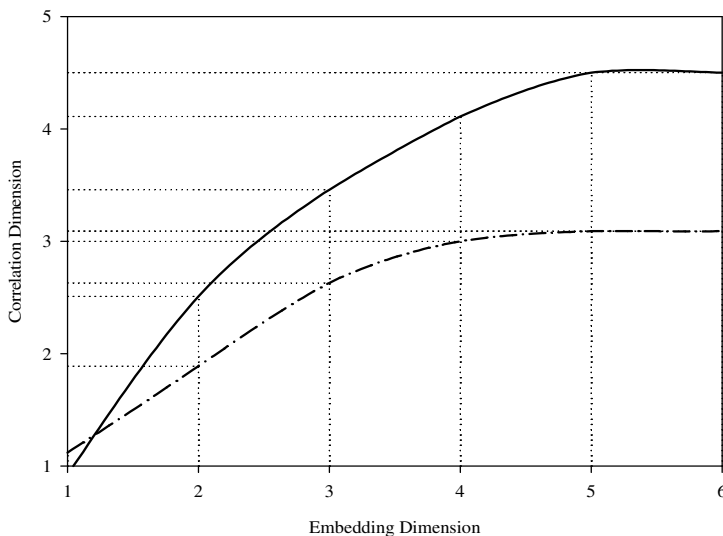


Fig. 3. The values of the correlation dimension of the time series for the patient with acute cellular rejection on day 0 (dash-dot line) and day 3 (solid line). The values of the correlation dimension saturate in both cases in a manner typical for a chaotic process with a fractal structure. The values were calculated for the first minimum of the mutual information (AMI) equal 3 as calculated by the VRA software by Eugene Kononov, ver. 4.9.

3.3 *Waveform Types and Their Frequency*

Table 1 summarizes frequency of waveforms seen during this study. The waveform Type 5, i.e., the waveform with high systolic peak was not seen at all during this study. The waveform Type 3 was seen only once in the course of pyelonephritis. The most frequently seen waveform was the waveform Type 1, i.e., the waveform with a gradual slope descent and continuous forward flow in diastole. This waveform appeared not only in the grafts with normal function, but also in all cases of acute rejection or pyelonephritis as well as in acute tubular necrosis. This waveform appeared also in majority of grafts with chronic rejection. The waveforms Type 2, i.e., the waveform with an early diastolic notch, waveform Type 4, i.e., the waveform with a rapid fall off in flow at end diastole and normal flow in early and mid diastole, and waveform Type 6, i.e., the waveform with reversed diastolic flow were seen in the cases of acute rejection with severe alterations of blood flow. Thus, there is no waveform typical exclusively for a given entity.

4 Discussion

4.1 *Temporal Evolution of Both the Hurst Exponent and Fractal Dimension*

There are no typical values of the parameters, such as H, D, V, RI or PI, which would indicate unequivocally the nature of the hemodynamic alterations in the kidney graft. However, those parameters change the mean values if alterations of blood flow become more severe. As shown in Table 1, alterations of blood flow end up in a significant decrease of the mean blood flow velocity. The only parameter that increases significantly during acute cellular rejection or acute tubular necrosis is the Hurst exponent (compare Table 1). This exponent is not only increased during the acute phase of blood flow alterations associated with both entities, but also seems to possess a specific temporal evolution in each of those clinical situations (see Fig. 2).

The low number of cases involved in that study does not allow more accurate definition of the mathematical model of those processes. It seems, however, that the observed evolution of the Hurst exponent can be described by the exponential function of time, and, therefore, the entire dynamic system represents a system of the first order, i.e., the system with dynamics described by the differential equation of the first order.

Acute cellular rejection is a fast process. Cytotoxic lymphocytes need a few hours to infiltrate the kidney graft. This is associated with a significant deterioration of its function including alterations of blood flow. Dynamics of that rapid phase remains unknown, and has not been analyzed in details during this study. However, it is worth to notice that dynamics of acute cellular rejection in the course of treatment is represented by the function of exponential decay (see

Fig. 2). Treatment of cellular rejection relies upon elimination of cytotoxic lymphocytes from the kidney graft owing to apoptosis. Since apoptosis of lymphocytes possesses the dynamics of exponential decay, it might be the underlying molecular cellular mechanism, which also determines the emergence of the observed dynamics of exponential decay in the evolution of the Hurst exponent.

The evolution of the Hurst exponent during acute tubular necrosis varies from the above discussed evolution in the course of treatment of acute cellular rejection. Yet, it is also described by the exponential function, i.e., the two-parametric function of exponential growth. This difference may result from a different pathological mechanism underlying acute tubular necrosis. In the latter case, tubular epithelial cells undergo irreversible damage owing to ischemia. This process is not so rapid as lymphocyte infiltration of the kidney graft and resulting acute cellular rejection. Hence, we observe relatively slow exponential evolution of the Hurst exponent. Owing to the design of this pilot study, it was not possible to analyze in details evolution of the Hurst exponent in the phase of recovery of the kidney graft from acute tubular necrosis, which takes some longer time.

The correlation dimension calculated by the VRA software for the patients with acute cellular rejection or with acute tubular necrosis supports the existence of the fractal structure in time series of blood flow in kidney graft. In addition, visual recurrence analysis enables graphical, qualitative evaluation of the structure of the corresponding time series, which varies significantly from the structure typical of the white noise or Brownian motion (data not shown).

This report indicates some additional benefits of the Doppler method, which have not been previously noticed. First, time series analysis of a single parameter, such as velocity of blood flow in the kidney graft reveals the existence of fractal structure in those series. Second, the temporal evolution of the Hurst exponent indicates an ability of the graft to self-regulation of blood flow; an indicator of the healthy functional status of the grafted kidney. This aspect is particularly interesting because the kidney graft does not possess any innervations, and blood vessels can only react to the hormonal signals. Third, acute alteration of that regulation seems to lead to deterministic chaos in time series of blood flow in the cases of severe anatomic changes, such as those occurring in the course of acute cellular rejection or acute tubular necrosis. Those findings are in concert with the previous findings from the study of blood flow in human brain. It has been reported that cerebral blood flow reveals multifractal structure. This structure undergoes significant changes when autoregulation of blood flow fails, e.g., during migraine [9].

4.2 Role of Ultrasonography in Evaluation of Kidney Graft

Although sonography allows some evaluation of the anatomic conditions in the kidney graft as well as the intensity of the inflammatory process, it is impossible to differentiate on that basis between the various clinical entities leading to alterations of blood flow. More specifically, Doppler sonography can help to appreciate severity of the alterations of blood flow in kidney graft. The patient

must however spend a long time motionless in the same position during recording of Doppler signals. Hence, recording of time series in transplant patients is even more challenging and time consuming procedure. This approach can be used in the clinic to appreciate severity of alterations of blood flow occasionally rather than systematically.

Results of this pilot study indicate that there are no specific waveforms, which would enable a correct diagnosis of the underlying pathological condition. Certain waveforms, such as the waveform Type 2 or the waveform Type 6 seem to be more specific for the conditions, in which severe alterations of blood flow occur. The larger clinical studies demonstrated findings similar to the findings of this study [7], [8]. Indeed, in severe acute rejection, the kidney transplant became more echogenic. Swelling was present, hypoechogenicity of the medullary pyramids was seen, the corticomedullary junction was indistinct, and the globular size of the kidney was also increased as in the case of acute pyelonephritis. The vascular resistance of the kidney was high as reflected by the mean value of RI and PI, and waveforms Type 6 with reversed diastolic flow were present.

5 Conclusions

Doppler sonography allows an accurate measurement of velocity of blood flow in a variety of clinical conditions in kidney graft. However, no specific waveforms corresponding to the particular clinical entity have been identified. Time series analysis of the velocity of blood flow reveals the existence of fractal structure in those series. This structure changes in the course of the severe alterations of blood flow in the graft, such as acute cellular rejection or acute tubular necrosis. The above outlined approach can be used to determine the severity of acute graft rejection or acute tubular necrosis. However, it is not appropriate for the differential diagnosis of pathological entities in the kidney graft. The accurate diagnosis can be made exclusively by a histological examination of the biopsy specimen obtained from the graft.

References

1. Schnuelle, P., Lorenz, D., Trede, M., Vanderwoude, F.J.: *J. Am. Soc. Nephrol.* 9, 2135 (1998)
2. Barth, K.H., Lightman, N.I., Ridolfi, R.L., Catalona, W.J.: *J. Urol.* 116, 650 (1976)
3. Barry, J.M.: In: Walsh, P.C. (ed.) *Campbell's Urology*, 8th edn., p. 345. Saunders, Philadelphia (2002)
4. Alexander, R.W., Richman, A.V.: In: Silverberg, S.G. (ed.) *Principle and Practice of Surgical Pathology*, p. 1091. Churchill Livingstone, New York (1988)
5. Kaplan, I.: *Estimating the Hurst exponent* (2003), http://www.bearcave.com/misl/misl_tech/wavelets/hurst/index.html
6. Renowden, S.A., Griffiths, D.F.R., Nair, S., Krishnan, H., Cochlin, D.L.: *Clinical Radiology* 46, 265 (1992)

7. Lockhart, M.E., Wells, C.G., Morgan, D.E., Fineberg, N.S., Robbin, M.L.: AJR 190, 650 (2008)
8. Solez, A., Axelson, R.A., Benediktsson, H.: Kidney Int. 44, 411 (1993)
9. West, B.J., Latka, M., Glaubic-Latka, M., Latka, D.: Physica A: Statistical Mechanics and its Applications 318, 453 (2003)
10. Peitgen, H.O., Jürgens, H., Saupe, D.: Chaos and Fractals. New Frontiers of Science, pp. 491–496. Springer, New York (1992)

Appendix: The Hurst Exponent and the Wavelet Method

The mathematical measure of self-affinity is the Hurst exponent, H . This exponent between 0 and 1 determines the roughness of any curve with fractal structure, i.e., the larger H the smoother is the curve. In other words, the exponent serves as a measure of bias in fractional Brownian motion. It is related to fractal dimension D with a simple formula $D=2 - H$.

Time series can be divided on that basis into three categories. The case $H = 0.5$ characterizes the ordinary Brownian motion. That motion has independent deviations of values in time. There is no correlation between them. They create a so-called random walk. Deviation of the value of the Hurst exponent from 0.5 denotes a fractional Brownian motion in the time series.

For $0.5 < H < 1.0$ deviations in the time series are persistent, i.e., $x(t+1)$ tends to deviate from the mean the same way $x(t)$ did, where t stands for scalar time; the probability that $x(t+1)$ deviates from the mean in the same direction as $x(t)$ increases as H approaches 1. There is a positive correlation between the consecutive deviations, i.e. increments or decrements in the time series. Such the time series reveals the existence of a long memory dynamic process behind, i.e., a process with a random component, where a past event has a decaying effect on future events [5]. Since Hurst exponent determines the rate of chaos, this value of the Hurst exponent denotes that dynamic system producing the given time series generates deterministic chaotic dynamics.

For $0 < H < 0.5$ deviations in the time series are anti-persistent, i.e., the probability that $x(t+1)$ deviates from the mean in the opposite direction from $x(t)$ increases as H approaches 0. There is a negative correlation between the increments of the consecutive values. The corresponding plot oscillates still with some regularity and order, which are totally missing in the first case of the random ordinary Brownian motion for $H = 0.5$ [10].

Wavelet analysis is a tool for analyzing localized variations in power by decomposing a trace into time frequency space to determine both the dominant modes of variability and how those modes vary in time. This method is appropriate for analysis of non-stationary traces, i.e. where the variance does not remain constant with increasing length of the data set. Fractal properties are present where the wavelet power spectrum is a power law function of frequency. The Wavelet method is based on the property that Wavelet transforms of the self-affine traces have self-affine properties.

Nonlinear Model Based Predictive Control of Visual Servoing Systems Using Image Moments

Cosmin Copot, Corneliu Lazar, and Adrian Burlacu

Department of Automatic Control and Applied Informatics,
"Gheorghe Asachi" Technical University of Iasi, Romania
{ccopot, clazar, aburlacu}@ac.tuiasi.ro

Abstract. Advanced techniques like image based predictive control are embedded now in visual servoing applications. As it can be seen in literature, the main visual features used in the nonlinear predictive servoing architecture are point features. In order to increase performances of the visual servoing systems, a model predictive controller based on image moments derived from point features is proposed in this paper. The new predictive control technique uses a local model based predictor for image moments and a reference trajectory in order to ensure a robust behavior. Based on the new predictive control strategy, a visual control architecture is designed and a simulator is developed. Simulation results are obtained for a visual sensor attached to a 6 d.o.f. robot manipulator in an eye-in-hand configuration. The results revealed high efficiency and robustness for the proposed approach.

Keywords: Visual servoing, predictive control, nonlinear system, image moments.

1 Introduction

During the last decades the research on merged domains, like robotics and control theory, generated multiple interesting results [1-4]. Image based visual servoing (IBVS) is an important technique used for solving the complex problems of controlling robot systems. An image based controller can be designed not only using a proportional control law, but also with advanced control techniques which imply the knowledge of an open-loop model of the visual servo control system. Predictive control strategies for visual servo control have already been proposed. Thus, in [5], an ARIMAX multivariable model is used to implement a GPC controller for high speed visual servoing of a robot manipulator. In [6], an IBVS structure based on nonlinear model predictive control (NMPC) is proposed. The visual features considered for this approach are point features while constraints regarding visibility and joint and torque limits are included. Another model of a visual servo system is given in [7] which employs a camera and a position controller with a robust disturbance observer in the joint space. In this way, each joint axis is decoupled and the inner velocity loop can be expressed in the frequency

region below the cut-off frequency of the robust disturbance observers as a diagonal transfer matrix.

An interesting method of visual feedback control is obtained when image based visual servoing and NMPC are combined. In [8], a predictive control architecture which integrates the reference trajectory and the image prediction based on a local model is presented. Traditionally, NMPC visual servoing is designed using point features. The predicted values of the point features over a prediction horizon can be computed using two types of model. The first one is based on a nonlinear global model for the image prediction and the second one used a local model based on interaction matrix to predict the point features regarding the camera velocity [6, 9]. Due to major disadvantages which emerge when using point features in visual servoing applications, image moments can be a solution [10]. For the first time, the authors proposed in 2010 an approach of NMPC visual servoing using image moments [11]. The proposed method proved to be more robust and stable as the ones which used point features.

In the present paper, an extension of the NMPC architecture from [11] is developed. Taking into account directly the image moments and the hypothesis that the interaction matrix related to it has slow changes around the desired camera pose [12], an one step ahead predictor was designed. In order to obtain a prediction over a larger horizon a recursive technique is applied to the one step ahead image moments predicted predictor. For better performances of the predictive control, a reference trajectory of the visual features must be considered. Thus, a method of generating reference trajectory for image moments from paths derived for point features is presented. The proposed architecture was implemented, tested and validated using a simulator developed in Matlab. The simulation results revealed good performances and showed how the dynamic behavior of the image based predictive controller is influenced by the reference trajectory.

The paper is structured as follows: in Section 2, an analytical method of computed interaction matrix related to image moments is presented. Section 3 is dedicated to the image moments predictive control algorithm. The experimental results are presented in section 4 and the conclusions are detailed in the last section.

2 Interaction Matrix Based on Image Moments

If an object in the image is described by a discrete set of n point features of coordinates (x, y) , then image moments m_{ij} of order $(i + j)$ are defined as:

$$m_{ij} = \sum_{k=1}^n x_k^i y_k^j. \quad (1)$$

The coordinates of the gravity center $x_g = \frac{m_{10}}{m_{00}}$, $y_g = \frac{m_{01}}{m_{00}}$, with $m_{00} = a = n$ the object area, are used to compute the centered moments μ_{ij} of order $(i + j)$ given by:

$$\mu_{ij} = \sum_{k=1}^n (x_k - x_g)^i (y_k - y_g)^j. \quad (2)$$

In visual servoing applications for eye-in-hand configuration, a set of image moments $f_m = [x_n, y_n, a_n, \tau, \xi, \alpha]^T$ derived from point features can be used to obtain an image-based control law [12]. The camera velocity $\mathbf{v}_c = [v, \omega]^T$ is composed from linear velocities $v = [v_x, v_y, v_z]^T$ and angular ones $\omega = [\omega_x, \omega_y, \omega_z]^T$. In order, to control the linear camera velocities, the following three components of image moments vector f_m can be considered [12]:

$$x_n = a_n x_g, \quad y_n = a_n y_g, \quad a_n = Z^* \sqrt{\frac{a^*}{a}}, \quad (3)$$

where Z^* represents the desired depth between the desired configuration of the point features and the camera and a^* is the desired object area. For a discrete object, the area a represents the number of points which describe the object and cannot be used as visual feature. Thus, in [12] is recommended to employ:

$$a = \mu_{20} + \mu_{02}. \quad (4)$$

The image moments used to control the angular camera velocities ω_x, ω_y and ω_z are τ, ξ and α defined by [12]:

$$\tau = \frac{I_{n1}}{I_{n3}}, \quad \xi = \frac{I_{n2}}{I_{n3}}, \quad \alpha = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right), \quad (5)$$

where:

$$\begin{aligned} I_{n1} &= (\mu_{50} + 2\mu_{32} + \mu_{14})^2 + (\mu_{05} + 2\mu_{23} + \mu_{41})^2 \\ I_{n2} &= (\mu_{50} - 2\mu_{32} - 3\mu_{14})^2 + (\mu_{05} - 2\mu_{23} - 3\mu_{41})^2 \\ I_{n3} &= (\mu_{50} - 10\mu_{32} + 5\mu_{14})^2 + (\mu_{05} - 10\mu_{23} + 5\mu_{41})^2. \end{aligned} \quad (6)$$

Having an object described by a set of point features, the link between the time variation of any point feature f and the camera velocity is given by [13]:

$$\dot{f} = L_f \mathbf{v}_c. \quad (7)$$

In (7) L_f is the interaction matrix having for one point $\mathbf{x} = (x, y)$ the expression:

$$L_x = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix}, \quad (8)$$

where Z is the point depth. Similarly with (7), the time variation of the image moments vector f_m can be computed:

$$\dot{f}_m = L_{f_m} \mathbf{v}_c. \quad (9)$$

Considering the centered image moments μ_{ij} as visual features, the time variation $\dot{\mu}_{ij}$ is obtained through derivation of (2):

$$\dot{\mu}_{ij} = \sum_{k=1}^n i(x_k - x_g)^{i-1} (y_k - y_g)^j (\dot{x}_k - \dot{x}_g) + j(x_k - x_g)^i (y_k - y_g)^{j-1} (\dot{y}_k - \dot{y}_g). \quad (10)$$

If planar objects are considered $\left(\frac{1}{Z} = Ax + By + C\right)$ [14] then the velocity of any object point (x_k, y_k) becomes:

$$\begin{cases} \dot{x}_k = -(Ax_k + By_k + C)v_x + x_k(Ax_k + By_k + C)v_z + x_k y_k \omega_x - (1+x_k^2)\omega_y + y_k \omega_z \\ \dot{y}_k = -(Ax_k + By_k + C)v_y + y_k(Ax_k + By_k + C)v_z + (1+y_k^2)\omega_x - x_k y_k \omega_y - x_k \omega_z \end{cases}. \quad (11)$$

Using (11) together with (10), the interaction matrix of the centered moments μ_{ij} is obtained:

$$L_{\mu_{ij}} = [\mu_{vx}; \mu_{vy}; \mu_{vz}; \mu_{\omega x}; \mu_{\omega y}; \mu_{\omega z}]. \quad (12)$$

The analytical form of the parameters $\mu_{vx}, \mu_{vy}, \mu_{vz}, \mu_{\omega x}, \mu_{\omega y}, \mu_{\omega z}$ is detailed in [14]. Based on (12), the interaction matrix related to the image moment features vector $f_m = [x_n, y_n, a_n, \tau, \xi, \alpha]^T$ can be computed using:

$$L_{f_m} = \begin{bmatrix} -1 & 0 & 0 & a_n e_{11} & -a_n(1+e_{12}) & y_n \\ 0 & -1 & 0 & a_n(1+e_{21}) & -a_n e_{11} & -x_n \\ 0 & 0 & -1 & -e_{31} & e_{32} & 0 \\ 0 & 0 & 0 & \tau_{\omega x} & \tau_{\omega y} & 0 \\ 0 & 0 & 0 & \xi_{\omega x} & \xi_{\omega y} & 0 \\ 0 & 0 & 0 & \alpha_{\omega x} & \alpha_{\omega y} & -1 \end{bmatrix}. \quad (13)$$

The analytical form of parameters from (13), $e_{11}, e_{12}, e_{21}, e_{31}, e_{32}$, $\tau_{\omega x}, \tau_{\omega y}, \xi_{\omega x}, \xi_{\omega y}, \alpha_{\omega x}, \alpha_{\omega y}$ can be found in [12].

3 Visual Predictive Control Based on Image Moments

The design of the classical IBVS structures is based on a proportional controller in order to generate a visual servoing control law. Even if the classical proportional approach to IBVS has a straightforward implementation, its major drawbacks conducted to the development of advanced control techniques for visual servoing systems. Starting from the local model predictor proposed by the authors in [11], a new visual predictive strategy based on image moments is developed. Any predictive control method is composed of three major components: a model based predictor, a reference trajectory and a cost function, each of these components will be detailed in the following.

3.1 Visual Servoing Architecture

The image moments based visual predictive control architecture proposed is presented in Fig. 1. In order to control a 6 d.o.f. robot manipulator with an eye-in-hand camera configuration for grasping a fixed object, an image based predictive controller (IbPC) is considered. A reference trajectory is used to define the behavior of controlled output predictions in going from the current features $f_p(k)$ to the desired features f_p^* considered as set point over the prediction horizon hp . The image moment features vector f_m is computed from the point features ensemble using a transformation denoted \mathbb{M} which implements equations (1) to (6).

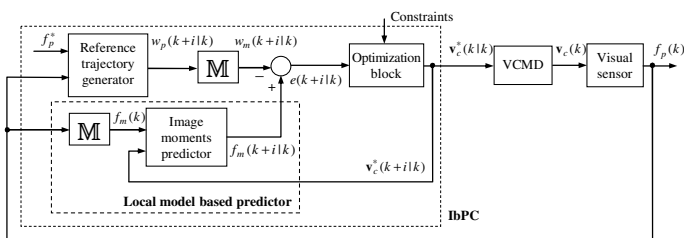


Fig. 1. Image based visual servoing

Having the image center coordinates (u_0, v_0) and the intrinsic camera parameter (p_x, p_y) , the point feature $\mathbf{x} = (x, y)$ expressed in pixels is:

$$f_p = (u_i, v_i) \text{ with } u_i = \frac{x_i}{Z} p_x + u_0, \quad v_i = \frac{y_i}{Z} p_y + v_0. \tag{14}$$

The manipulator is modeled as a Virtual Cartesian Motion Device (VCMD), which has as input the reference of camera velocity \mathbf{v}_c^* and as output the camera

velocity \mathbf{v}_c . The camera velocity should be controlled in the camera coordinates to obtain a linear diagonal system of a 6 d.o.f. robot manipulator, thus:

$$\mathbf{v}_c(s) = G(s)\mathbf{v}_c^*(s). \quad (15)$$

One way to achieve a decoupled system is to employ a robust control strategy based on the joint space disturbance observer (DOB) [7] and thus, each joint axis is considered decoupled under the cut-off frequency of DOB. Assuming that the velocity controller is designed as a diagonal matrix $\mathbf{k}_v = \text{diag}\{k_v, \dots, k_v\}$, the following linear discrete model of the VCMD system is obtained:

$$G(z^{-1}) = (1 - z^{-1}) \mathbb{Z}(G(s)/s); \quad G(s) = \frac{k_v}{s + k_v} I_6, \quad (16)$$

where k_v is a proportional gain controller and \mathbb{Z} represents the z transform. The output of the VCMD, camera velocity \mathbf{v}_c , is sampled with the sampling period T , resulting $\mathbf{v}_c(k)$ which is used as input in Visual Sensor block (Fig. 2) in order to compute the new camera pose.

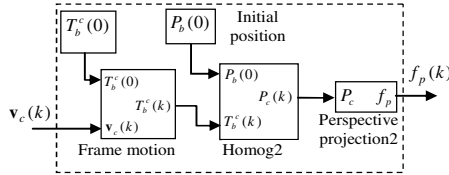


Fig. 2. Visual sensor

Using screw velocity $\mathbf{v}_c(k)$ and knowing the camera position related to the robot $T_c^b(0)$, at each iteration, the 'Frame motion' block computes the new camera pose $T_c^b(k)$ [15]. Assuming that, the starting object features position related to the robot base $P_b(0)$ are known and using the camera pose stored in $T_c^b(k)$, the current features coordinates $f_p(k)$ are obtained.

3.2 Local Model Based Predictor

Based on the assumption that the interaction matrix related to image moments has slow changes around the desired camera pose [12], a new local model based predictor was designed using directly the image moments features (Fig. 3).

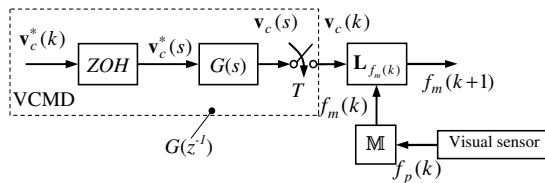


Fig. 3. Local model for image prediction

Having $\mathbf{v}_c(k)$ as output of the VCMD discrete time model, the discrete time relation between camera and image moments velocities is obtained by discretization of (9) using Euler's method:

$$f_m(k+1) = f_m(k) + TL_{f_m}(k)\mathbf{v}_c(k). \tag{17}$$

In (17) $L_{f_m}(k)$ is the interaction matrix related to a set of image moments f_m derived from point features acquired at the current discrete time k with a camera and an appropriate feature point detector. The one-step ahead prediction of the image moments evolution, when the plant model is considered, can be computed using (17) and the discrete model (16) of the VCMD, resulting:

$$f_m(k+1|k) = f_m(k) + TL_{f_m}(k)G(z^{-1})\mathbf{v}_c^*(k|k), \tag{18}$$

where the notation $f_m(k+1|k)$ indicates that the prediction is computed at the discrete time k .

For the design of an image moments based predictive controller, a multi step predictor is necessary. Shifting the one-step ahead prediction model (18) by recursion, the next predictors over prediction horizon hp are obtained:

$$\begin{aligned} f_m(k+2|k) &= f_m(k+1|k) + TL_{f_m}(k)G(z^{-1})\mathbf{v}_c^*(k+1|k) \\ &\dots\dots\dots \\ f_m(k+i|k) &= f_m(k+i-1|k) + TL_{f_m}(k)G(z^{-1})\mathbf{v}_c^*(k+i-1|k) \\ &\dots\dots\dots \\ f_m(k+hp|k) &= f_m(k+hp-1|k) + TL_{f_m}(k)G(z^{-1})\mathbf{v}_c^*(k+hp-1|k), \end{aligned} \tag{19}$$

and thus, the i -step ahead predictor $f_m(k+i|k)$ can be computed using:

$$f_m(k+i|k) = f_m(k) + TL_{f_m}(k)G(z^{-1})\sum_{j=1}^{i-1} \mathbf{v}_c^*(k+j|k). \tag{20}$$

The prediction is initiated with the image moments $f_m(k)$ derived from point features, at discrete time k , that are obtained from an acquired image with a suitable feature point detector.

3.3 Reference Trajectory Generator

In predictive control theory, the reference trajectory usually starts at the current plant output and defines an ideal trajectory along which the plant output should go to the set point over prediction horizon and the dynamics of the closed loop. In this paper a reference trajectory for feature points is designed and transformed into a reference trajectory of image moments. Such a reference trajectory for visual servoing systems is presented in (Fig. 4). Beginning at the current discrete time k with the current image I_k having the point features $f_p(k)$, a reference trajectory is designed from the image sequences $I_{k+i}, i = \overline{1, hp}$ with point features $w_p(k+i|k)$ in order to obtain $w_p(k+hp|k) = f_p^*$. The notation $w_p(k+i|k)$ specifies that the reference trajectory depends on the initial conditions $f_p(k)$ at discrete time k .

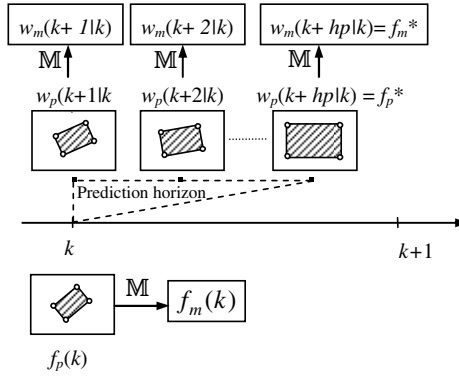


Fig. 4. Reference trajectory over hp

In the following, a method for deriving a reference trajectory at step k using the path planning approach from [16] is detailed. Taking into account the collineation matrix \mathbf{G} , representing the projective homography between the initial image I_k and the final image I_{k+hp} , the homogeneous coordinates of the four point features from the final image $\tilde{f}_j^* = [u_j^*, v_j^*, 1]^T$ can be expressed with respect to the coordinates of points from the initial image $\tilde{f}_j(k) = [u_j(k), v_j(k), 1]^T$, resulting:

$$\tilde{f}_j^* = \mathbf{G}\tilde{f}_j(k), j = \overline{1, 4}. \quad (21)$$

In order to obtain a path between the current point features positions $f_p(k)$ at time $t = kT = 0$ and the desired one f_p^* at time $t = (k+hp)T = t_{hp}$, it is neces-

sary to build a sequence of collineation matrices that will be correlated with a time variation law. The sequence can be obtained using the parameter-dependent matrix:

$$\mathbf{G}_d(q) = \mathbf{K}\mathbf{H}_d(q)\mathbf{K}^{-1}, \quad (22)$$

where $q = q(\sigma)$ is a monotonic function of the non dimensional time $\sigma = t/t_{hp}$ ranging from $q(0)$ at $t = kT = 0$ to $q(1)$ at $t = (k + hp)T = t_{hp}$, \mathbf{K} is the matrix of the intrinsic camera parameters and \mathbf{H} represents the Euclidean homography with $\mathbf{H}_d(q(0)) = \mathbf{I}$, $\mathbf{H}_d(q(1)) = \mathbf{H}$.

For a smooth q function a quintic-polynomial can be used:

$$q(\sigma) = a_5\sigma^5 + a_4\sigma^4 + a_3\sigma^3 + a_2\sigma^2 + a_1\sigma^1 + a_0. \quad (23)$$

The coefficients $a_i, i = \overline{0,5}$ can be found as the solution of the following linear system:

$$\begin{cases} q(0) = 0; & \dot{q}(0) = \psi; & \ddot{q}(0) = 0 \\ q(t_{hp}) = 1; & \dot{q}(t_{hp}) = 0; & \ddot{q}(t_{hp}) = 0. \end{cases} \quad (24)$$

The tuning of the ψ parameter will generate different behaviors of the time function q . The geometrical interpretation of ψ is the tangent of the angle under which the time function q starts. Thus, if the value of ψ is increased then, a faster response for q is obtained, this implies that ψ is a parameter that will influence the dynamic of the resulting path.

Starting from the initial conditions $f_p(k)$, the reference trajectory is generated:

$$w_p(k+i|k) = \mathbf{G}_d(q)\tilde{f}_j(k), i = \overline{1, hp}, j = \overline{1, 4}, \quad (25)$$

in order to obtain $w_p(k+hp|k) = f_p^*$ at the end of the prediction horizon. As in [16], a helicoidal shape of the reference trajectory will be considered. Using the transformation \mathbb{M} , the point features reference trajectory $w_p(k+i|k)$ is converted to image moments reference trajectory $w_m(k+i|k)$, thus ensuring that $w_m(k+hp|k) = f_m^*$.

3.4 Cost Function

The general aim of the optimization block is to make predicted system outputs to converge for a desired reference trajectory $w_m(k+i|k), i = \overline{1, hp}$. For that the cost function J is established, generally defined as a quadratic function of predicted control error and control. The error in image space over the prediction horizon hp is given by:

$$e(k+i|k) = f_m(k+i|k) - w_m(k+i|k), i = \overline{1, hp}. \quad (26)$$

The cost function is defined by:

$$J = \frac{1}{2} \sum_{i=1}^{hp} e^T(k+i|k) \mathbf{Q} e(k+i|k) + \sum_{i=0}^{hu-1} \mathbf{v}_c^{*T}(k+i|k) \mathbf{W} \mathbf{v}_c^*(k+i|k), \quad (27)$$

\mathbf{Q} and \mathbf{W} denote positive definite, symmetric weighting matrices and hu is the control horizon in (27). The main constraints are associated to the limits of the image called the visibility constraint, ensuring that all the features are always visible:

$$(u_j(k), v_j(k)) \in [(u_{\min}, v_{\min}), (u_{\max}, v_{\max})], i = \overline{1, m}, \quad (28)$$

Other constraints related to the robot and frequently used, are the torque constraints, the joint boundaries and camera velocity constraints.

4 Simulation Results

In this section, a simulator of the proposed visual control architecture was developed. A high number of experiments were conducted and the obtained results were analyzed.

4.1 Visual Control Architecture

The proposed predictive structure from Fig. 1 was implemented and a simulator was developed. This simulator illustrated in Fig. 5 was constructed starting from an existing toolbox proposed in [15]. From this existing Cervera's toolbox two entities were used: the Desired visual features and the Visual Sensor blocks and other two were developed: IbPC and VCMD.

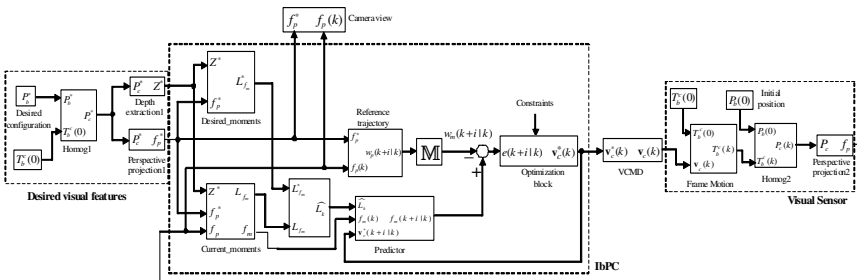


Fig. 5. Predictive control architecture for visual servoing

Planar objects are defined using points in Cartesian space. The blocks 'Initial position' and 'Desired configuration' are used to represent the start position $P_b(0)$ and the desired position P_b^* of the object. Considering the frames attached to the

robot base F_b , to the camera F_c and to the object F_o , the homogeneous matrices T_c^b and T_o^b between the frames F_c and F_b and, respectively F_o and F_b are given. Knowing the position of the desired points related to the robot P_b^* and the camera position related to the robot $T_c^b(0)$, the points position P_c^* , can be detected relatively to the camera coordinate system F_c by using a homogeneous transformation implemented with 'Homog1' block. By a similar procedure, the initial position $P_b(0)$, respectively the current positions of the points are transposed from F_b to F_c frames using 'Homog2' block, resulting the initial/current position of the object points in the camera frame. The image of the object described by points given in Cartesian space is built using 'Perspective projection' blocks (Fig. 5).

The image moments selected to control a manipulator robot, $f_m = [x_n, y_n, a_n, \tau, \xi, \alpha]^T$, are derived from point features and are computed using 'Current_moments' and 'Desired_moments' blocks. Image moments are calculated based on (1)-(6). The depth Z^* , necessary for image moments computation, is given by the 'Depth extraction' block. For visualization of the object points in the image plane, the 'Camera view' block from Fig. 5 is employed.

Using the desired image moments f_m^* and current ones $f_m(k)$, the interaction matrices $L_{f_m}^*$ and $L_{f_m}(k)$ are computed employing (13). For better performances of the predictor (19), the interaction matrix $L_{f_m}(k)$ can be replaced by $\hat{L}_k = 1/2(L_{f_m}^* + L_{f_m}(k))$. The 'Reference trajectory' block implements the approach presented in Subsection 3.3 to generate the reference trajectory. The cost function (27) with the constraints (28) is minimized by 'Optimization block' using the Matlab function *fmincon*.

4.2 Simulations

The visual predictive control strategy based on image moments proposed in the present paper was implemented, tested and validated. Considering a planar object defined by 4 points in Cartesian space, the proposed control architecture from Fig. 5 was implemented in Matlab, and the conducted experiments revealed. Considering a visual servoing task configuration (Fig. 6, 7), an image-based predictive controller was designed in order to minimize the cost function defined by (27).

Assuming that the prediction model contains the VCMD model which is designed as a diagonal matrix with equal value, $k_v = 160$, the sampling period T

must be at least $\frac{4}{k_v}$. All the simulation results from this paper were done

considering $T = 1[\text{sec}]$. The intrinsic camera parameters $p_x = p_y = 1000$ were chosen.

To evaluate the performances of the proposed image based predictive approach based on image moments and a reference trajectory, Matlab simulations were conducted. The image plane feature points trajectory is depicted in (Fig. 7(a), 8(a), 9(a)) with stars. The desired configuration is illustrated with red squares, while the starting configuration is represented with blue circles. For the prediction of the image moments with (19) derived from point features, the interaction matrix $L_{f_m(k)}$ of the current configuration is considered over a prediction horizon. The parameters of the predictive controller are set on $hp = 4$ and $hu = 1$. In order to ensure a better tracking of the desired reference trajectory the weighting matrixes $\mathbf{Q} = e^{-i} \mathbf{I}_6, i = \overline{1, hp}$ and $\mathbf{W} = \mathbf{I}_6$ were chosen. The following experiments show how the distribution of the time function (24) will influence the dynamic behavior of the image based predictive controller. Each value chosen for the parameter ψ , which represent the slope under which the time function leaves the origin, generates a different distribution of the reference trajectory in the image plane (Fig. 6).

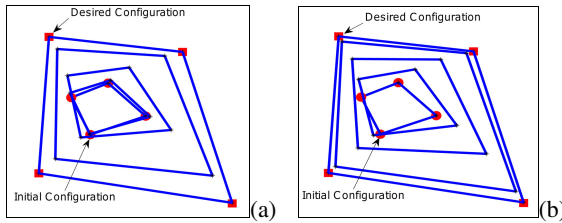


Fig. 6. Reference trajectory for : (a) $\psi = 0$; (b) $\psi = \text{tg}(\pi/4)$

In the first experiment the parameter ψ is set on 0. The point features trajectory is depicted in (Fig. 7(a)) and the camera velocities are illustrated in (Fig. 7(b)). Increasing the values of ψ ($\psi = \tan(\pi/12)$, $\psi = \tan(\pi/4)$) will result in a higher density of point features near the desired configuration. This will ensure a faster convergence of the predictive controller (Fig. 8(b), 9(b)). The results are also transposed into the trajectory of the point features in the image plane (Fig. 8(a), 9(a)). Analyzing the evolution of the camera velocities components (Fig. 7(b), 8(b), 9(b)) it can be observed that a large value of the parameter ψ , will imply a faster convergence of the system. An important issue is that when increasing the parameter ψ , an increase over the control effort is obtained. Thus, it is necessary to establish a compromising value for the ψ parameter in order to obtain good performances without great effort.

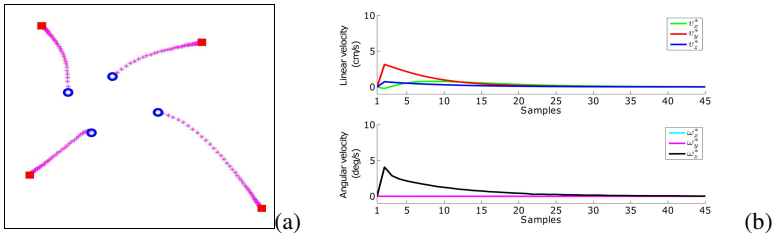


Fig. 7. (a) Image plane feature points trajectory and (b) camera velocities when $\psi = 0$

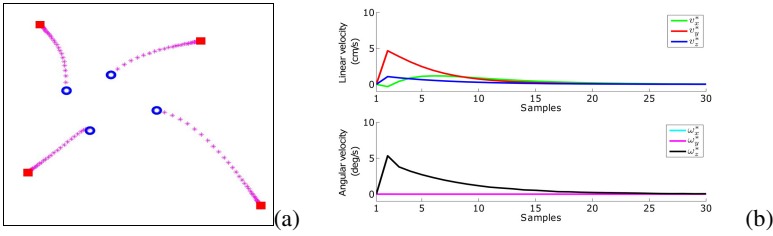


Fig. 8. (a) Image plane feature points trajectory and (b) camera velocities when $\psi = tg(\pi/12)$

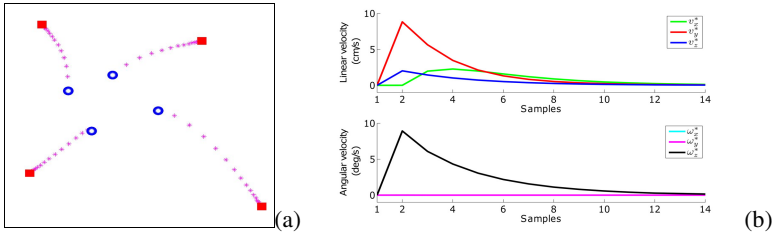


Fig. 9. (a) Image plane feature points trajectory and (b) camera velocity when $\psi = tg(\pi/4)$

5 Conclusion

In this paper a new image moments predictive control architecture applied to visual servoing has been proposed. The image based predictive controller designed uses a local model based predictor for image moments and a reference trajectory generator. A new parameter was proposed and its main advantage is that it can be used to control the dynamics of the visual servoing system. This parameter is part of the reference trajectory and can be tuned as so the behavior of the considered system is modified according to a known time function. Based on the proposed architecture, a simulator was constructed and simulations for a 6 d.o.f. with eye-in-hand configuration were conducted. The simulation results showed how the dynamic behavior of the predictive controller is influenced by the distribution of the time function.

Acknowledgments. This paper was supported by the project PERFORM-ERA "Postdoctoral Performance for Integration in the European Research Area" (ID-57649), financed by the European Social Fund and the Romanian Government.

References

1. Cherubini, A., Chaumette, F., Oriolo, G.: Visual servoing for path reaching with non-holonomic robots. *Robotica* 29(7), 1037–1048 (2011)
2. Cavalcanti, A., Shirinzadeh, B., Kretly, L.: Medical nanorobotics for diabetes control. *Nanotechnology, Biology and Medicine* 4(2), 127–138 (2008)
3. Manoiu-Olaru, S., Nitulescu, M.: Stability Analysis Software Platform Dedicated for a Hexapod Robot. In: Proc. of 18th International Conference on Control Systems and Computer Science, pp. 385–390 (2011)
4. Filipescu, A., Minzu, V., Minca, E., Filipescu Jr., A.: Path Following Fuzzy Control and Bubble Rebound Obstacle Avoidance Method of a WMR Mobile Platform. In: Proc. of 18th International Conference on Control Systems and Computer Science, pp. 404–409 (2011)
5. Gangloff, J.A., De Mathelin, M.F.: High speed visual servoing of a 6 dof manipulator using multivariable predictive control. *Advances Robotics* 21(10), 993–1021 (2003)
6. Allibert, G., Courtial, E., Chaumette, F.: Visual servoing via nonlinear predictive control. In: *Visual Servoing via Advan. Numer. Methods*, pp. 375–393. Springer (2010)
7. Fujimoto, H.: Visual servoing of 6 dof manipulator by multirate control with depth identification. In: Proc. of 42nd IEEE Conference on Decision and Control, Hawaii, pp. 5408–5413 (2003)
8. Lazar, C., Burlacu, A., Copot, C.: Predictive Control Architecture for Visual Servoing of Robot Manipulators. In: Proc. of the 18th IFAC World Congress, Milano, pp. 9464–9469 (2011)
9. Lazar, C., Burlacu, A.: Predictive control strategy for image based visual servoing of robot manipulators. In: Proc. of 9th International Conference on Automation and Information, Bucharest, pp. 91–97 (2008)
10. Chaumette, F., Hutchinson, S.: Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine* 13(4), 82–90 (2006)
11. Copot, C., Burlacu, A., Lazar, C.: An image moment based approach for visual predictive control. In: Proc. of 14th Int. Conference on System Theory and Control (ICSTC), Sinaia, pp. 154–159 (2010)
12. Tahri, O., Chaumette, F.: Point-based and region based image moments for visual servoing of planar objects. *IEEE Transaction on Robotics* 21(6), 1116–1127 (2005)
13. Chaumette, F., Hutchinson, S.: Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine* 14(1), 109–118 (2007)
14. Chaumette, F.: Image moments: A general and useful set of features for visual servoing. *IEEE Transaction on Robotics* 20(4), 713–723 (2004)
15. Cervera, E.: Visual servoing toolbox. Jaume I University (2003), <http://vstoolbox.sourceforge.net/>
16. Allotta, B., Fioravanti, D.: 3D motion planning for image-based visual servoing tasks. In: Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 2173–2178 (2005)

Mobile Robot Navigation Using Graph Search Techniques over an Approximate Cell Decomposition of the Free Space

Radu Robotin, Gheorghe Lazea, and Petru Dobra

Technical University of Cluj-Napoca, Faculty of Automation and Computer Science,
Department of Automation,
26-28 Gh Baritiu str., 400027 Cluj-Napoca, Romania
{Radu.Robotin, Gheorghe.Lazea, Petru.Dobra}@aut.utcluj.ro

Abstract. Mobile robots often operate in domains that are incompletely known. This article addresses the goal-directed navigation problem in unknown terrain where a mobile robot has to move from its current configuration to given goal configuration. We will present tests performed with various implementations of graph search algorithms (A^* , D^* , focused D^*) as path planners for a mobile robot, focusing on the inherent strong points and drawbacks of each implementation.

Keywords: mobile robot, graph search, path planning, heuristic search, comparative tests.

1 Introduction

In a number of applications, the problem of determining the optimum path occurs. This applications range from finding the fastest path in a network, to determining the safest path for mobile vehicle or wandering on the surface of Mars. In this context, we shall limit our scope to the case of finding paths in Euclidean two-dimensional space. Mobile robots often operate in domains that are only incompletely known. This article presents the goal-directed navigation problem in unknown terrain where a mobile robot has to move from its current configuration to given goal configuration. Robotics researchers have investigated various navigation strategies to solve it, including the well-known bug algorithms.

The idea behind the approach is that the robot always plans a shortest path from its current coordinates to the goal coordinates under the assumption that unknown terrain is traversable. Moreover, the robot may use the initial information on the environment if available. If it observes obstacles as it follows this path, it inserts them into its map and then repeats the procedure, until it eventually reaches the goal configuration or it cannot find any traversable path. This navigation strategy is an example of sensor-based motion planning. According to Berg, if we model the navigation problem as a graph-traversal problem on an eight-connected grid with edges that are either traversable or un-traversable, it must terminate because

the robot either follows the planned path to the goal vertex or increases its knowledge about the true edge costs, which can happen only once for each edge. To be specific, we shall look at the case of finding the optimum path for a mobile robot moving along a flat surface, the robot's configurations in the configuration space being the graph's nodes whereas the graph's arcs represent the cost of moving from one configuration to another.

Researchers have tried to come with new and better navigation technologies in the last years. With the development of path finding, several new classical routing algorithms have been introduced to generate better routing solution. For example, Eklund et al., use the Dijkstra algorithm as a path planner, one of the most famous routing algorithms, which evaluates the moving cost from one node to any other node and sets the shortest moving cost as the connecting cost of two nodes. Around the same period of time, Best-First-Search algorithm is also introduced in the researchers' community.

LaValle presents a solution for a path-planner a little different from the Dijkstra algorithm: the Best-First-search algorithm which has a different approach because it estimates the distance from current position to goal position, and it chooses the step that is closer to the goal position. The difficulty was growing with the new path finding situations so the old path finding algorithm had to be improved to meet the new introduced requirements. In the artificial intelligence community, around late 70's, a new path finding algorithm was introduced and it was named the A* algorithm. The A* algorithm tries to combine the advantages offered by Dijkstra algorithm and Best-First-Search algorithm.

This paper presents tests performed with various implementations of graph search algorithms (A*, D*, focused D*) as path planners for a mobile robot, focused on strong points and drawbacks of each search algorithm, from the point of view of the path planner's efficiency.

2 Path planners for Mobile Robots with Approximate Cell Decomposition

2.1 General Graph Search

The search process in a graph can be seen as applying a set of operators to the graph's nodes until the goal node is found. The process usually starts in the goal node and then moves to the successors of the node.

The above procedure does not have any mention about the order in which the successors of a node should be selected for further explorations. The way a node, n , is selected for exploration, determines the overall behavior of the search algorithm and resulting path to the goal. For example, if the nodes are selected for expanding in the order in which they are generated, the search is performed in a "breadth-first" fashion. On the other hand, if the most recent generated successor is selected for expanding, then the algorithm performs a "depth-first" search. These types of search are not influenced by either the selection criteria of the successors of a node or by the position of the goal node in the searched graph, as they

perform a blind search. To implement the navigation strategy, the robot needs to re-plan a shortest path from its current vertex to the goal vertex whenever it detects that its current path is un-traversable. Brummitt and Stentz, suggested the robot could use conventional graph-search methods but this is inefficient since most edge costs do not change between re-planning episodes.

From the planner's point of view, beside the search strategy, the representation of the robot's free space is of equal importance. Bonet and Geffner, showed that many methods for discretizing continuous terrain have been investigated in computer science, all of which attempt to balance the inherent tradeoff between two conflicting criteria, namely the path planning runtime and the length of the resulting path. Visibility graphs contain the start vertex, the goal vertex and the corners of all blocked cells. The shortest paths on visibility graphs are also shortest paths in the continuous terrain but path planning is slow on large visibility graphs since the number of edges can be quadratic in the number of cells and the runtime complexity of the search algorithm remains linear with the number of cells. On the other hand, Koenig and Likhachev, showed that path planning is faster on grids than visibility graphs, since the number of edges is linear in the number of cells. However, paths formed by grid edges can be sub-optimal and unrealistic looking since the possible headings are artificially constrained.

2.2 The A* Algorithm

The A* algorithm was proposed by Nilson, refined by Buckland and Goldberg et al., to name but a few. It uses a specific evaluation function that, it can be proven, minimizes the number of visited nodes during search. The algorithm returns the minimum cost path between the start node and the goal node. The evaluation function, \hat{f} , is defined in such a fashion so that its value, $\hat{f}(n)$, for any node, n , is an estimate of the minimum cost path passing through n . This estimate is computed as a sum between an estimate of the minimum cost path from the start node to node n , and the estimate of the minimum cost path from node n to goal:

$$f(n) = g(n) + h(n) \quad (1)$$

It is necessary the evaluation function, \hat{f} , to be an estimate of the function f , so that, let \hat{g} be an estimate of g and \hat{h} be an estimate of h . The evaluation function is:

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \quad (2)$$

The value of $\hat{g}(n)$ can be easily computed by adding the arc costs on the path from the start node, s , to node n . Finding an expression for $\hat{h}(n)$ is not an easy task. Information contained in the graph must be used along with the proper choice of metric for measuring distances. If $\hat{h}(n)$ is an optimistic estimate of

$h(n)$, ($\hat{h}(n) \leq h(n)$), then A^* will find the minimum cost path and the algorithm is admissible (it always finds the minimum cost path from the start node to the goal node).

2.3 The D^* Algorithm

If a planner is based on A^* , the affected nodes and corresponding arcs must be updated in the graph that is used for storing the map, before the search and navigation process continues. The approach can be rather inefficient, especially when the information in the map does not reflect the reality or in the case when states change during the mobile robot navigation or when dealing with incomplete information. Such an approach is based on the following scenario: every time a discrepancy between the data in the map and the data provided by the sensors on-board the mobile robot is found, the planner updates the map followed by a new planning process.

The inefficiency of the approach is visible especially in the case when the robot is close to the goal state or when large portions of the map have to be recomputed. Stentz has proposed a different approach: D^* algorithm. D^* starts from the fundamental ideas of A^* and it can be used to find an optimal path in a graph. Graph nodes represent possible robot locations in the configuration space (states) whereas the arcs represent the cost of moving from one state to another.

Considering a start node and a goal node in the graph, let the robot current state be denoted by r . Every node, y , in the graph has a backpointer to its parent node, x , denoted by $b(x)=y$. Just like in the case of A^* algorithm, when the search process is completed, the path is returned using sequences of backpointers from goal to start. The cost of moving the robot from one node, x , to another node, y , is $c(x,y)$, a positive number.

The D^* algorithm uses an OPEN list to propagate arc cost changes and to store sub-optimal paths in the graph. Each node has also attached a tag: $t(x)=NEW$ if the node has never been in the OPEN list before, $t(x)=CLOSED$ if the node was removed from the OPEN list and $t(x)=OPEN$ if the node is currently in the OPEN list.

For each visited node x , the algorithm maintains an estimate of the sum of the arc costs from x to goal given by the path cost function $h(x)$. Given the proper conditions, this estimate is equivalent to the minimal cost from node x to goal node.

For each node x on the OPEN list (i.e., $t(x)=OPEN$), the key function, $k(x)$, is defined to be equal to the minimum of $h(x)$ before modification and all values assumed by $h(x)$ since node x was placed on the OPEN list. The key function classifies a node x on the list into one of two types: a RAISE node if $k(x) < h(x)$, and a LOWER node if $k(x) = h(x)$.

The algorithm uses RAISE nodes in the OPEN list to propagate information about path cost increases and LOWER nodes in the OPEN list to propagate information about path cost reductions. Just like in the case of A^* algorithm, the propagation takes place through the repeated removal of nodes from the list. Each time a node is removed from the OPEN list, it is expanded to pass cost changes to its

neighbors. These neighbors are in turn placed on the OPEN list to continue the process.

Nodes in the OPEN list are sorted by the key function. An important threshold in the functioning of the algorithm is the k_{min} parameter. It is defined as:

$$\forall x | t(x) = OPEN, k_{min} = \min(k(x)) \quad (3)$$

Paths with cost less or equal with k_{min} are optimal, paths with costs greater than k_{min} , may not be optimal. The parameter *kold* is the value of k_{min} before the last node was extracted from the OPEN list.

The aim is to construct, for each node, a sequence of optimal paths to goal. The algorithm consists of two functions: Modify-Cost and Process-State. The Modify-Cost function has the role of changing the arc costs as the robot sensorial system discovers new information during the environment exploration and places the affected nodes in the OPEN list. Function Process-State computes optimal paths to the goal.

The robot starts following the sequence of backpointers to goal until either reaches the goal configuration or its sensors discover a discrepancy between the information in the map (arc cost changes in the graph do not match sensor measurements) and the environment. In this last case, the function Modify-Cost is automatically called to correct the arc costs and place the affected nodes in the OPEN list

2.4 Focussing the D* Algorithm

The drawback of the D* algorithm is in the way it propagates cost changes. These changes are propagated to the affected states regardless of their importance to the robot navigation. The aim is to focus the search and the propagation of cost changes to those states that are likely to generate optimal paths to the goal.

Similar to the case of A* algorithm, D* can also use a heuristic function for decreasing the number of expanded nodes and search focus. Let $g(x,r)$ be the estimated path cost from robot position, r , to the node x . This function will be the focusing heuristic. Furthermore, a new function, f , the estimated robot path cost is defined as follows:

$$f(x,r) = h(x) + g(x,r) \quad (1)$$

All the LOWER nodes in the OPEN list will be sorted using function $f()$ as a sort key. Function $f()$ is the estimated path cost from node r to node goal, passing through node x . Function $f()$ will provide the optimum cost path from r to goal, passing through x , if $g()$ is satisfying the monotonic restriction, due to the fact that $h(x)$ is optimal when a LOWER node is extracted from the OPEN list.

For RAISE nodes, the previous value of function $h()$ defines a lower bound on the $h()$ value of all the LOWER nodes that can be discovered. Thus, if the same focusing heuristic is used, the previous value of $f()$ for the RAISE nodes defines a lower bound for the value of $f()$ for all the LOWER nodes that can be discovered.

Thus, if the value of $f()$ for the LOWER nodes in the OPEN list is larger than the previous value of $f()$ for the RAISE nodes, it is useful to expand the RAISE nodes in order to discover more advantageous LOWER nodes.

Using this work hypothesis, the RAISE nodes in the OPEN list should be sorted using the value of the function $f(x,r)$ as a sort key, and to avoid infinite loops in the backpointers, ties in this key are to be sorted using the value of $k()$.

The process terminates when the lowest value of $f()$ function for all the nodes in the OPEN list is greater or equal to the path cost, since further expanding will not be able to produce a LOWER node with a sufficiently small value of the cost function and located close enough to the current node to influence the search. According to Stentz, this termination is more drastic and abrupt than in the previous case (D^* without the focusing heuristic).

The major problem in using a focusing heuristic is that once an optimal path to the goal has been found, the robot starts following backpointers to the goal state and moves to another node and the problem is that the nodes in the OPEN list are sorted based on the value of the path cost computed for the old robot position and thus the nodes in the OPEN list have incorrect values for the functions $f()$ and $g()$. One possible solution is to calculate these functions each time the robot moves or a node is inserted in the OPEN list. Empirical results of Berg, Hansen and Zhou, have shown that this is a major slow-down in the algorithm and the speed-up gained through focusing search is outrun by the slow-down introduced by the recalculation of $f()$ and $g()$.

It has been proven by Stentz, that is an advantage that usually the robot moves only a few nodes before a re-planning operation is necessary. Thus, the values of $f()$ and $g()$ functions are only slightly deviated. Assuming that a node x is placed in the OPEN list at the time the robot is in the configuration indicated by the node $r0$ and the value of $f()$ is $f(x,r0)$. If the robot moves to another node, $r1$, $f(x,r1)$ may be computed and the position of node x in the OPEN list may be adjusted. On the other hand, to avoid the computational cost, one may compute a lower bound on the value of $f(x,r1)$:

$$f_L(x,r_1) = f(x,r_0) - g(r_1,r_0) - \varepsilon \quad (5)$$

Function f_L represents a lower bound on $f(x,r_1)$, since it assumes that the mobile robot has moved in the direction of the node x , thus the cost of $g(r1,r0)$ is subtracted. The parameter ε is a positive constant.

States are sorted on the OPEN list by a biased $f()$ value, given by $f_B(x,r_i)$, where x is the node in the OPEN list and r_i is the robot's state at the time x was inserted or adjusted on the OPEN list. Let $\{r_0, r_1, \dots, r_m\}$ be the sequence of nodes occupied by the robot when the nodes were inserted in the OPEN list. The value of $f_B()$ is given by:

$$f_B(x,r_i) = f(x,r_i) + d(r_i,r_0) \quad (2)$$

where $f()$ is the estimated robot path cost given by:

$$f(x, r_i) = h(x) + g(r_i, r_{i-1}) \tag{7}$$

and $d()$ is the accrued bias function given by:

$$\begin{cases} d(r_i, r_0) = g(r_1, r_0) + g(r_2, r_1) + \dots + g(r_i, r_{i-1}) + \varepsilon, i > 0 \\ d(r_0, r_0) = 0, i = 0 \end{cases} \tag{3}$$

The function $g(x,y)$ is the focusing heuristic, representing the estimated path cost from a node y to a node x . The nodes in the OPEN list are sorted by increasing $f_B(\)$ value, with ties in $f_B(\)$ ordered by increasing $f(\)$, and ties in $f(\)$ ordered by increasing $k(\)$. Ties in $k()$ are ordered arbitrarily. Thus, a vector of values $\langle f_B, f, k \rangle$ is stored with each node in the list.

3 Experimental Results

Several experiments were made, in both simulation and real life, using the Pioneer2 mobile robot, to determine the advantages and disadvantages of using A* and D*.

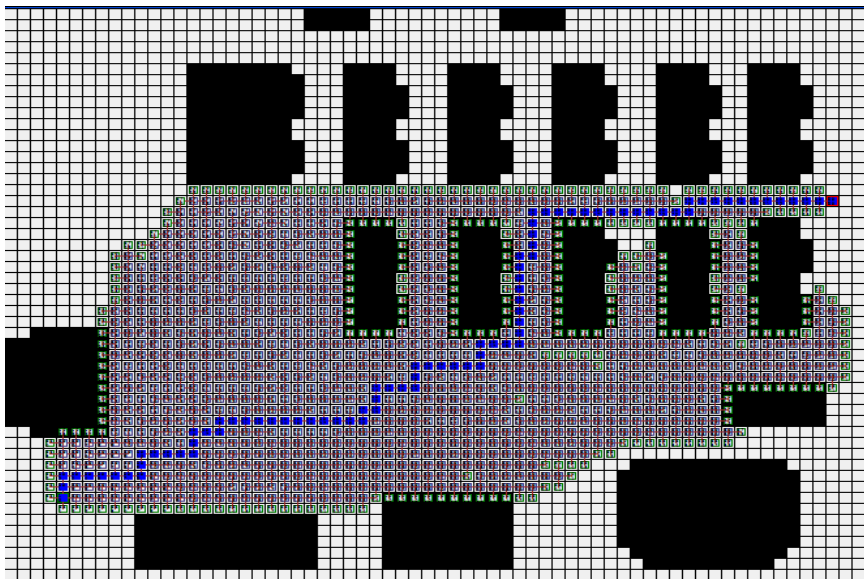


Fig. 1. Path returned by A* algorithm

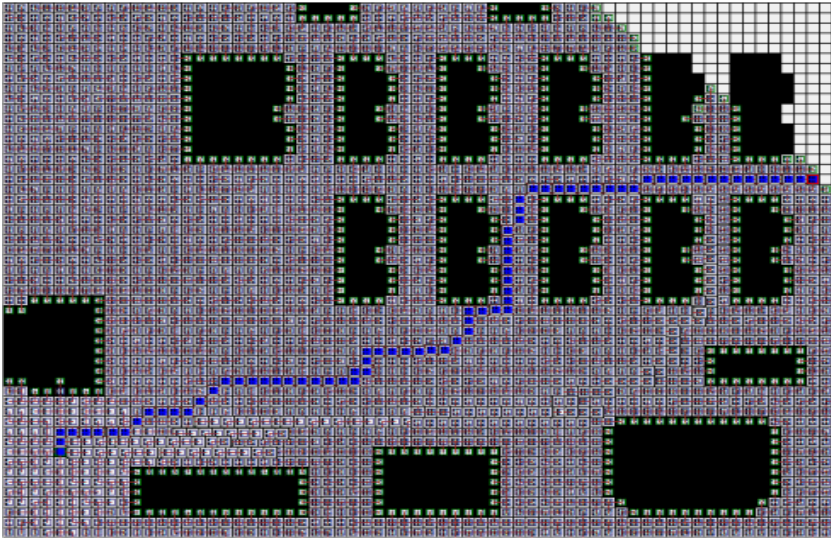


Fig. 2. Path returned by D* algorithm without focusing heuristic

Fig. 1 presents a simulated environment having the configuration space similar to the obstacles distribution in the Robotic Research Lab at Technical University of Cluj-Napoca. Each cell in Fig. 1 represents a square area of 15 cm \times 15 cm. The initial robot configuration is in the lower left corner (green square) whereas the goal configuration is in the upper right corner (red square).

The path generated by the A* algorithm is also presented in Fig. 1. Distances between nodes were measured using the Manhattan metric, so that any neighboring node on a N, S, E or W direction is at a distance of 1 from the current node, whereas nodes on NW, NE, SW and SE direction are at a distance of 2 from the current node.

Fig. 2 presents the path generated by the D* algorithm without the focusing heuristic, for the same environment configuration as in Fig. 1. Expanded nodes are presented in both Fig. 1 and Fig. 2; nodes depicted with a green rectangle are the nodes in the OPEN list (on the frontier of the area representing the set of expanded nodes) whereas the nodes on the optimal path are presented in dark blue.

Fig.3 presents the path generated by the D* algorithm with the focusing heuristic, for the same environment configuration as in Fig. 1

By analyzing Fig. 1 to Fig. 3, several questions have to be answered: first, why the resulting path in the three cases is not identical? And second, what are the benefits of using D* like algorithms, since, at a first glance, the A* seems to provide best results?

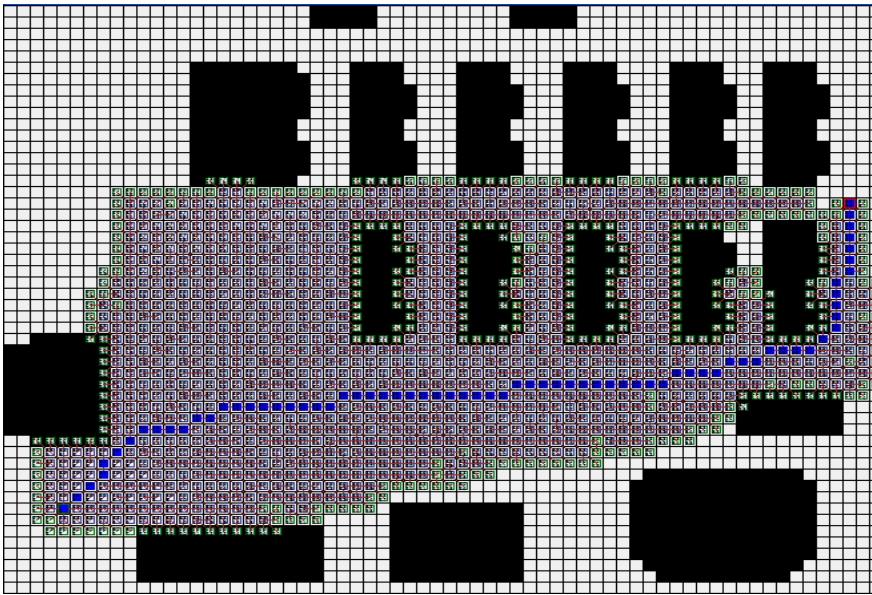


Fig. 3. Path returned by focussed D* algorithm

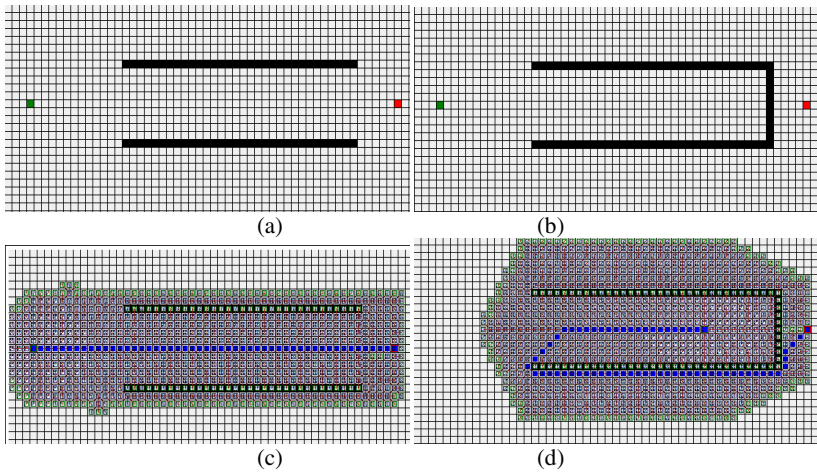


Fig. 4. Example of A* planning

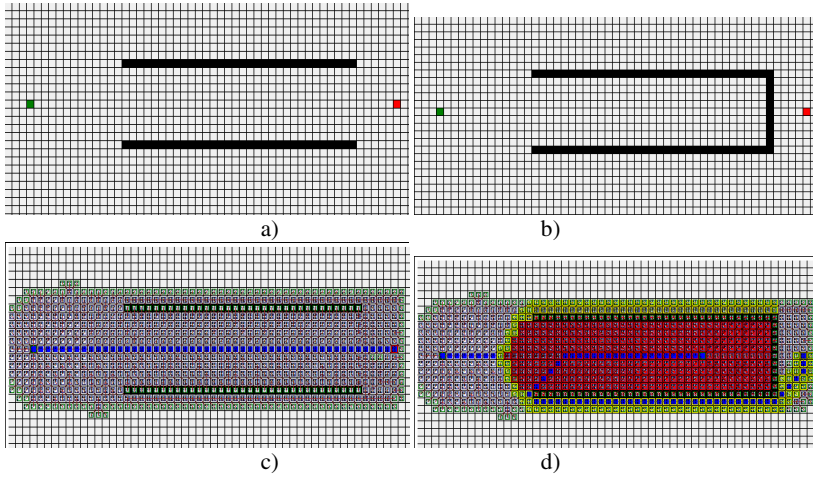


Fig. 5. Example of D* path planning

The fact that both A* and focused D* are using a focusing heuristic, whereas D* is not using it must be kept in mind when analyzing the path length in the three situations (Fig. 1 to Fig. 3). Moreover, the distances are determined using the Manhattan metric, thus the results may appear different due to aliasing. The path cost (the sum of all arcs) is minimum in the three cases. In addition, focused D* uses three keys to sort the nodes in the OPEN list, ties resulted by using the first sort criterion (value of f_B) are solved by using the value of f whereas ties in this case are solved using the third sort key, that is the value of k .

The second problem is that both A* and focused D* algorithms expand the same number of nodes (Fig. 1 and Fig. 3), whereas D* expands a larger number of nodes. We will analyze the situation when the information in the map is incomplete or the structure of the environment changes. Assume the robot is equipped with sensors capable of detecting the environment on a radius of 10 nodes around the mobile robot. The robot is supposed to traverse a corridor that has a door at the end. The robot has no information about the presence of the door.

Fig. 4 presents an example of A* path planning: subfigure (a) presents the environment as it is known by the planner, whereas subfigure (b) presents the real structure of the environment. Subfigure (c) presents the initial path plan (through the closed door) whereas subfigure (d) presents the re-planned path, after the robot discovers the closed door.

Fig.5. shows an example of D* path planning: subfigure (a) presents the environment as it is known by the planner, whereas subfigure (b) presents the real structure of the environment. Subfigure (c) presents the initial path plan (through the closed door) whereas subfigure (d) present the re-planned path, after the robot discovers the closed door; nodes in red represent RAISE nodes, whereas nodes in yellow are LOWER nodes.

Experiments presented in Fig. 4 and in Fig. 5 shows the different way algorithm A* and focused D* operate when facing the same problem. Both algorithms plan an initial path through the closed door (subfigure (c) in both Fig. 4. and Fig. 5.) because, based on the initial information, this is an unobstructed path. Then the robot starts following backpointers to the goal configuration until the closed door enters the range of the mobile robot sensorial system. At this moment the planner based on A* updates the map and starts a new planning process having the start node the robot current position, whereas the goal node remains unchanged.

On the other hand, the D* algorithm tries to repair the map (the affected portion of the map containing the initial path through the closed door). The number of expanded cells is smaller than in the case of A* because the algorithm uses portions of mps that are unaffected by the cost changes.

Fig. 6 presents a navigation of the Pioneer 2 mobile robot in a real-life environment, having the same characteristics as the environment used for simulations. Even if the information stored in map is completely accurate (the algorithm is completely informed), cost changes in arcs are due to a series of external factors such as: localization errors, error in specifying the initial robot position, errors in the data provided by the sensorial system of the robot and last but not least, error of the robot's odometric system.

Several tests have been performed in order to determine the average running time between breadth-first search, Fast A* implementation, D* without focusing heuristic and focused D*. The tests were performed off-line on random generated maze-like maps, represented as eight-connected grid. The maps contain 35% of blocked cells and have adjustable dimensions of 100×100 cells, 1000×1000 cells and 10000×10000 cells (except for the breadth-first search which was inefficient and the memory requirement was too large for such a high number of cells).

Table 1 presents the run-time results (in seconds) whereas Table 2 presents a comparison between the number of expanded cells for each complete planning-replanning process. In addition to these tests, we present an real life experiment with the Pioneer2 mobile robot. The robot had to traverse a partially known environment, having the same structure as presented in fig. 1. Additional information on the environment is provided by the robot sensorial system; however, localization errors and sensor noise affect the map-matching of these data. Fig 6 presents the such an experiment where the mobile robot has detected an obstacle that was not in the map.

The Open List in A* and D* is implemented as a balanced binary tree sorted on corresponding key values, with tie-breaking mechanism. This tie-breaking mechanism results in the goal state being found on average earlier in the last $f()$ value pass.

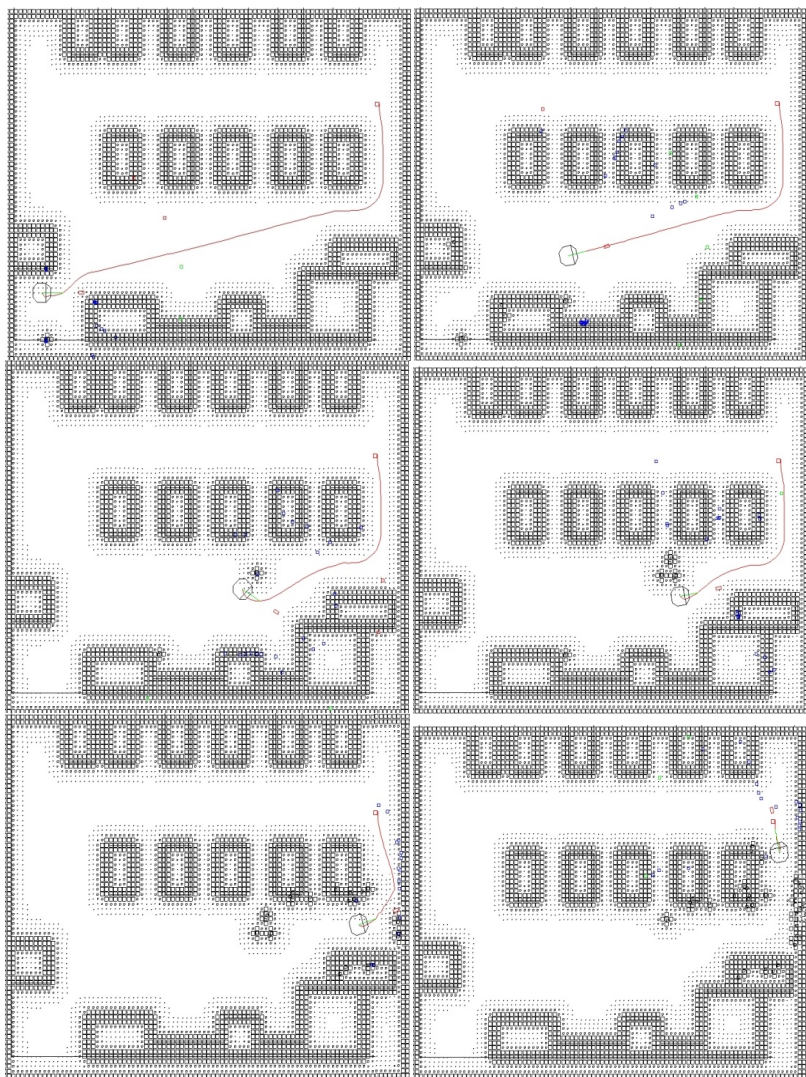


Fig. 6. Intermediate steps in navigation of Pioneer 2 mobile robot with Focused D*

In addition to the standard Open/Closed Lists, marker arrays are used for answering (in constant time) whether a node is in the Open or Closed List.

We use a “lazy-clearing” scheme to avoid having to clear the marker arrays at the beginning of each search. Each path finding search is assigned a unique (increasing) id that is then used to label array entries relevant for the current search. The above optimizations provide an order of magnitude performance improvement over a standard “textbook” A* implementation. All experiments were run on a 2.1 GHz PC under MS Windows XP.

Table 1. Comparison between running time of the breadth-first search, A*, D* and focused D* in planning and re-planning paths

Dimension	Breadth-First	Fast A*	D*	Focused D*
Planning 10^4 cells	35.2s	5.7 s	8.0 s	6.2 s
Re-planning 10^4 cells	12.7s	3.0 s	2.1 s	1.3 s
Planning 10^6 cells	178.9s	37.3 s	55.8 s	50.7 s
Re-planning 10^6 cells	113.2s	28.2 s	10.1 s	7.6 s
Planning 10^8 cells	-	136.4 s	335.0 s	298.7 s
Re-planning 10^8 cells	-	126.8 s	87.4 s	54.3 s

Table 2. Comparison between number of expanded cells

Dimension	Breadth-First	Fast A*	D*	Focused D*
10^4 cells	625982	9658	15352	1672
10^6 cells	5569854	21566	36254	7625
10^8 cells	-	153694	279125	16369

4 Conclusions

Although specific to artificial intelligence, the A* and D* demonstrate their impact on any applications requiring graph search, including mobile robotics. This is due to the fact that both A* and D* are generic algorithms, applicable to any optimum path problems.

The A* algorithm is capable of producing optimum paths (lowest cost path) as long as the structure of the environment is completely known (arc costs do not change during robot traverse). In the case where discrepancies exist between the map and the structure of the environment, the efficiency of A* is limited, due to the necessary re-planning operation.

These operations are time consuming since the algorithm is not capable of using information retrieved between searches or the costs of partially expanded nodes, thus any re-planning operation means another planning from with zero information from the previous search. These deficiencies are eliminated by D*. As opposed to A*, D* can cope with arc cost changing during robot traverse. This because the algorithm is capable of using the partially expanded nodes and subsequent path costs leading to smaller wait time between re-planning operations.

Like in the case of A*, D* can also use a heuristic to focus the search and propagate the cost changes in graph.

Acknowledgment. This paper was supported by the project Progress and development through post-doctoral research and innovation in engineering and applied sciences – PRiDE - Contract no. POSDRU/89/1.5/S/57083", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

References

1. Berg, J.V.D.: Anytime path planning and replanning in dynamic environments. In: Proceedings of the International Conference on Robotics and Automation, pp. 2366–2371 (2006)
2. Eklund, P.W., Kirkby, S., Pollitt, S.: A dynamic multi-source Dijkstra' algorithm for vehicle routing. In: Proc. of Conf. on Intelligent Information Systems, Australia and New Zealand (1996)
3. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
4. Bonet, B., Geffner, H.: Planning as heuristic search. *Artificial Intelligence* 129, 5–33 (2004)
5. Goldberg, A.V., Harrelson, C.: Computing the shortest path: A* search meets graph theory. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 156–165 (2005)
6. Brumitt, B., Stentz, A.: GRAMMPS: a generalized mission planner for multiple mobile robots. In: Proceedings of the International Conference on Robotics and Automation (1998)
7. Koenig, S., Likhachev, M.: Incremental A*. In: *Advances in Neural Information Processing Systems*, vol. 14. MIT Press (2002)
8. Buckland, M.: AI techniques for game programming. Premier Press, Portland (1992)
9. Hansen, E.A., Zhou, R.: Anytime heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 28, 267–297 (2007)
10. Stentz, A.: The focussed D* algorithm for real-time replanning. *International Journal of Robotics and Automation* (1995)
11. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: Proceedings IEEE International Conference on Robotics & Automation, pp. 3310–3317 (1996)

Stability Analysis Software Platform Dedicated for a Hexapod Robot

Sorin Mănoiu-Olaru and Mircea Nițulescu

Department of Mechatronics, Blvd. Decebal. 107, 200440, Craiova, Romania
manoiusorin2006@yahoo.com, nitulescu@robotics.ucv.ro

Abstract. In this paper the authors present a software program to simulate hexapod robot stability in gravitational field for a certain configuration of legs using Matlab software package. The simulation software was created using geometrical modelling based on Denavit-Hartenberg algorithm and analyses the static stability of the robot in different stages of locomotion on horizontal surface for different leg configuration. The paper includes some experimental results related to the static gravitational stability depending on the support polygon formed by the legs on the ground.

Keywords: Matlab, gravitational stability, Denavit-Hartenberg representation, model, hexapod.

1 Introduction

Walking machines allow locomotion on rough and irregular surfaces with a high degree of softness [1]. This is why legged machines received increasing attention by the scientific community [2]. Current vehicles we are used to have wheels for locomotion. Wheeled vehicle can achieve high speed with a relative low control complexity but only on structured terrain. Since most of the earth's land surface is inaccessible to regular vehicles there is a need for mobile robots that can handle difficult terrain.

The conventional walking machine with three degrees of freedom for each leg has great flexibility during terrain motion [3]. An important drawback of legged machines is the complexity of the control required to achieve walking even in completely flat and horizontal surface in which much simpler wheeled machines work perfectly well [4]. This means that the use of legged machines is only justified if they can walk on irregular terrain with certain degree of confidence.

Most popular hexapods can be grouped into two categories: rectangular (with two groups of three legs distributed symmetrically on the two sides) and hexagonal (round or hexagonal body with evenly distributed legs).

The motion of legged robots can be divided into statically and dynamically stable. Static stability means that the robot is stable at all times during its gait cycle.

Dynamic stability means that the robot is only stable when it is moving. For legged robots, static stability demands that the robot has at least three legs on the ground and the robot's centre of mass is inside the support polygon, i.e. the convex polygon formed by the feet supporting the robot (Fig.1).

On the left side four legs provide support and the centre of mass is located inside the support polygon so the robot is statically stable. On the middle the bottom left leg has been lifted, putting the centre of mass outside the support polygon which made the robot unstable. On the right side three legs provide support and the centre of mass is located on one side of the support polygon. This case is called critical stability.

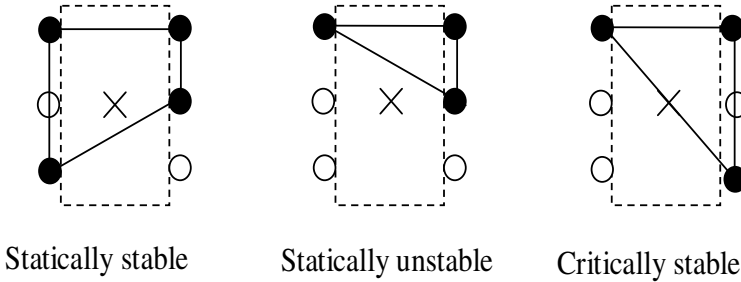


Fig. 1. Stability cases for a hexapod robot: statically stable, unstable cases and critically stability

Some of the most important advantages of legged locomotion are [5]:

- accommodation to uneven terrain
- use of isolated footholds
- providing active suspension
- environmental effects of legged vehicles are less than wheeled or tracked vehicles

Among disadvantages of legged locomotion we can enumerate:

- artificial walking mechanisms are so far heavy due to large number of actuators
- control of walking is very complex and so far walking vehicles are rather slow
- bad payload-weight-to-mechanism-weight ratio compared to wheeled or tracked vehicles
- appearance of an impact force with each step made.

2 Hexapod Robot Model

The legged locomotion on natural terrain presents a set of complex problems (foot placement, obstacle avoidance, load distribution, general stability) [6] that must be

taken into account both in mechanical construction of vehicles and in development of control strategies. One way to handle these issues is using models that mathematically describe the different situations. Therefore modelling becomes a useful tool in understanding systems complexity and in testing and simulating different control approaches [7], [8].

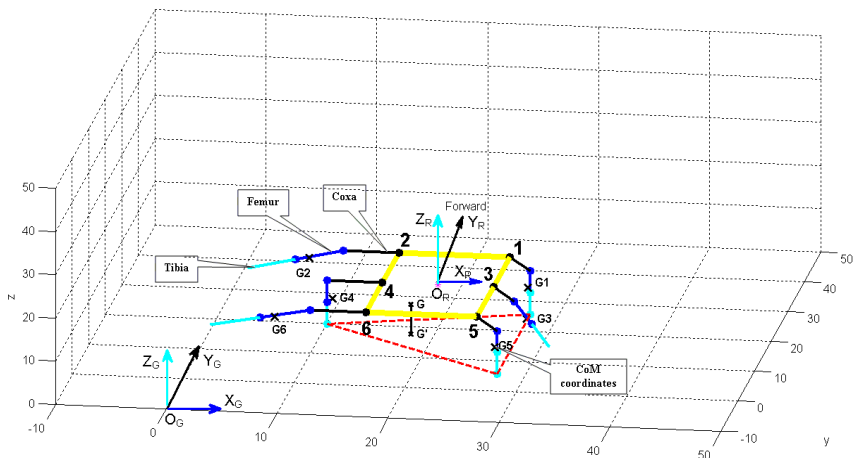


Fig. 2. Hexapod robot structure

The robot structure considered has 6 identical legs and each leg has 3 degree of freedom (RRR) (Fig. 2). All the relevant points have been put on the model as can be seen from figure 2: coordinates of the centre of mass of each leg $G_i, i=1...6$; leg numbering for easy understanding (1 to 6), coordinates of the centre of mass of the robot G , projection of the centre of mass into the support polygon G' , robot's centre of symmetry with the attached frame $O_R(X_R, Y_R, Z_R)$, the global frame $O_G(X_G, Y_G, Z_G)$, and direction of motion.

The global frame is the frame that all other frames will be defined relative to. The global frame is rigidly attached to the lower left corner of the world so that the z-axis is vertical and the xy-plane is aligned with the floor surface.

The origin of the robot coordinates is attached in the centre of symmetry with the z-axis pointing up, the x-axis pointing left and the y-axis pointing forward.

2.1 Robot Leg

The successful design of a legged robot depends to a large extent on the leg design chosen. Since all aspects of walking are ultimately governed by the physical limitations of the leg, it is important to select a leg that will allow a maximum range of motion and that will not impose unnecessary constraints on the walking.

A three-revolute kinematical chain has been chosen for each leg mechanism in order to mimic the leg structure (Fig. 3). A direct geometrical model for each leg mechanism is formulated between the moving frame $O_i(x_i, y_i, z_i)$ of the leg base, where $i=1 \dots 6$, and the fixed frame $O_R(X_R, Y_R, Z_R)$.

The coordinate frames for the robot legs are assigned as in fig. 3. The assignment of link frames follows the Denavit- Hartenberg direct geometrical modelling algorithm. The robot leg is made of links and joints as noted in figure 3. The different links of the robot legs are called *coxa*, *femur* and *tibia*.

The robot leg frame starts with link 0 which is the point on the robot where the leg is attached; link 1 is the coxa, link 2 is the femur and link 3 is the tibia. Legs are distributed symmetrically about an axis in the direction of motion (Y in this case).

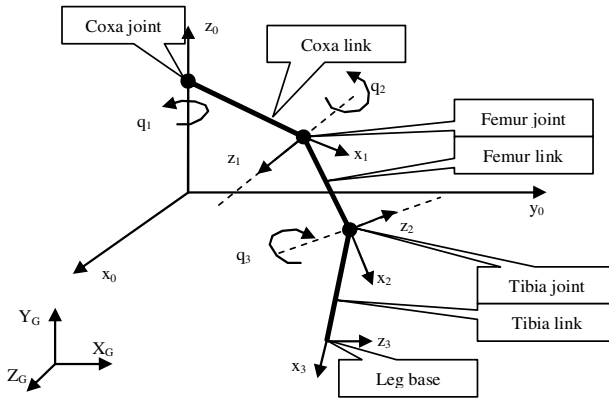


Fig. 3. Model and coordinate frame for leg kinematics

The general form for the transformation matrix from link i to link $i-1$ using Denavit Hartenberg parameters is given in equation 1:

$$T_i^{i-1} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The transformation matrix is a series of transformations:

- translate d_i along z_{i-1} axis,
- rotate θ_i about z_{i-1} axis,
- translate a_i along x_{i-1} axis,
- rotate α_i about x_{i-1} axis.

The overall transformation is obtained as a product between five transformation matrixes:

$$T_{O_G}^{base} = T_{O_G}^{O_R} T_{O_R}^{coxa} T_{coxa}^{femur} T_{femur}^{tibia} T_{tibia}^{base} \quad (2)$$

The product of the last three matrixes determines the geometrical model of the leg:

$$T_0^3 = T_0^1 T_1^2 T_2^3 \quad (3)$$

The centre of mass of each link is positioned relative to the link frame by a position vector $p_i = [x_i, y_i, z_i, 1]^T$. To find the position of the centre of mass of each link relative to leg frame, the coordinates p_i are multiplied with the D-H transformation giving the centre of mass positions:

$$P_{CoM_i} = T_0^i p_i, i = 1 \dots 3 \quad (4)$$

The position of centre of mass of the leg is calculated using the equations:

$$x_g = \frac{\sum_{i=1}^3 x_i m_{li}}{\sum_{i=1}^3 m_{li}}, y_g = \frac{\sum_{i=1}^3 y_i m_{li}}{\sum_{i=1}^3 m_{li}}, z_g = \frac{\sum_{i=1}^3 z_i m_{li}}{\sum_{i=1}^3 m_{li}} \quad (5)$$

where:

m_{li} – mass of link i

The position of robot's centre of mass is calculated using the following equations:

$$X_G = \frac{\sum_{i=1}^6 x_{gi} m_{Li}}{\sum_{i=1}^6 m_{Li}}, Y_G = \frac{\sum_{i=1}^6 y_{gi} m_{Li}}{\sum_{i=1}^6 m_{Li}}, Z_G = \frac{\sum_{i=1}^6 z_{gi} m_{Li}}{\sum_{i=1}^6 m_{Li}} \quad (6)$$

where:

$$m_{Li} = \sum_{i=1}^3 m_{li} \quad (7)$$

m_{Li} - mass of leg i

Limitations for each joint are:

- $q_{coxa} = [-\pi/4, \pi/4]$,
- $q_{femur} = [-\pi/4, \pi/2]$,
- $q_{tibia} = [0, 3 \cdot \pi/4]$.

3 Simulation Platform

The main purpose of this simulation platform is to show how the support polygon modifies when legs lose contact with ground and if the projection of the centre of mass is within the support polygon.

The simulation program was made using MatLab GUIDE [9] (Graphic User Interface Design Environment).

The developed software platform can be used to analyze what happens with the hexapod robot in gravitational field and also allows communication with the leg in the real world.

The communication between the physical leg and Matlab environment is made using Arduino Duemilanove development board.

The analysis of the hexapod robot in gravitational field can be group in two modes:

- Free fall mode. In this mode the mechanical configuration of the legs is defined by their joints values, no additional move is allowed.
- Transitory analysis. This mode is used to analysis what happens with the robot between two static regimes.

Giving a set of values for legs joints yields a stationary mechanical configuration which generates a certain support polygon in relation with which we analyze the gravitational stability of the hexapod robot.

For a certain configuration of legs, the robot is statically stable if the projection of the centre of mass is inside the support polygon; it is at stability limit if the projection of the centre of mass is on one side of the support polygon and it is statically unstable if the projection of the centre of mass is outside the support polygon. In this last case the robot is shifting gradually its support polygon by lifting/touching the ground with its legs, due to gravity, until the condition for static stability is accomplished.

The shape of the support polygon needed for minimum static stability is the triangle.

3.1 Program Interface

When the interface is launched the robot is first drawn in a stable configuration as shown in fig. 4. The interface is divided basically into 2 areas: in the upper left the robot is displayed (plotted) according to the values set by the user and in the upper right and bottom we can find the controls for the robot. The controls for the robot are structured mainly in 2 parts: joints control, position control. Joints control and position control consists of a list where the leg number it put on and a panel where the user can set the values for each joint or position.

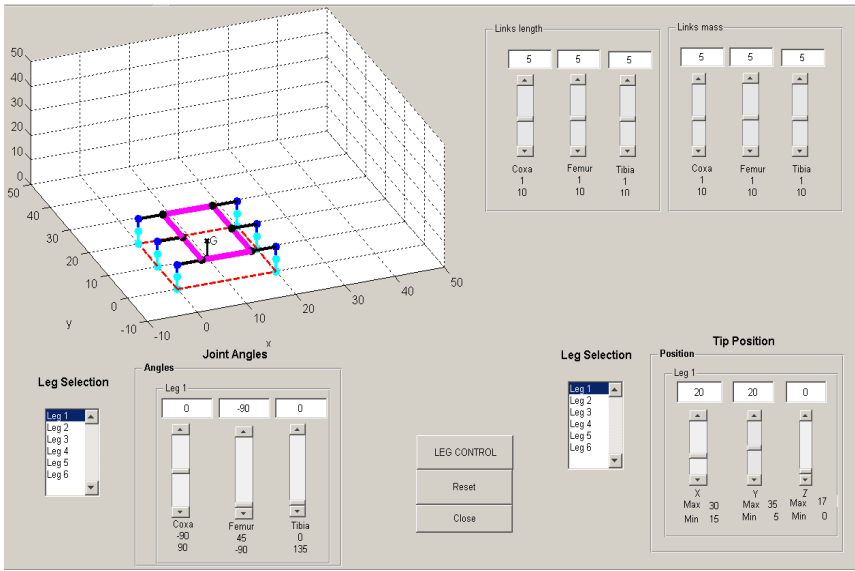


Fig. 4. Matlab robot platform interface

The controls for each leg are encapsulated into a panel identified by leg number. Every slider has an editable textbox where the value is displayed and controls a certain link of the robot. If we use a slider the associated editable textbox value is updated and vice versa. Also the robot leg position is updated with the data from the slider or from the textbox. The controls for link length or mass affect all the legs because they are considered identical.

The button label *Leg Control* from fig. 4 launches a second interface like in fig. 5. The second interface allows control of a single leg. This interface can be used both in offline mode or online mode.

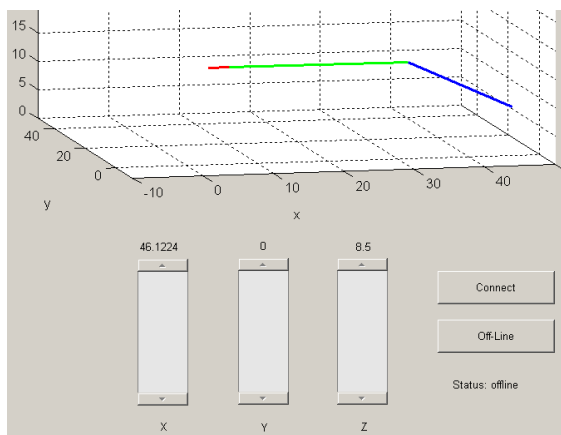


Fig. 5. The communication interface used to control the real leg using Matlab

The graphical representation of the robot also allows seeing the shape of the support polygon which is updated according to the legs on the ground. The support polygon is drawn only if the distance from the tip of the leg to the ground is smaller than a threshold. This threshold is modifiable (but not present in the interface) and was introduced as a way to compensate certain position errors that may occur due to real servomotors. The simulation program shows all the stages the robot goes through for a better understanding. For simplicity and better understanding of the robot stages the model is drawn in a simpler way.

3.2 Leg Control

The system proposed by the authors (Fig. 6) is similar with the system xPC-Target component of Matlab. Of course our system does not have its performance but it is a lot cheaper. The software that make possible the communication between Arduino board and Matlab has been released with the last version of Matlab. Mathworks [10] has also developed support for Arduino in Simulink. A part of the communication software is uploaded on the Arduino board and plays the server role.

There are two programs that can be uploaded on the board:

- *adiosrv.pde* with which all the input and output of the board can be manipulated.
- *motorsrv.pde* designed exclusively for motor control via a motor shield.

The major drawback of using the original *motorsrv.pde* was that this file was developed as support for a specific motor shield that can only control 2 servomotors. So we modify the file in a way that now allows using all 6 PWM channels available.

The other part is a Matlab class (*arduino.m*) and plays the role of the client. Once the class is instantiated it makes possible sending commands over USB port.

The direction of motion of the servo is automatically determined. If we set a rotation with 90^0 and then a 30^0 rotation the servo will rotate 60^0 anticlockwise.

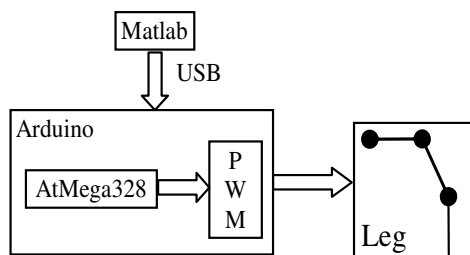


Fig. 6. Leg control diagram

3.3 Walking Algorithm

During walking or running the leg move cyclically and in order to facilitate analysis or control, the motion of the leg is often partitioned in two parts:

- support phase or stance when the robot uses the leg to support and propel.
- transfer phase or swing when the leg is moved from one foothold to the next.

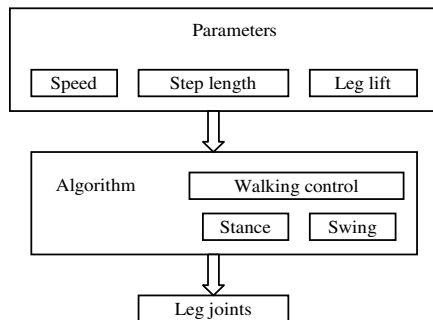


Fig. 7. Walking Algorithm

The stance part of the walking algorithm is supposed to move the leg in a straight line. The swing part of the algorithm must lift the leg off the ground, move it back to the starting position and lower it down to the ground again. The walking algorithm is based on the kinematical model of the leg. Parameters introduced in the algorithm (Fig. 7) are:

- speed; define the speed of the leg tip,
- step length; define the length of the step,
- leg lift; this defines how high the leg is lifted when it's in the swing-cycle.

For a smooth straight motion, the swing time must be equal to the stance time for each leg.

In the stance part of a step, the leg only moves in a straight line. At time t from the start of the step, the coordinates (x, y, z) for the trajectory will be:

$$\begin{aligned}
 x &= \text{leglength} \\
 y &= \frac{\text{steplength}}{2} - t * \text{speed} \\
 z &= -\text{legheight}
 \end{aligned} \tag{8}$$

In the swing part of the step, the leg first has to be lifted off the ground, and then moved back to where the next step is supposed to start and then lowered to the ground. To find out where in the swing-cycle the leg is at time t from the start of the cycle we first have to calculate the total length the leg has to travel:

$$dist = \frac{(2 * leglift - steplength) * t}{swing_cycle} \quad (9)$$

One step consists of one stance cycle and one swing cycle but to maintain a smooth motion of the leg, it's important that the swing cycle continues where the stance cycle ended, and that the swing cycle ends where the new stance cycle starts. This does not only hold for the position, but also for the speed and the direction of the speed.

3.4 *Static Stability Condition*

The determination of static stability condition is resumed at finding if the projection of robot's centre of mass is inside the support polygon. For this the authors used the following algorithm:

- support phase or stance when the robot uses the leg to support and propel.
- determine the convex polygon
- determine the area of the convex polygon (A)
- form the n triangle using 2 consecutive sides of the support polygon and the projection of centre of mass, G', (e.g. ABG', BCG'... etc.)
- determine the areas of the n triangles formed (A_i)
- if (ΣA_i = A) then condition is true
 - else condition is false

3.5 *Simulation Algorithm for Stability Analysis*

The authors elaborated an algorithm in order to achieve the goal of analyzing the static stability of a hexapod robot in gravitational field. The algorithm is structured in 5 steps as following:

- support phase or stance when the robot uses the leg to support and propel.
- setting the joints values
- determine the mechanical configuration
- determine which legs are on the ground
- evaluation of the static stability condition
- while (condition of static stability = false)
 - determine the rotation line using the minimum distance from G' to support polygon's sides
 - rotate the robot about the line found
 - determine which legs are on the ground
 - evaluation of the static stability condition

4 Experimental Results

4.1 Free Fall Analysis

The free fall analysis represents what happens with the robot left to fall on the ground from a height greater than the extension of the legs. Keeping in mind that the joints are locked by the values prescribed by the user, no extra movements are allowed (no active stability). The only force that acts upon the robot is the gravitational force. For a given set of joint values the robot passes through many transitory stages until it becomes statically stable (Fig. 8.a, b, and c). Legs that have contact with the ground determine the shape of the support polygon (triangle, quadrilateral, pentagon or hexagon). In order to know if the robot achieves static stability the projection of G (G' for now on) must be inside the support polygon. To solve this problem the above algorithm is applied.

As it can be seen in figure 8.a, when the robot falls the first legs that reach the ground are those nearest the ground. Also G' is not inside the support polygon and the robot continuing its falling and rotates about the line determined by the leg 2 and leg 3. The rotation line is determined by calculating the minimum distance from G' to polygon's sides. In this case the first leg closest to ground that will provide support is leg 5.

In figure 8.b it can be seen that even in this configuration G' is not inside the support polygon and the robot continues its falling and rotates about the line determined by leg 2 and leg 5 until the first leg touches the ground, which in this case, is leg 4.

In figure 8.c a new configuration is formed and if the algorithm described above it is applied, point G' is inside the support polygon and the robot becomes statically stable and the falling stops.

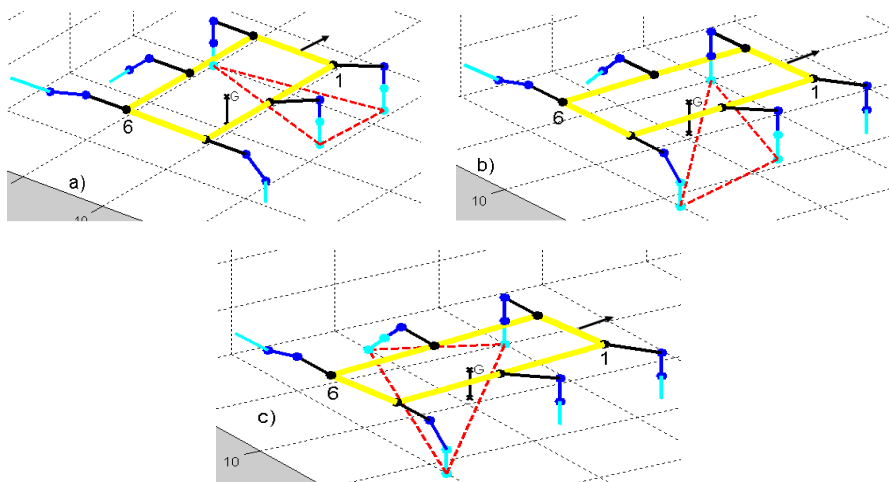


Fig. 8. Free fall analysis **a.** Phase one of falling, **b.** Phase two of falling, **c.**Phase three – statically stable

4.2 Transitory Analysis

In this mode of analysis the robot passes between two static regimes. A static regime is identified by the condition of stability. The user can alter a static regime using the controls for joint values. This analysis can also be interpreted as a continuation of *free fall analysis* case if the condition of stability has been met.

At legs considered on the ground the tip will become rounded as can be seen in all figures.

In figure 9.a the robot has static stability, the support polygon described by the legs on the ground is a quadrilateral and the projection of the centre of mass is inside the support polygon.

Next, lifting leg 5 the support polygon changes its shape becoming a triangle. Using the algorithm described above, the projection of G is not inside the support polygon and the robot becomes statically unstable and starts falling (Fig. 9.b).

Following the algorithm the next leg closest to the ground is leg number 4. In figure 9.c the projection of G is inside the newly support polygon, the robot stops falling and becomes statically stable.

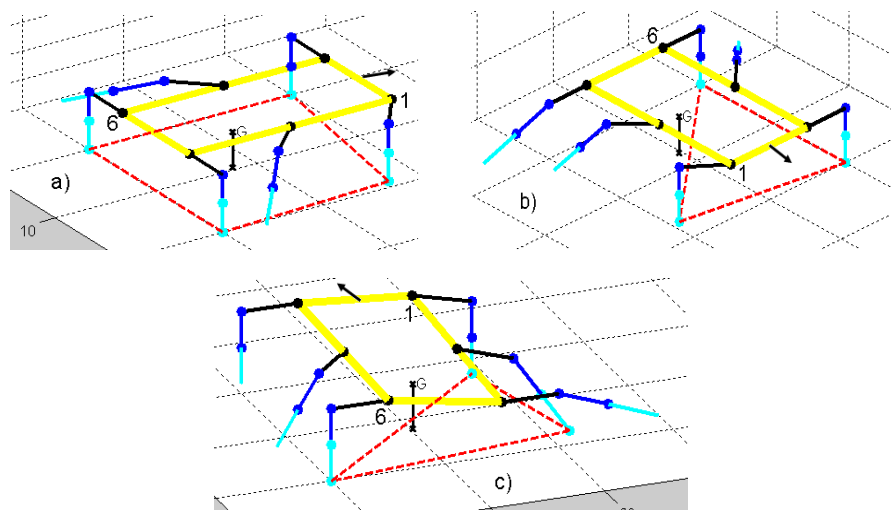


Fig. 9. Transitory analysis. a. Phase one: statically stable, b. Phase two of falling c. Phase three: statically stable

4.3 Hardware Leg Control

The algorithm for controlling the leg joints, including direct kinematics and inverse kinematics, was implemented in Matlab in order to analyze leg performance.

The main errors occurred due to the fact that we can only send integer numbers for joint values.

Other errors occurred due to the fact that the joint are assembled using bolts and nuts. Normal usage of the robot causes the nuts and bolts to loosen causing a lot of clearance in the joints.

The results of the tests can be view in the chart below.

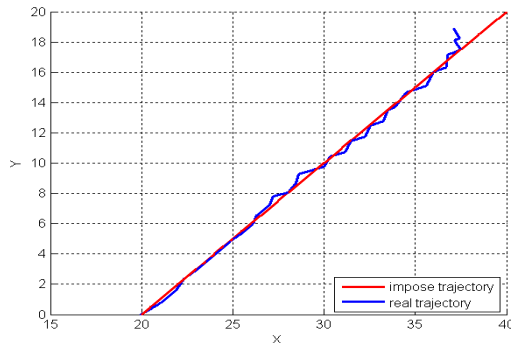


Fig. 10. Hardware leg control results

5 Conclusion

In this paper a simulation platform for legged mobile robots was presented, that allows stability analysis and full control of the robot.

Free fall analysis is useful for investigating what happens with the robot on uneven terrain or for accidents that may happen due to lose of contact, slippery surface, servomotor failure, power supply failure.

Transitory analysis represents a more important case for locomotion, gait generation. When starting to develop a gait cycle we can have a big picture of what happens when the robot starts to move, how the support polygon changes and what actions (what leg should be actuated) must be applied to the robot in order to meet condition of stability.

The interface was designed to be simple and intuitive and to offer the user a simple and efficient way to control every aspects of the robot (angles, masses, lengths).

The two analyses made in this article represent the bases for next activities on static stability.

The program can also be used for legged mobile robots with 4 or 8 legs using minor code modifications.

In the future the program will be upgraded permitting additional controls and functions for stability analysis (including dynamic stability) on uneven ground and implementing collision detection algorithms. Also the results of these studies represent the bases for different strategies of locomotion on different terrains. The experimental results will become a standard for a real hexapod robot.

The interface will be imbuing with additional walking algorithms and controls.

Hardware leg control will be imbuing to better precision.

Acknowledgment. This work was partially supported by strategic grant POSDRU/88/1.5/S/50783, Project ID 50783 (2009), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resource Development 2007 – 2013.

References

1. Giorgio, F., Pierluigi, R.: Mechanics and Simulation of Six-Legged Walking Robots. In: Climbing & Walking Robots Towards New Applications, p. 546. Itech Education and Publishing, Vienna (2007) ISBN 978-3-902613-16-5
2. Nelson, G.M., Quinn, R.D., Bachmann, R.J., Flannigan, W.C.: Design and Simulation of a Cockroach-like Hexapod Robot. In: Proceedings of the 1997 IEEE International Conference on Robotics and Automation Albuquerque, New Mexico (April 1997)
3. Maki, K.H.: Bioinspiration and Robotics: Walking and Climbing Robots. I-Tech Education and Publishing, Croatia (2007) ISBN 978-3-902613-15-8
4. Carbone, G., Ceccarelli, M.: Legged Robotic Systems. In: Cutting Edge Robotics, pp. 553–576. ARS Scientific Book, Wien (2005)
5. Nitulescu, M.: Robotic Systems with Navigation Capabilities. Universitaria Craiova, Romania (2002) ISBN: 973-8043-143-3
6. Krzysztof, W., Dominik, B., Andrzej, K.: Control and environment sensing system for a six-legged robot. *Journal of Automation, Mobile Robotics & Intelligent Systems* 2(3) (2008)
7. Lungu, V.: Artificial Emotion Simulation Model and Agent Architecture. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, Romania, pp. 716–721 (May 2011)
8. Robotin, R., Lupea, D., Lazea, G., Dobra, P.: Mobile Robots Path Planning with Heuristic Search. In: Proc. of the 18th Int. Conf. on Control Systems and Computer Science, Romania, pp. 391–396 (May 2011)
9. Brian, R., Hunt, R., Lipsman, L., Rosenberg, J.M.: A Guide to MATLAB for Beginners and Experienced Users. Cambridge University Press (2001) ISBN-I3: 978-0-521-00859-4
10. Matlab Guide Help, <http://www.mathworks.com/help/techdoc/ref/guide.html>
11. Barreto, J., Trigo, A., Menezes, P., Dias, J.: Kinematic and dynamic modeling of a six legged robot
12. Bensalem, S., Gallien, M., Ingrand, F., Kahloul, I.: Nguyen Thanh-Hung, Designing autonomous robots. *IEEE Robotics & Automation Magazine* 16 (March 2009)
13. Fahimi, F.: *Autonomous Robots: Modeling, Path Planning, and Control*. Springer (2008)
14. Knight, A.: *Basic of Matlab and Beyond*. CRC Press, LLC (2000) ISBN: 8493-2039-9
15. Marchand, P., Thomas, H.O.: *Graphics and GUIs with MATLAB*, 3rd edn. Chapman & Hall/CRC (2003) ISBN 1-58488-320-0
16. Sarjoun, S., Kantor, G., Maiwand, G., Alfred, A.: Inertial Navigation and Visual Line Following for a Dynamical Hexapod Robot. In: Proceedings of the 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems Las Vegas, Nevada (2003)

A Comparison of Adaptive Supervisory Switching Control Schemes for High Maneuverability Aircrafts

Andrei-Sorin Neamtu^{1,2} and Adrian-Mihail Stoica¹

¹ Faculty of Aerospace Engineering, Politehnica University of Bucharest, Romania

² National Aerospace Research Institute (Institutul National de Cercetari Aeronautice)
"Elie Carafoli" Bucharest, Romania

Abstract. Adaptive Supervisory Switching Control schemes work by introducing in the control scheme a supervisory unit which chooses, from a set of candidate controllers the one most suited for the current plant. There are two main classes of methods in this category. The first called Unfalsified Control Adaptive Supervisory Switching Control (UASSC) works by calculating for each candidate controller at discrete moments of time using the input/output data recorded up to that point a performance index and discarding from the candidate controller set those controllers which surpass a given threshold of this index. This process is called falsification. The second called Multi-Model Adaptive Switching Supervisory (MASSC), works by associating a dynamic nominal model with every candidate controller and comparing norms of sequences of estimation errors based on the various nominal models, as the candidate controller associated to the nominal model yielding the prediction norm of minimum magnitude is believed to be the most suitable one. Recently a new categories called Multi Model Unfalsified Adaptive Supervisory Switching Control (MMUASSC) was introduced. The schemes belonging to this category combine the advantages of both Unfalsified and Multi-Model Control techniques. In this paper we review the theory behind the control techniques belonging to the first (UASSC) and third (MMUASSC) of the categories above and adapt it to the case of controlling a fighter aircraft. We also provide a case study, where we compare these control schemes on a simulated fighter aircraft. Equation Chapter 1 Section 1

1 Introduction

Adaptive Control was first introduced in the 1950's in an attempt to alleviate some of the problems in controlling high speed fighter aircraft which often find themselves in conditions which are very hard to model due to their highly non-linear nature, or high number of uncertainties. Despite initial success Adaptive Control soon showed many deficiencies which have confined it solely to research studies. However, recently, methods have been devised that might make Adaptive Control a reality.

Over the last two decades a lot of research has been put in Adaptive Switching Supervisory Control (ASSC) (see [1], [2], [3], and [4]). ASSC is in fact an adaptive variant of classical gain scheduling, turned, by the use of a supervisory logic based on plant input/output recorded data, from an open loop switching mechanism to a closed loop one. A typical ASSC is depicted in Figure 1. A data driven “high-level unit” S , called *supervisor*, controls each plant G belonging to the given set \mathcal{G} of plant models by connecting an appropriate controller K from the set \mathcal{K} of candidate controllers. The supervisor decides if the currently switched-on controller works properly, and, in the negative case, it replaces it by another candidate controller. The scheduling task (when to substitute the acting controller) and the routing task (which controller to switch on) are carried out in real time by monitoring purely data-driven test functional [1]. The main current approaches to ASSC can be subdivided into two different groups: the first consists of the so called *Multi-Model* ASSC (MMASSC), wherein a dynamic nominal model is associated with every candidate controller, the second called *Unfalsified* ASSC (UASSC) [1], [4] wherein a switching logic that dispenses with the need for a-priori knowledge of the dynamic model is used. Both these methods have their advantages and disadvantages. Which will be highlighted below.

The main purpose of the current paper is to evaluate the performance provided by the Adaptive Supervisory Switching Control techniques in an adaptive control law for the stabilization of the short period dynamics of a fighter aircraft. In addition to UASSC and MMASSC another classes of techniques used in ASSC, called *Multi-Model Unfalsified Adaptive Supervisory switching Control – MMUASSC*, allowing the removal of the usual restrictive assumptions in the UASSC theory will be presented, extending thus the authors previous results presented in [5]. The paper is organized as follows. In Section 2 the UASSC, MMASSC and MMUASSC methods as well as the modifications required for its implementation in aviation are summarized. Section 3 outlines a simulation example run using the ADMIRE aircraft model. In the final section of the paper some concluding remarks are presented together with intended future developments.

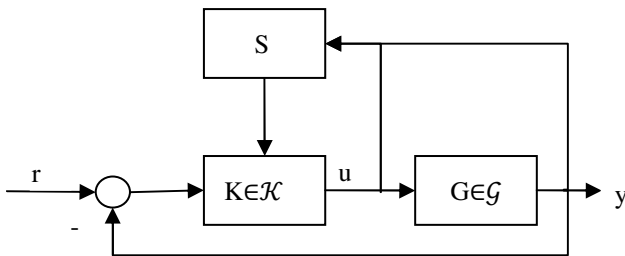


Fig. 1. Adaptive Supervisory Switching Control scheme

2 Adaptive Supervisory Switching Control

In this section a brief presentation of the three classes of methods used in Adaptive Supervisory Switching Control as have been developed so far will be given. As will be seen from the presentation, discarding all presumptions given by the name of the individual methods, the main difference between the three methods is in how one constructs the cost function by which the Supervisory Unit evaluates the suitability of each of the candidate controllers. Thus in *Unfalsified Adaptive Supervisory Switching Control (UASSC)* the cost function J_i for each candidate controller K_i is dependent on the input-output data and the expression describing the candidate controller:

$$(J_i)_{UASSC} = f(K_i, (u, y)).$$

In the case *Multi-Model Adaptive Supervisory Switching Control (MMASSC)* the cost function is a dependent on the expression which describes the model M_i associated to each controller K_i :

$$(J_i)_{MMASSC} = f(M_i, (u, y)).$$

While in *Multi-Model Unfalsified Adaptive Supervisory Switching Control (MMUASSC)* the cost function is built to take into account both the expression of the controller and of the model associated to it:

$$(J_i)_{MMUASSC} = f(M_i, K_i, (u, y)).$$

In the above equations (u, y) designates the vector containing the recorded inputs (u) and outputs (y) of the plant.

2.1 Unfalsified Adaptive Supervisory Switching Control (UASSC)

The main advantage of UASSC schemes as described by [4], is that they can select in finite time a final controller yielding a finite affine gain from the reference to the data, under the minimal conceivable requirement regarding the existence of a stabilizing candidate controller. This along with the fact that the plant does not need to be linear makes this schemes from the robustness point of view much better suited to aerospace applications than MASSC. Thus, the asymptotic stability properties of the latter are typically only guaranteed if the unknown plant is tightly approximated by at least one nominal model. However the main disadvantage of UASSC schemes used so far, as noted in [1] and [3], stems from the fact that they do not provide protection against the temporary insertion in the loop of destabilizing controllers, this leads to long transient times before the final stabilizing controller is switched on. In the examples provided in [4] where this method was introduced the supervisor needs about 70 seconds before finding the stabilizing controller, which wouldn't be convenient when trying to stabilize the short period longitudinal dynamics of an aircraft.

In the following some notations, preliminary results and main ideas of the UASSC concepts will be briefly presented (for more details, see [4], [3]).

Consider the following closed loop control system:

$$\begin{aligned} y(s) &= G(s)u(s) \\ u(s) &= K(s)(r(s) - y(s)) \end{aligned} \quad (1)$$

where $G(s)$ denotes the transfer function of the controlled plant, $K(s)$ stands for the controller, r is the reference signal, u and y are the control variable and the system output respectively. Though UASSC methods can be used on non-linear plants, linearity will be assumed in throughout the paper for simplicity and also the later case study is conducted using robust controllers designed based on linearization of aircraft dynamics.

It is assumed G belongs to a plant uncertainty set \mathcal{G} . The controller K belongs to a finite family \mathcal{K} of *linear time invariant* (LTI) controllers.

Definition 1. Given a signal $x(t)$, $t \geq 0$ is said that $x_\tau(t) = \begin{cases} x(t), & t \in [0, \tau] \\ 0, & \text{otherwise} \end{cases}$

represents a truncation of $x(t)$ with the truncated norm $\|x\|_\tau = \left(\int_0^\tau x^2(t) dt \right)^{\frac{1}{2}}$.

With the above definition the following slight generalization of input-output stability will be adopted throughout the paper [1], [3], [6]

Definition 2. A dynamic system with the input r and the output y is called stable, or the stability is unfalsified by the data (\mathbf{u}, \mathbf{y}) , if there exist $\alpha, \beta \geq 0$ such that $\|y\|_\tau \leq \alpha \|r\|_\tau + \beta$, $\forall \tau \geq 0$ and for all $r \in L_{2e}, L_{2e}$ denoting the space of all functions

with finite energy on any finite interval. Otherwise if $\sup_{\tau \geq 0, \|r\|_\tau \neq 0} \frac{\|y\|_\tau}{\|r\|_\tau} \rightarrow \infty$, it is said

that the stability of the system is falsified by the data (u, y)

The presence of the term $\beta \geq 0$ in the above definition is related to the situation where non-zero initial conditions, of the system, are taken into account. The next definition will be used in the following developments (see also [1], [3])

Definition 3. The adaptive control problem is feasible if, for every $\mathbf{G} \in \mathcal{G}$, there exists at least one controller $K \in \mathcal{K}$ such that the resulting system obtained by coupling K to G is stable and it accomplishes the performance objectives.

The unfalsified adaptive control techniques are essentially based on the so-called *fictitious reference signal* and on an associated *performance index* which allows choosing appropriate controllers, $K \in \mathcal{K}$, for which the problem is feasible [4]

Definition 4. Let the data (u, y) be the input and output measurements of a plant G over the time interval $[0, \tau]$. Then the fictitious reference signal \tilde{r}_K associated to a controller $K \in \mathcal{K}$ is the signal defined over $[0, \tau]$ that produces the same set of data (u, y) if K would be connected to G .

Note that the above definition requires the *invertibility* of K in which case the fictitious reference signal is given by $\tilde{r}_K = K^{-1}u + y$. This expression of \tilde{r}_K reveals another major constraint for K , namely it must be *minimum phase* since otherwise the fictitious reference \tilde{r}_K can be unbounded for $t \rightarrow \infty$. Some aspects concerning these constraints will be discussed below.

The performance index $J(K, u, y, \tau)$ is a positive defined function defined on $\mathcal{K} \times \mathcal{U} \times \mathcal{Y} \times \mathbb{R}_+$ where u and y are truncated on the interval $[0, \tau]$. It is defined according to the design specification of the controller K and it represents a measure of the performance provided by K on the time interval $[0, \tau]$.

Definition 5. A controller $K \in \mathcal{K}$ is called *falsified* at the time τ with respect to a given cost level $\gamma > 0$ by the data (u, y) measured on the time interval $[0, \tau]$ if $J(K, u, y, \tau) > \gamma$. Otherwise the controller K is called *unfalsified* by the measurements (u, y) on $[0, \tau]$.

According to the terminology used in [6] the set of all unfalsified controllers with the unfalsified cost level $\gamma > 0$ at time t stands for the *unfalsified controller set*.

The unfalsified adaptive controllers are not always safe, in the sense that some unfalsified destabilizing controllers inserted in the closed-loop can produce large signals for long intervals of time.

Definition 6. Consider the reference signal r and the measured set of data (u, y) obtained by a finite number of switches of controllers $K \in \mathcal{K}$, mapping $\begin{bmatrix} r(t) \\ y(t) \end{bmatrix}$ to $u(t)$ and denote by t_f the final switching time and by K_f the final controller. Then the pair (J, \mathcal{K}) is called *cost-detectable* if the following assertions are equivalent:

- a) $J(K_f, u, y, \tau)$ is monotone increasing and bounded for $\tau \rightarrow \infty$;
- b) The closed loop system in Figure 1 with K_f is unfalsified by the data (u, y) when $\tau \rightarrow \infty$.

In [3], [4], [6] the following performance index is considered:

$$J(K, u, y, \tau) = \frac{\|w_1 * (y - \tilde{r})\|_{L_2[0, \tau]}^2 + \|w_2 * u\|_{L_2[0, \tau]}^2}{\|\tilde{r}\|_{L_2[0, \tau]}^2} \tag{2}$$

where $*$ denotes convolution and $w_1(t)$ and $w_2(t)$ denote dynamic weighting functions used for determining controllers K as solutions of the mixed sensitivity problem

$$\left\| \begin{bmatrix} W_1 S \\ W_2 K S \end{bmatrix} \right\|_{\infty} \leq \gamma$$

$S \doteq (I + GK)^{-1}$ denoting the sensitivity function and $\|\cdot\|_{\infty}$ representing the H_{∞} norm of the system (\cdot) (for more details see [7])

A typical Unfalsified Adaptive Control Scheme is presented in Figure 2, while Figure 3 presents the algorithm by which controllers are chosen. In [4] an example is given that shows the strength of Unfalsified Control when used correctly even with the constraints mentioned above concerning the invertibility and minimum phase properties of $K \in \mathcal{K}$. However such limitations would make Unfalsified Control unusable for aerospace applications where often the controllers are so complex that even if they are invertible they might not be minimum phase. Fortunately as shown in [3] these constraints can be removed considering the coprime factorization of the controllers $K \in \mathcal{K}$. A similar idea is used in a discrete-time version in [1].

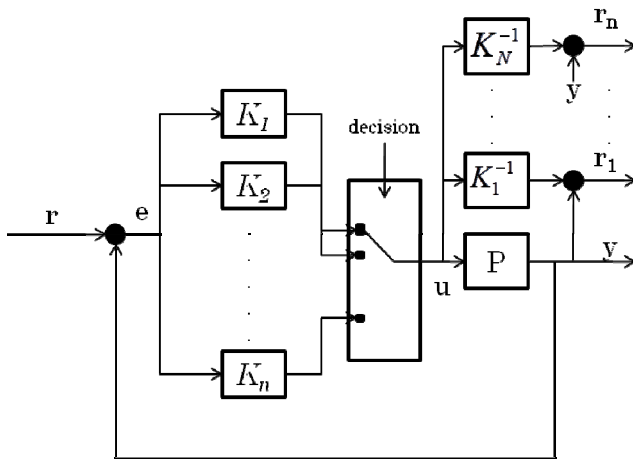


Fig. 2. Unfalsified Adaptive Supervisory Switching Control Scheme

Definition 7. The ordered pair (U, V) , with $U, V \in RH_{\infty}$, where RH_{∞} denotes the space of all asymptotically stable transfer function matrices, is called a left-coprime factorization of the transfer function G if:

- 1) V is square and invertible;
- 2) $G = V^{-1}U$.

Moreover if $VV^* + UU^* = I$ where $V^*(s) = V^T(-s)$ and $U^*(s) = U^T(-s)$ denote the adjoints of V and U respectively, then the pair (U, V) is said to be a normalized left-comprime factorization of $G(s)$.

A computation method to determine left comprime factorization of a given $G(s)$ may be found for instance in [8].

The closed loop system with $K=V^{-1}U$ is shown in Figure 4.

The above configuration can be alternatively implemented using the so-called “observer-form” configuration (see also [3] and their references) in Figure 5, where direct algebraic computations show that the new reference signal \tilde{v} , which generates the data set (u, y) , is given by:

$$\tilde{v} = Vu + Uy \tag{3}$$

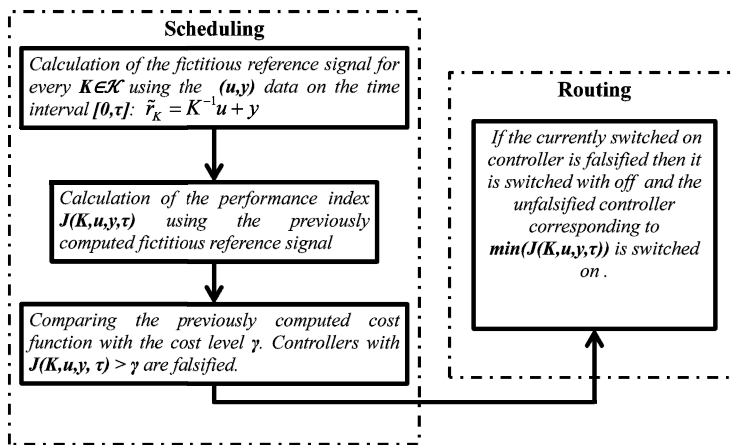


Fig. 3. Falsification algorithm

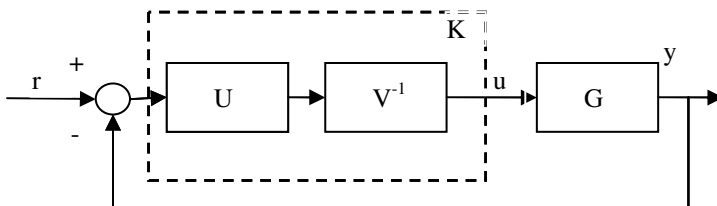


Fig. 4. Closed loop control system with $K=V^{-1}U$

Therefore the computation of the new fictitious reference \tilde{v} does not require an invertibility condition; moreover when u and y are bounded, \tilde{v} is bounded to, since U and V are stable. The performance index will be determined as in (2) replacing the fictitious reference signal \tilde{r} by \tilde{v} .

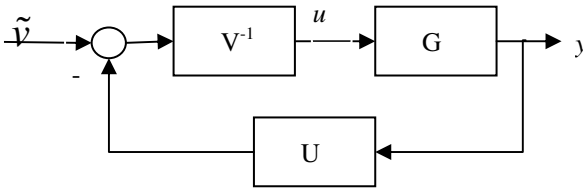


Fig. 5. “Observer form” configuration

Most literature on the subject of Unfalsified Control recommends using some form of parameterization to represent the candidate controller set. However the parameterizations given in [4] and [6], for example, are of simple controllers suitable for plants less complex than aircraft or for a limited envelope, as is the case in controlling a missile. To be representative for aerospace applications the candidate controller set would have to include some form of robust controllers, covering a large envelope, which have much higher complexity. The authors, therefore, chose, for the application considered in the next section, the following polytopic representation of the plant family \mathcal{G} :

$$\mathcal{G} \doteq \left\{ G(s) \mid G(s) = \lambda_1 G_1(s) + \dots + \lambda_n G_n(s), \sum_{i=1}^n \lambda_i = 1; \lambda_i \geq 0, i = 1 \dots n \right\}$$

where $G_i(s), i = 1, 2 \dots n$ are known transfer matrices corresponding to “ n ” nominal flight conditions.

For each $G_i(s)$ one determines via the mixed-sensitivity design method mentioned in the previous section, the controller $K_i(s) = V^{-1}(s)U_i(s)$ where the coprime factors $V_i(s)$ and $U_i(s)$ are stable. Then the following parameterization of the controller set is defined:

$$\mathcal{K} \doteq \left\{ K(s) \mid K(s) = \sum_{i=1}^n \lambda_i K_i, \sum_{i=1}^n \lambda_i = 1; \lambda_i \geq 0 \right\}$$

Based on the left coprime factorization of $K_i(s), i = 1, \dots, n$ one obtains:

$$K_\lambda(s) = \sum_{i=1}^n \lambda_i V_i^{-1}(s) U_i(s) = V_1^{-1} \left(\lambda_1 U_1 + \sum_{i=2}^n \lambda_i V_i V_1^{-1} U_i \right) = V_1^{-1} U_\lambda \quad (4)$$

where the following notation has been introduced: $U_\lambda \triangleq \lambda_1 U_1 + \sum_{i=2}^n \lambda_i V_i V_1^{-1} U_i$.

From (4) it follows that the fictitious reference \tilde{z} is in fact a function of λ and so is the performance index (2). Using the collected data set (u, y) on the interval $[0, \tau]$ one can determine the optimal unfalsified controller $K_{\lambda^*}(s)$ of form (4) with

$$\lambda^* = \operatorname{argmin} J(K, u, y, \tau),$$

subjected to $\sum_{i=1}^n \lambda_i = 1; \lambda_i \geq 0$. The stability of the coprime factors V_i and U_i , $i = 1 \dots n$ ensures the cost-detectability property of the pair (J, \mathcal{K}) .

This method allows for a large set of candidate controllers to be obtained by interpolating a limited number of pre-computed controllers.

2.2 Multi-Model Adaptive Supervisory Switching Control (MMASSC)

Multi Model Adaptive Supervisory Switching control (MMASSC) works by comparing norms of sequences of estimation errors based on the various nominal models, as the candidate controller associated to the nominal model yielding the prediction norm of minimum magnitude is believed to be the most suitable one. The main advantage is the fact that transient times before finding a stabilizing controller tend to be small. However this can be achieved only by using a very dense model distribution. If this condition is not enforced neither convergence to a final controller, nor boundness can be guaranteed.

In the following section we will briefly discuss MMASSC for the case of a class of discrete time controllers and models associated to them. For more details the reader is referred to [9].

We define a set of controllers as follows:

$$K_i : R_i(d)\delta u(t) = S_i(d)[r(t) - y(t)], h \in \mathbb{N} \cap [1, N]$$

where $S_i(d)$ and $R_i(d)$ denote polynomials with strictly Schur greatest common divisor and $R_i(d)$ monic. To each controller in the candidate controller set we associate the model for which it was designed:

$$M_h : A_h(d)y(t) = B_h(d)\delta u(t), h \in \mathbb{N} \cap [1, N]$$

$A_i(d)$ and $B_i(d)$, denote polynomials with strictly Schur greatest common divisor, d is the unit backward shift operator, $A_i(1)=0$, $A_i(d)$ monic. In both of the above equations $t \in \mathbb{Z}_+, \mathbb{Z}_+ = \{0, 1, 2, \dots\}$, input-increment $\delta u(t) = u(t) - u(t-1) \in \mathbb{R}$, output $y(t) \in \mathbb{R}$.

The unit backward shift operator d in the above equations is equal to z^{-1} . It is preferred that the transfer transform the expressions of the transfer functions from the usual z to the backward shift operator so as to pass from the frequency domain to the time domain and be able to use the input-output data which is recorded at discrete time intervals.

One can define the plant *output prediction* at time t based on model M_h with input-output data collected when controller K_k is the loop as follows:

$$\hat{y}_k(t, h) = [1 - A_h(d)] y_k(t) + B_h \delta u_k(t) \quad (5)$$

where $y_k(t)$ and $\delta u_k(t)$ designated the input-output data collected when the „ k ”-th controller is the loop and $\hat{y}_k(t, h)$ represents the estimation of the output if controller K_h would be in the loop, given input-output data collected with controller K_k in the loop.

A *prediction error based on M_h given the input – output data collected with the controller K_k in the loop* can now be defined:

$$\varepsilon_k(t, h) = y_k(t) - \hat{y}_k(t, h) = A_h(d) y_k(t) - B_h \delta u_k(t) \quad (6)$$

The *mean square prediction error*:

$$p(h/k) := E[\varepsilon_k^2(t, h)] \quad (7)$$

where E denotes ensemble average, can be theoretically used to construct a cost function but usually a time-average evaluation of it is preferred. Such an evaluation is the actual cost function by which the supervisor functions:

$$\begin{cases} J_t(h/k) = \lambda \pi_{t-1}(h/k) + (1 - \lambda) \varepsilon_k^2(t, h) \\ J_0(h/k) = 0 \end{cases} \quad (8)$$

where $J_t(h/k)$ is the approximation of (7), and $\lambda \in [0, 1)$ is according to the terminology from [9] a forgetting factor, $t \in Z_1 := \{1, 2, \dots\}$ and $t=0$ is the moment at which the current controller became operational.

Thus the controller considered optimal for the current conditions is the one minimizing the cost function $J_t(h/k)$ from expression (8):

$$h(k) = \arg \min_{k \in N} J_t(h/k)$$

As is shown in detail in [9] *MMASSC* has problems associated with its use of models to estimate the suitability of each candidate controller. Namely in the case in which none of the models closely approximate the current plant the algorithm cannot return a suitable controller.

2.3 Multi-Model Unfalsified Adaptive Supervisory Switching Control (MMUASSC)

To alleviate the problems associated with both Multi-Model Adaptive Supervisory Switching Control and also the disadvantages of “classical unfalsified control”, in [1] a scheme called Multi Model Unfalsified Adaptive Switching Supervisory Control, that combines the advantages of both methods (low transient times for MASSC and asymptotic stability for UASSC), was proposed.

In this section we will give a brief description of the differences between MMUASSC and UASSC. For further details on how the theory was developed and on its background we refer the reader to [1] and [9].

So far MMUASSC theory has been developed only for use with discrete time models and controllers. To simplify the notations, the single-input, single-output case will be considered in the following. We redefine the model/controller paring (M_i/K_i) from above as:

$$M_i = \frac{B_i(d)}{A_i(d)}, \quad i \in \bar{N}$$

$$K_i = \frac{S_i(d)}{R_i(d)}, \quad i \in \bar{N}$$

$$y(t) = M_i(d)\delta u(t)$$

$$\delta u(t) = K_i(d)(r(t) - y(t))$$

Where $t \in \mathbb{Z}_+, \mathbb{Z}_+ = \{0, 1, 2, \dots\}$, input-increment $\delta u(t) = u(t) - u(t-1) \in \mathbb{R}$, output $y(t) \in \mathbb{R}$, $A_i(d)$ and $B_i(d)$, denote polynomials with strictly Schur greatest common divisor, d is the unit backward shift operator, $A_i(1)=0$, $A_i(d)$ monic, similarly $S_i(d)$ and $R_i(d)$ denote polynomials with strictly Schur greatest common divisor and $R_i(d)$ monic and r denotes the reference to be tracked. The controlled action is realized via shared state multi-controller implementations [10]. An example presented in [1] is given bellow. As a shared state one uses the vector:

$$\xi(t) := \begin{bmatrix} \epsilon(t-1) \\ \vdots \\ \epsilon(t-p) \\ \delta u(t-1) \\ \vdots \\ \delta u(t-p) \end{bmatrix}$$

where $\epsilon = r - y$, $p = \max\{degS_i, degR_i, i \in \bar{N}\}$ thus the output of K_i at t is:

$$K_i(\epsilon) = [s_{i1} \dots s_{ip} - r_{i1} \dots - r_{ip}] \xi(t) + s_{io} \epsilon(t)$$

with s_{ij} and r_{ik} , $i \in \bar{N}$, $j+1 \in \overline{p+1}$ and $k \in \bar{p}$ the coefficients of the polynomials $S_i(d)$ and $R_i(d)$, respectively and $\xi(t)$ is the shared state vector. As shown in [1]

the uncertain plant $G = \frac{B(d)}{A(d)}$ can be represented, as in Fig. 6 in a coprime factor perturbed form:

$$P: \begin{cases} A_i(d)y(t) = B_i(d)\delta u(t) + e_i(t) \\ e_i(t) = \tilde{A}_i(d)y(t) + \tilde{B}_i(d)\delta u(t) \end{cases}$$

with $\tilde{A}_i(d) = A_i(d) - A(d)$, $\tilde{B}_i(d) = B(d) - B_i(d)$, and e_i representing the equation error. One, finds the transfer matrix $H_{zei}(d)$ from e_i to $z = \begin{bmatrix} \delta u(t) \\ y(t) \end{bmatrix}$:

$$\begin{aligned} H_{ze_i}(d) = Q_i(d) &= \frac{\begin{bmatrix} -K_1(d) \\ 1 \end{bmatrix}}{(1 + M_i(d)C_i(d))A_i(d)} \\ &= \frac{\begin{bmatrix} -S_1(d) \\ R_i(d) \end{bmatrix}}{A_i(d)R_i(d) + B_i(d)S(d)} \end{aligned}$$

and $H_{eiz}(d)$ the transfer matrix from z to e_i :

$$H_{e_i z}(d) = \tilde{L}_i(d) = \begin{bmatrix} \tilde{B}_i(d) \\ \tilde{A}_i(d) \end{bmatrix}$$

The following test functional derived from the stability condition $\|Q_i\|_\infty \|L_i\|_\infty < 1$, given by the Small Gain Theorem, can be used in these conditions:

$$J_i(t) = \max \Lambda_i^{t-1} \Lambda_i^{t-1}(t) = \begin{cases} 0, \text{ if } \|z^t\| = 0 \\ \left\| \frac{z_i^t}{z_{i0}^t} \right\|, \text{ otherwise} \end{cases} \tag{9}$$

with $z_{i0} = z - \tilde{z}_{i0}$, and $\tilde{z}_{i0}(t) = T_i(d)z_i(t)$, where $T_i(d)$ coincides with the generalized system matrix of (G/K_i) :

$$T_i(d) = (1 + GK_i)^{-1} \begin{bmatrix} -GK_i & K_i \\ G & -1 \end{bmatrix}$$

As further demonstrated in [1], this test functional ensures cost detectability (see Definition 6) of the system, a finite number of switches until a final controller is selected and that no destabilizing controllers will be inserted in the loop.

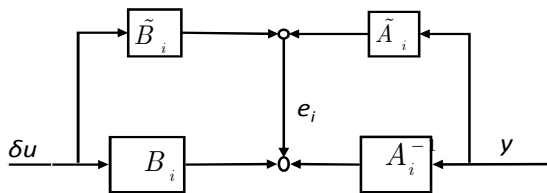


Fig. 6. Coprime factor model error representation

To implement this cost function in a practical case a few artifices have to be made to ensure that the MMUASSC algorithm works with controllers that are not stable causally invertible, thus solving another problem of UASSC. The cost function is implementable in the following form:

$$J_i(t) = \begin{cases} \Lambda_i(t-1), & 0 \leq t \leq t_0 \\ \max \Lambda_i|_{t_0}^{t-1}, & \text{elsewhere} \end{cases}$$

where $\max \Lambda_i|_{t_0}^{t-1} = \max \{ \Lambda_i(\tau), \tau = t_0, \dots, t-1 \}$, and

$$\Lambda_i^{\frac{1}{2}}(t) = \begin{cases} |\tilde{\zeta}_i(t)| \bar{J}_i^{-\frac{1}{2}}, & 0 \leq t \leq t_0 \\ \frac{\|\tilde{\zeta}_i|_{t_0}^t\|}{\|z^{t_0-1} + (z - \tilde{\zeta}_i)|_{t_0}^t\|}, & \text{elsewhere} \end{cases} \tag{10}$$

where $\tilde{\zeta}_i(t) = (Q_i(d)L_i(d))_0 z(t) = (Q_i(d))_0 \epsilon_{i0}(t)$ and \bar{J}_i is a constant depending on the magnitude of the mixed sensitivity of (M_i/K_i) .

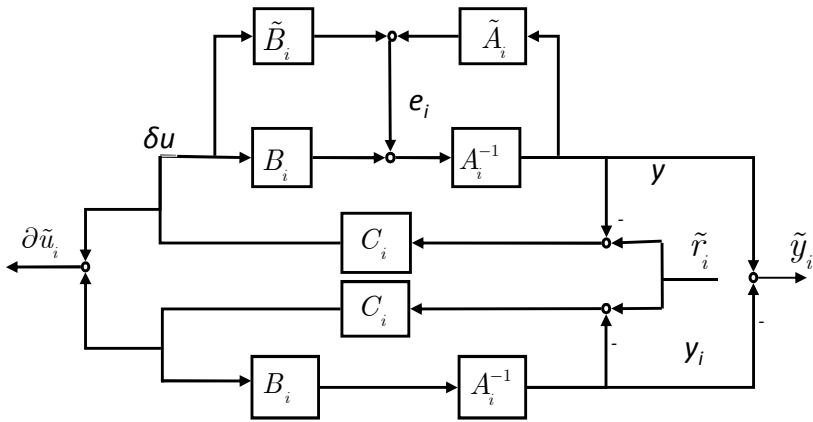


Fig. 7. Details of multi-model UASSC

For more details on how these cost function was developed the reader is referred to [1].

3 Case Study

For the case study the ADMIRE aircraft model, provided as freeware by the Swedish Research Administration, has been used.

The aim of the case study is to compare UASSC and MUASSC in controlling the airplane short-period dynamics. Therefore only the equations containing the angle of attack and pitch rate were used. Also to simplify the study just one of

the control inputs was considered out of the three available for maneuvers in the longitudinal plane. Four schemes were considered:

- 1) the first was a UASSC schemes with four candidate controllers in the candidate controller set
- 2) the second was a UASSC scheme that extended de candidate controller set to 286 controllers by interpolation using the polytopic representation presented above.
- 3) The third scheme was a Mmassc scheme using the four controllers with thier associated models
- 4) The forthd scheme was a MMUASSC scheme, again using the four controllers with their associated models.

First the controllers were designed each corresponding to a different modeled flight condition. The design was carried out on linearizations of the short period dynamics of the ADMIRE model, in the following for flight cases:

- Flight Condition 1: Mach 0.4, altitude 4500m;
- Flight Condition 2: Mach 0.6, altitude 1500m;
- Flight Condition 3: Mach 0.85, altitude 5500m;
- Flight Condition 4: Mach 0.9, altitude 2000m.

The controllers for these flight conditions were designed in the Matlab software package using the Weighted Mixed Sensitivity Criteria. As weighting functions

we have used $W_1(s) = \frac{0.5s + 5}{s + 0.15}$ and $W_2(s) = \frac{s + 10}{0.05s + 5}$.

The mixed sensitivity problem was slightly altered to require the following minimization

$$\left\| \begin{bmatrix} W_1 S \\ W_2 GKS \end{bmatrix} \right\|_{\infty} \leq \gamma$$

as such, and also because of the implementation of the modification from [3], the performance specification for the UASSC was changed from (2) to the following:

$$J(K, u, y, \tau) = \frac{\|w_1 * (y - \tilde{v})\|_{L_2[0, \tau]}^2 + \|w_2 * y\|_{L_2[0, \tau]}^2}{\|\tilde{v}\|_{L_2[0, \tau]}^2} \quad (11)$$

where w_1 and w_2 are the impulse responses of the weighting transfer functions $W_1(s)$ and $W_2(s)$ used in the design of the four controllers ($K_1 \dots K_4$), corresponding to the four flight conditions. Thus we impose on the falsification algorithm that only those controllers meeting the same design criterions as the

pre-designed controllers be unfalsified. The “*” symbol means convolution, \tilde{v} represents the fictitious reference signal as defined by (3), y is the plant output signal, and u is the control signal.

This translates into the following cost function.

$$J = \frac{\|\tilde{w}_1\|_{L_2[(j-1)k, jk]}^2 + \|\tilde{w}_2\|_{[(j-1)k, jk]}^2}{\|v\|_{[(j-1)k, jk]}^2}$$

where k is the *dwell interval* (the minimum time for which a controller is in the loop, and during which measurements of u and y are performed), j represents the index of the current dwell interval, \tilde{w}_1 is a vector containing the responses of the W_1 transfer function to the inputs contained in the vector $(y-v)$, \tilde{w}_2 is a vector containing the responses of the W_2 transfer function to the inputs contained in the vector y and v is the vector containing the fictitious reference signals calculated over the $[0, (j-1)k]$ using (3).

For the UASSC scheme with interpolation the cost function J is updated at the end of each interval equal to the dwell interval (in this case the value was 1 second). The algorithm then switches-on the controller with the lowest value of J . For the polytopic representation in this scheme, a precision of one digit was considered for the λ coefficients. This yielded 286 candidate controllers. The possible values for the coefficients were stored in a vector which represented the set of candidate controllers, and were calculated in the initialization phase of the study.

For the Mmassc and MMUASSC schemes the controllers and their associated models had to be discretized first, and connected in loop with the continuous time plant. The sampling time used was 0.01 seconds. The UASSC scheme without interpolation was run in the exact same conditions to obtain the best comparison of the relative performance of the three schemes.

The case study was conducted in Matlab and Simulink: a plant obtained by linearizing the Admirer model was connected in a feedback-loop with a random candidate controller. The simulations were run for 40 seconds. The response of the closed loop system can be observed in figures 8 through 11, along with the reference. As can be seen all four schemes stabilize the plant quickly. However as can be seen the Mmassc scheme as anticipated has problems because none of the associated models closely approximates the actual plant. In figure 8, which shows the input/output of the UASSC scheme with interpolation, vertical black lines represent the moments at which the system switched. Figures 12, 13 and 14 show which of the four controllers was in the loop for the UASSC scheme without interpolation, figure 12, for the Mmassc scheme, figure 13, respectively for the MMUASSC in figure 14. As can be seen all schemes switched the controller fast, but while the UASSC and MMUASSC schemes needed only one switching the Mmassc needed two. This further illustrates the better performance of switching algorithms based on Unfalsified theory. Of the two UASSC schemes the one using interpolation offered a better reference tracking, mostly because the controllers were continuous time.

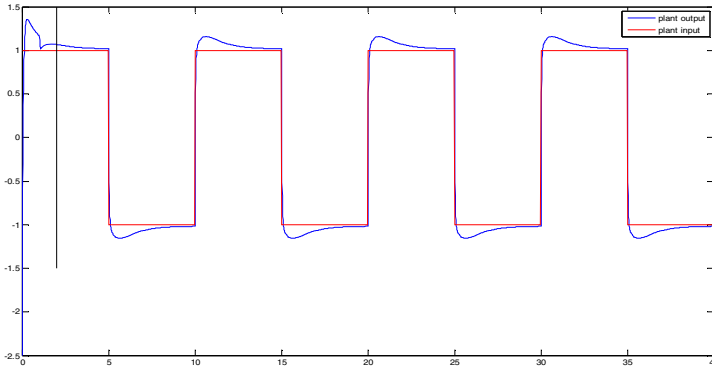


Fig. 8. Response of the closed-loop system with the plant linearized at Mach 0.75, altitude 5500 and an angle of attack of 12 degrees. UASSC scheme with interpolation. Dwell interval 1 seconds.

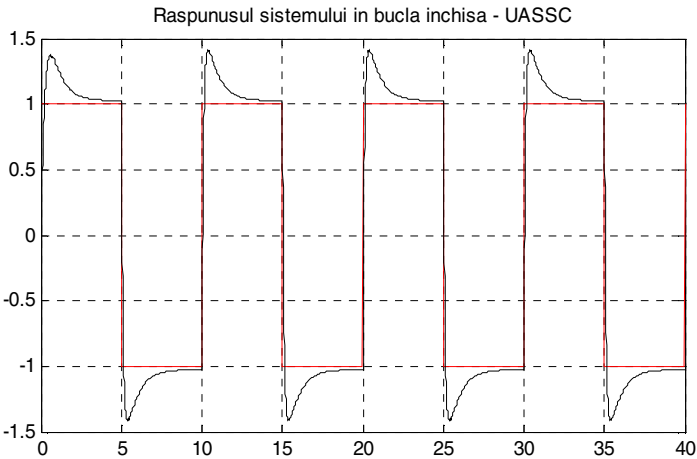


Fig. 9. Response of the closed-loop system with the plant linearized at Mach 0.75, altitude 5500 and an angle of attack of 12 degrees. UASSC scheme

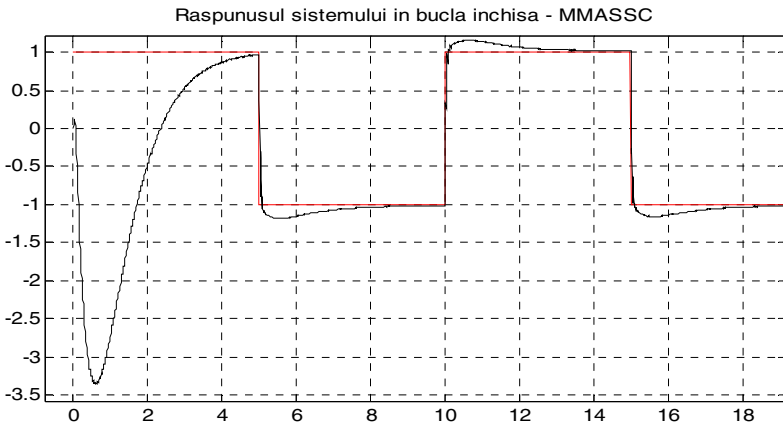


Fig. 10. Response of the closed-loop system with the plant liniarized at Mach 0.75, altitude 5500 and an angle of attack of 12 degrees. MMASCC scheme.

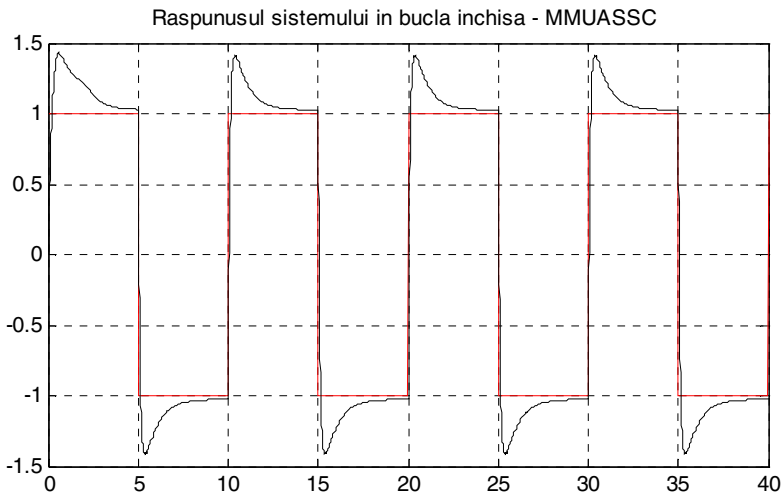


Fig. 11. Response of the closed-loop system with the plant liniarized at Mach 0.75, altitude 5500 and an angle of attack of 12 degrees. MMUASCC scheme.

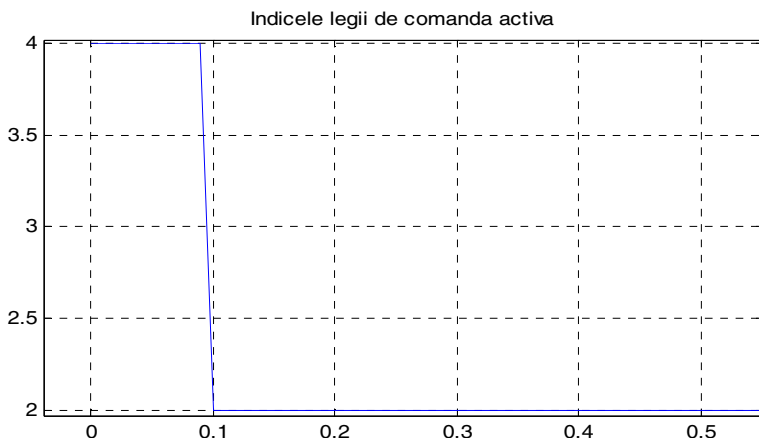


Fig. 12. Active candidate controllers for the UASSC scheme

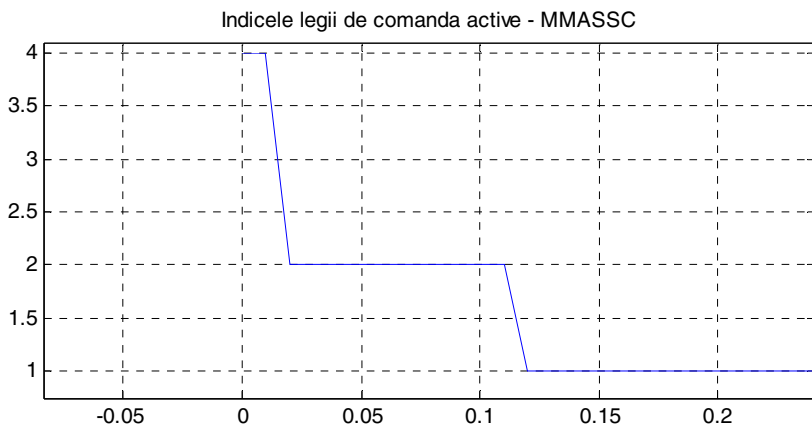


Fig. 13. Active candidate controllers for the MMUASSC scheme

We conducted tests with several plants, although for lack of space we introduced in this paper the results from a single one. It is worth mentioning here that in simulations ran with one of the models as plant, in several cases, the UASSC schemes failed to switch on the controller tuned to that model instead choosing another controller. MMUASSC on the other hand never failed to correctly identify the model and switch on the appropriate controller. This behavior of the UASSC schemes foreshadows the possibility of introducing a destabilizing controller in the loop as illustrated in [3].

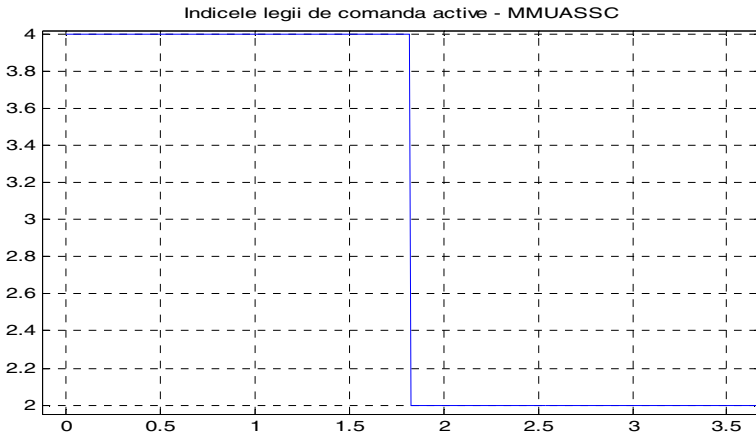


Fig. 14. Active candidate controllers for the MMUASSC scheme

4 Conclusions

All four of the methods tested in this paper worked well with the considered plant. Out of the three schemes, MMUASC offered the best performance and accuracy in identifying the appropriate controller. This coupled with its intrinsic protection against switching on a destabilizing controller makes this method the most promising of the three.

Further research will be conducted by the authors to address the problems illustrated above. Namely the investigation of behavior in the multivariable case and also research in the possibility of implementing a form of controller interpolation for the MMUASSC scheme.

Acknowledgment. This paper is financed by grant POSDRU 7173 through contract POSDRU/6/1.5/S/19 and by CNCISIS Grant no. 1721 also, we extend our gratitude to professor Edoardo Mosca and his team at Università di Firenze especially Pietro Tesi and Daniele Mari for enlightening discussions and useful suggestions.

References

- [1] Baldi, S., Battistelli, G., Mosca, E.: Pietro Tesi Dipartimento Sistemi e Informatica, DSI - Università di Firenze, Via S. Marta 3, 50139 Firenze, Italy, Multi-model unfalsified adaptive switching supervisory control. *Automatica* 46, 249–259 (2010)
- [2] Brugarolas, P.B., Fromion, V., Safonov, M.G.: Robust Switching Missile Autopilot. In: Proc. American Control Conference, Philadelphia, PA, June 24–26 (1998)
- [3] Dehghani, A., Anderson, B.D.O., Lanzon, A.: Unfalsified Adaptive Control: A New Controller Implementation and Some Remarks. In: Proceedings of the European Control Conference 2007 Kos, Greece, July 2–5 (2007)

- [4] Safonov, M.G., Tsao, T.C.: The unfalsified control concept and learning. *IEEE Trans. Autom. Control* AC-42(6), 843–847 (1997)
- [5] Neamtusi, A.S., Stoica, A.M.: Adaptive Robust Design of Unmanned Combat Air Vehicle Automatic Control System. In: *27th Congress of the International Council of the Aeronautical Sciences*, Nice, France, pp. 19–25 (2010)
- [6] Wang, R., Paul, A., Stefanovic, M., Safonov, M.G.: Cost detectability and stability of adaptive control systems. *Int. J. Robust Nonlinear Control* 17, 549–561 (2007)
- [7] Chiang, R.Y., Safonov, M.G.: *Robust-Control Toolbox*, Mathworks, South Natick, MA (1988)
- [8] MacFarlane, C., Glover, K.: *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*. Springer (1990)
- [9] Mosca, E., Agnoloni, T.: Inference of candidate loop performance and data filtering for switching supervisory control. *Automatica* 37, 527–534 (2001)
- [10] Morse, A.S.: Control using logic-based switching. In: Isidori, A. (ed.) *Trends in Control: A European Perspective*, pp. 69–113. Springer, London (1995)

Box-Counting and Multifractal Analysis in Neuronal and Glial Classification

Herbert F. Jelinek¹, Nebojša T. Milošević^{2,*}, Audrey Karperien¹,
and Bojana Krstonošić³

¹ Centre for Research in Complex Systems and School of Community Health, Charles Sturt
University, Albury, Australia

hjelinek@csu.edu.au, akarpe01@postoffice.csu.edu.au

² Department of Biophysics, Medical faculty, University of Belgrade, KCS-Institute of
biophysics pp. 122, 11129 Belgrade102, Serbia

mtn@med.bg.ac.rs

³ Department of Anatomy, Medical Faculty, University of Novi Sad, Hajduk Veljkova 3
21000 Novi Sad, Serbia

bojana.80@eunet.rs

Abstract. Fractal analysis in the neurosciences has advanced over the past twenty years. The fractal dimension, besides its ability to discriminate among different cell types, can work as a reliable parameter in cell classification. A qualitative analysis of the morphology of neurons and glia cell types involves a detailed description of the structure and features of cells, and accordingly, their classification into defined classes and types. This paper outlines how fractal analysis can be used for further quantitative classification of these cell types using box-counting and multifractal analysis.

Keywords: Box dimension, Cell classification, Human, Fractal analysis; Multifractal, Microglia, Aspinous, Neostriatum.

1 Introduction

Pattern analysis plays an important role in many fields of inquiry including neuroscience. One type of pattern analysis that has been used to investigate phenomena relevant to the neuroscientist is fractal analysis [1-3], which has proven especially valuable for investigating various cell types found within the spinal cord, brainstem, cerebellum, and cerebral hemispheres [1, 4-7]. In essence, fractal analysis of cellular morphology involves examining the scaling inherent in patterns or datasets extracted from digital images of cells, in order to assign a fractal dimension, a number that is a statistical index of complexity having no units [8-10]. If the scaling within the dataset from a cell is consistent, analogous to the ideal self-similarity of mathematical monofractals, it is appropriate to assign a

* Corresponding author.

single, global fractal dimension to the dataset [8,11]; but if the scaling varies over the dataset it may be more appropriate to assess the pattern as a multifractal and determine a range of values [12,13]. Multifractal analysis has been applied in neuroscience, but the multifractality of biological images remains a rather contentious subject [2,3,4,14,15].

In this paper, we report the use of box-counting methods to investigate digital images of both neuronal cells from the adult human neostriatum and microglial cells from the human brain and spinal cord. Following statistical analysis of two box dimensions of cells from the neostriatum and multifractal properties of microglial cells, the significance of our results in terms of these cell classifications are also discussed.

2 Materials

2.1 *Cells from the Human Putamen*

The material used in this study was collected during 2008 and 2009 by the Center for Forensic Medicine, Toxicology and Molecular Genetics at the Clinical Center of Vojvodina (Serbia). Ten male human brains were obtained from medico-legal forensic autopsies of adult bodies (ranging from 32 to 48 years of age), with no prior history of neurological diseases, or major liver, renal or cardiovascular dysfunctions. All material was collected with the approval of the Ethics Committee of the University of Novi Sad, Faculty of Medicine (Serbia) and the research was performed in accordance with the ethical standards defined by the 1964 Declaration of Helsinki and all subsequent revisions. The experiments were undertaken with the understanding and written informed consents of the families of the deceased, whose anonymity was preserved. For detailed histological procedure the reader is referred to [16,17].

After analyzing the microscopic images of Golgi impregnated neurons of the putamen, we initially divided neurons into two major groups: the spiny and aspiny cells, according to the presence of spines. In our sample of 301 collected neurons, 35.22% (106 cells) were aspiny cells. We classified our sample of cells, according to a previously described scheme, into three subgroups according to the number of primary dendrites, their branching order, length and course of dendrites and dendritic field density: into type 1, type 2 and type 3 (Fig. 1). Type 1 neurons were easily recognized by two to six primary dendrites which were extended and gave off a small number of side branches. From four to eighteen primary dendrites were abundantly branched close to the soma of type 2 neurons. They formed a relatively small, but very dense dendritic domain. From the body of type 3 neurons emerged three to eight relatively thick and short dendrites. Dendrites of type 3 neurons branched repeatedly close to the soma and formed a very small but, also, a very dense dendritic arbor. Our sample of aspiny cells consisted of 16 type 1 cells, 42 type 2 cells and 48 type 3 cells. Their representative images are shown in Fig. 1.



Fig. 1. Representative images of three cell types in the human putamen: type 1, type 2 and type 3 cells

2.2 Microglial Cells

This part of the material consisted of published images from central nervous system slides of post-mortem human and rat tissue. Detailed explanation of the histology and staining procedures can be found in [18,19].

Microglia cells can be classified into three subclasses with respect to their activation stage associated with their role in the development, maintenance and pathology [11,20] of the central nervous system (Fig. 2). Cells from the first class are resting or *ramified* cells (Fig. 2A), which typically perform sentinel functions. The second class are *bushy* and *hypertrophied* or *intermediate* cells (Fig. 2B), which usually are in the incipient stages of responding to noxious stimuli or returning to a resting state, whilst the third class are *activated* cells (Fig. 2C), which are generally in the macrophage stage.

2.3 Image Acquisition

Each section of the adult human neostriatum was classified using the “Leica DC 100” (Leica Microsystem Wetzlar GmbH), at a magnification of 400x and all impregnated neurons were recorded and transformed into digital images with the Digital Camera Systems software package (Leica Microsystem Wetzlar Ltd., Heerbrugg). Histological sections of each impregnated neuron were divided into twenty optical sections, and depending on the somal size and dendritic arborization, each nerve cell was represented by four to twenty focal planes [16]. The digital images of these focal planes were loaded into ImageJ (National Institute of Health, Bethesda, <http://rsbweb.nih.gov/ij>). Using the ‘ZProject’ command they were projected onto an image stack along the axis perpendicular to the image plane. Finally, using appropriate subroutines of the same software, a binary, outline and skeletonized image of the neurons was obtained [16].

Microglial cell images were converted to digital images by scanning the figures using a Microtek scanner and Adobe Photoshop LE (V5.0) [21]. Each cell was then converted to a greyscale image, the background subtracted using automatic filtering, and the resulting image converted to a binary contour. All necessary data was obtained using the public domain program ImageJ and the plug-in, FracLac 2.0 (<http://rsbweb.nih.gov/ij/plugins/fractalac/FLHelp/Introduction.htm>) written in our laboratory for analyzing morphological complexity of biological cells [11].

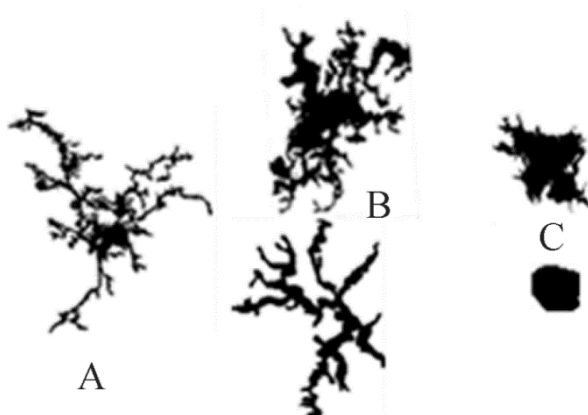


Fig. 2. Silhouettes illustrating typical microglial morphologies: ramified (A), intermediate (B) and activated (C) cells

2.4 Box-Count Analysis

The global fractal dimensions of the digitized images were investigated after each image was subjected to the box-counting algorithm incorporated in the Image J software. When the box-counting method is applied to neuronal images, the fractal dimension (precisely, the box dimension, D_B) depends on the image presentation. Box-counting is generally implemented within digital image analysis software to analyze binary (i.e., black and white) images. While the D_B of the outline (cell contour) of cells defines the irregularity in the shape of the neurons [2], the same parameter for skeletonized representations of the neurons indicates the degree of dendritic aberrations from a straight lines, in essence their tortuosity [22].

Briefly, the box-counting method consists of covering an object with sets of squares. Each set is characterized by the size of the square's edge, r . The number of squares $N(r)$ necessary to cover the object is presented as a function of r . The D_B is determined by the absolute value of the slope S of $\log N(r)/\log r$ [23,24].

The method was initially performed on outlines of the neurons and the corresponding $D_{B(\text{out})}$ calculated. Then, from the same binary image the cell body was removed digitally and the remaining dendritic tree skeletonized reducing the dendritic width to a single pixel. The skeletonized dendritic tree was processed by the same method, and the $D_{B(\text{skel})}$ obtained [16]. The box sizes ranged from 2^1 to 2^k

pixels, where k is the value for which N is equal to unity. In all cases the correlation coefficient of the straight line, fitted through 9 (or 10) data points and was higher than 0.95 [23,24].

2.5 Multifractal Analysis

The theoretical basis of multifractality has been reviewed in depth by several authors [15,25-27], but here we present the essential calculations we used. The analysis was based on the box-counting data gathering technique described above, except the data analyzed were the “masses” (pixels) per square (i) rather than the number of squares (N_ϵ) containing pixels at each size or scale, ϵ . These masses were used to determine the probability at each box ($P_{i,\epsilon}$) as the number of pixels in the box divided by the total number of pixels in the image, given by

$$P_{i,\epsilon} = \frac{\text{pixels}_{i,\epsilon}}{\sum_{i=1}^{N_\epsilon} \text{pixels}_{i,\epsilon}}. \quad (1)$$

For such pixel distributions, $P_{i,\epsilon}$ varies with ϵ raised to an exponent (α_i), i.e.

$$P_{i,\epsilon} \sim \epsilon^{\alpha_i} \text{ and } \alpha_i \sim \frac{\log P_{i,\epsilon}}{\log \epsilon}. \quad (2)$$

Thus, similar to the D_B , α_i can be found from the slope of the logarithmic relationship between mass and scale. For monofractals, α_i is consistent over the pattern and corresponds to the D_B , but for multifractals α_i varies over the pattern. Whether a pattern is monofractal or multifractal is generally not readily apparent in the pattern itself but can be seen in multifractal spectra that reflect variation in scaling [20]. In essence, these spectra address how mass behaves when the pattern is resolved into a series of ϵ -sized pieces and subjected to a series of distortions.

The distortions are done using a range of arbitrary parameters, Q , applied to $I_{Q,\epsilon}$ which describes the pixel distribution inspected at some scale ϵ

$$I_{Q,\epsilon} = \sum_{i=1}^{N_\epsilon} P_{i,\epsilon}^Q. \quad (3)$$

Whereas this sum equals to one when Q is also equal to one, and N_ϵ when Q is equal to zero, the sum of all $P_{i,\epsilon}$ after being raised to other values of Q is not so predictable and forms the basis of multifractal spectra. We used $I_{Q,\epsilon}$ to determine a series of generalized dimensions (D_Q) calculated over an arbitrary range of Q s. Formally, the generalized dimension is defined by

$$D_Q = \frac{\lim_{\varepsilon \rightarrow 0} \left(\frac{\ln I_{Q,\varepsilon}}{\ln \varepsilon^{-1}} \right)}{1-Q}. \quad (4)$$

From eqs. (2) and (3), it can be seen that $I_{Q,\varepsilon}$ is proportional to ε raised to an exponent τ such that

$$\tau_Q = \lim_{\varepsilon \rightarrow 0} \left(\frac{\ln I_{Q,\varepsilon}}{\ln \varepsilon^{-1}} \right) \quad (5)$$

offering an alternative from which to find D_Q

$$\tau = (Q-1) \cdot D_Q. \quad (6)$$

We also used the $f(\alpha)$ multifractal spectrum [27]. The function $f(\alpha)$ is the fractal dimension of the set of boxes where α is some exponent that varies over the fractal. According to the method of Chhabra and Jensen [25], which is simpler to calculate than most methods, this multifractal spectrum is calculated using following

$$f(\alpha(Q)) = Q\alpha(Q) - \tau_Q \text{ and } \alpha(Q) = \frac{d\tau_Q}{dq}. \quad (7)$$

If a pattern scales uniformly, as in a typical monofractal, distorting the probability distribution reveals little change, but if a pattern scales non-uniformly, as with multifractals, distinct mathematical views emerge for both the D_Q vs. Q and $f(\alpha)$ vs. α multifractal spectra. The function D_Q vs. Q is usually either essentially unchanging or decreasing around $Q = 0$, where $D_{Q=0} \geq D_{Q=1} \geq D_{Q=2}$. For non-fractals and mono-fractals, D_Q typically has low dispersion over Q , but for multifractals, D_Q typically decreases with Q , sigmoidally around $Q = 0$. In contrast, $f(\alpha)$ vs. α multifractal spectra (see Fig. 3) are typically singly humped graphs that for monofractals (Fig. 3B) converge, but for multifractals (Fig. 3A) rise and fall more broadly.

A practical consideration that affects box-counting multifractal data is how the distribution is determined. For distributions extrapolated from small samples, for instance, some monofractals yield diverging, ostensibly multifractal curves that vary with the size of the sampling unit and whether or not the edges of an image are included [28]. Even for distributions determined from entire images, how the image is partitioned matters, as the relative amounts of very small and very large probabilities change with each attempt to break an image into sample spaces [20]. To minimize such issues, we used sampling corrections built in to the software.

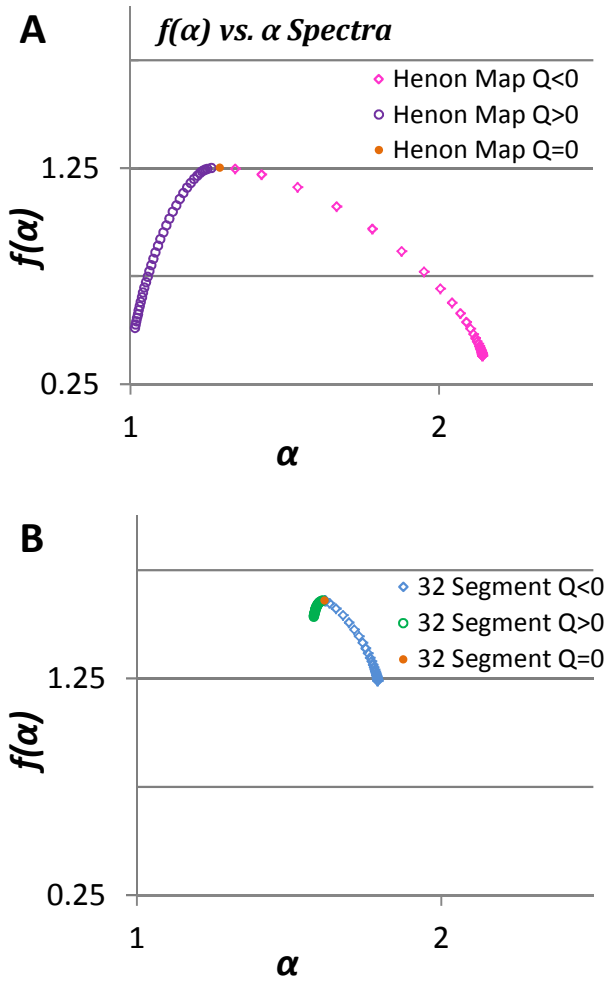


Fig. 3. Typical relation $f(\alpha)$ vs. α in multifractal spectra plotted for $-7 < Q < 7$; Henon map (A), 32 segment monofractal (B)

3 Results

3.1 Neurons

The mean values of two D_{β} s, as well as corresponding standard errors for three cell types (1, 2 and 3) of aspiny neurons were calculated, and shown in Tab. 1. As can be seen from this table, type 2 aspiny neurons have the largest mean values

for both the cell outlines or skeletons, as compared to the means of other types (1 and 3), whereas type 1 has the smallest value when compared to type 2 and 3 aspiny neurons.

To investigate whether there were significant differences between the means of the three types of neurons with respect to whether the outline or skeletonised representation is analyzed, a one-way ANOVA was used. In both cases, the calculated F -value was larger than the critical value, at a significance level $p < 0.05$ (Tab. 1) followed by the Schéffe test as a post hoc test [16]. This test pointed out significant differences for mean $(D_B)_{\text{out}}$ between type 1 and 3 ($p < 0.001$) and type 2 and 3 ($p < 0.01$). As far as mean $(D_B)_{\text{skel}}$ was concerned, we found significant differences between all three pairs: between type 1 and 2 ($p < 0.001$), between type 1 and 3 ($p < 0.05$) and between type 2 and 3 ($p < 0.01$).

Table 1. The values of two box dimensions, $(D_B)_{\text{out}}$ and $(D_B)_{\text{skel}}$, for three types of aspiny neurons in the human neostriatum. Each value is presented as mean \pm standard error. The last two columns show results of the one-way ANOVA (F is calculated F -value for degrees of freedom 103 and 2, respectively, while $F_{0.05}$ is tabulated F -value for degrees of freedom 120 and 2, respectively).

Cell type	$(D_B)_{\text{out}}$	$(D_B)_{\text{skel}}$
1	1.26 \pm 0.01	1.13 \pm 0.01
2	1.327 \pm 0.008	1.205 \pm 0.008
3	1.285 \pm 0.007	1.161 \pm 0.008
F	14.798	15.153
$F_{0.05}$		3.070

3.2 Microglia

For the multifractal analysis of microglia, 10% of the cells we investigated showed some sign of multifractal scaling. There was some variation attributable to the source of the cells (e.g., no microglia from rat hippocampus and only 2% from both human brain tumor and rat spinal cord scaled as multifractals, but 8% of microglia from elderly human brain tissue scaled as typical multifractals). There was also variation with activation state, where ramified cells scaled as multifractals significantly more than other types: in descending order of activation, 12% of ramified cells scaled as multifractals, 5% of hypertrophied, 3% of bushy, and 1% of reactive (4% of intermediate).

The spectra for cells scaling as multifractals generally had flatter D_Q vs. Q slopes and more peaked and converging $f(\alpha)$ spectra than typical multifractal spectra [20], where only half of the 10% scaled as typical multifractals and the resemblance to characteristic multifractal spectra varied. For instance, multifractal spectra obtained from a cell having one long process and one short, stout process is presented in Fig. 4.

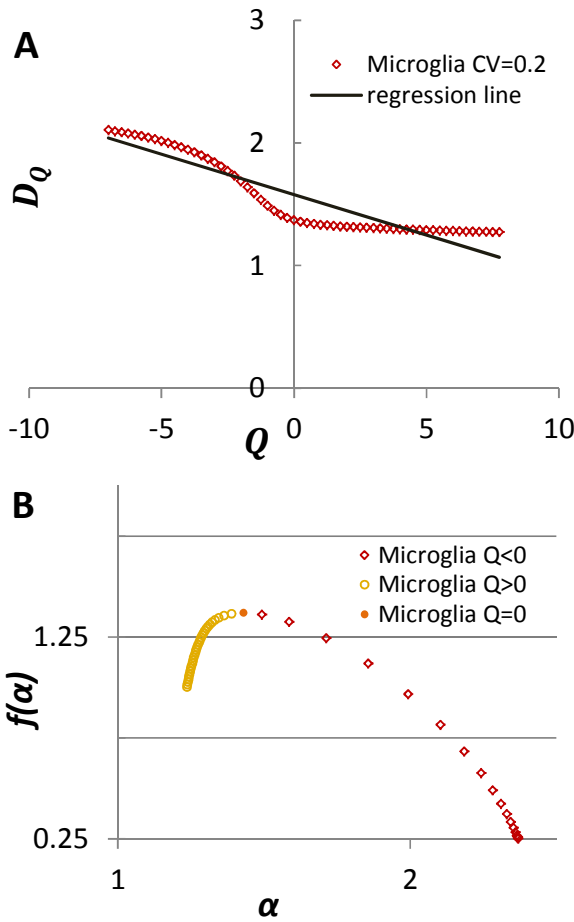


Fig. 4. Multifractal spectra for a ramified microglia

4 Discussion

4.1 Neuronal Types in the Human Neostriatum

The morphology of striatal neurons was extensively studied in the last century, recognizing a number of cell types in various mammals. As far as the neuronal morphology in the human neostriatum is concerned, subsequent authors identified five neuronal types [29,30], typically as a result of the qualitative analysis.

Aspiny neurons of the human neostriatum were classified qualitatively into three groups. Braak and Braak [29] recognized them as large multipolar cells with extended dendrites, small to medium-sized and large aspiny cells, while Graveland et al. [30] classified them as small neurons with a variable dendritic morphology,

medium-sized and large aspiny neurons. Our type 1 neurons had the smallest dendritic aberrations and cell's shape, compared to the same properties of two other types (Tab. 1). In contrast, the dendritic aberrations and the shape of the neuron, set type 2 cells into a quite different class of aspiny neurons. But, bearing in mind the dendritic aberrations only, type 3 cells can be recognized as a unique class of neurons. Nevertheless, further analysis, which covers more than two neuronal properties, will provide an appropriate link between present scheme of aspiny neurons with commonly used terms: large aspiny cells, aspiny cells with large soma and medium dendritic field and small aspiny cells [29-32].

4.2 Microglia

Microglia are dynamic cells: their function at any point in time is strongly reflected in their ad hoc form. They range from highly branched cells to cells that have pulled in their processes and become swollen, amorphous, blob-like entities. The more highly branched a cell is at any point in time, the more likely it is to be in an alert but non-activated state [11]. Previous work has demonstrated that the D_B is a useful metric for distinguishing morphological and functional differences in microglial activation states [11,21], which has important implications for determining the roles these cells play in normal development and pathology in the central nervous system. The work reported here assumed that multifractal analysis would detect fundamental relative patterns not disturbed by issues such as the size of the image, the origin of the microglia, or the intensity of staining. Whereas our previous work has indicated this to be the case for standard box-counting [11], owing to the small number of cells that scaled as multifractals in the present study, the relative importance of such features is difficult to determine and should be addressed in future studies.

Our finding that the number of microglial cells scaling as multifractals is relatively small indicates that multifractal analysis does not distinguish morphological categories as sensitively as does the D_B , however, other findings from this investigation indicate novel ways that multifractal analysis might be used to characterize microglia to supplement the power of the D_B . One such finding was that significantly more ramified cells scaled as multifractals than cells in any other category. The ramified morphology usually signifies that microglia are maintaining a resting, sentinel function in the nervous system, but previous work has shown that subtle changes that cross boundaries between traditional morphology-based categorization methods can represent important differences related to pathological state [20]. Multifractal scaling revealed by the method used here may reflect subtle or overt differences in parts of a cell (e.g., as when a single process remains in transition between states or perhaps is the only visible process).

In general, a fractal dimension describes the change in detail with scale of a pattern extracted from a digital image-but what does this number mean for a living cell? Inasmuch as the most highly reactive microglia usually have lower complexities than resting cells, decreasing complexity is ostensibly associated with retracting processes as when microglia respond to activating stimuli. But microglial activation is an active, continuous process and our analysis only

captures specific points in this progression. Certainly, fine processes retract as cells change from resting, where the tendrils swell, and stouter ones extend in the course of a response to noxious stimuli. Moreover, depending on the paradigm, ruffles or fringes may appear as microglia become reactive. In general, fluctuating relative differences in complexity with increases in activation level are in accordance with other evidence that morphology depends on multifarious features (e.g., the age of the organism the microglia were found in, the type of brain tumor, or the distance of a microglia from a lesion). Indeed, our preliminary results suggest it may be especially fruitful to look at multifractal scaling of microglia in aging and chronic disease processes such as Alzheimer's disease. In this regard, it would be interesting to clarify the nature of multifractal scaling in microglia, and pursue the potential for multifractal analysis to reveal cells in a transitional state having elements of more than one typical level of activation.

5 Conclusion

The results of the present paper emphasize the conclusion that fractal analysis has an important and growing place in neuroscience. Two fractal methods (traditional box-counting and multifractal analysis) proved useful for quantifying various features important to specific aspects of neural cell classification. Future studies should be undertaken to evaluate the potential of box-counting methods for morphometric analysis and classification of human neostriatal cells. In addition, our results entreat future investigators to clarify the nature of multifractal scaling in microglia and examine the potential for multifractal analysis to assist in quantitating and understanding the complex relationship between form and function in these morphologically and functionally dynamic cells.

Acknowledgement. The authors are grateful to The Ministry of Education and Science, Republic of Serbia, Research Project III41031.

References

1. Smith Jr., T.G., Lange, G.D., Marks, W.B.: Fractal methods and results in cellular morphology - dimensions, lacunarity and multifractals. *J. Neurosci. Meth.* 69, 123–136 (1996)
2. Fernández, E., Jelinek, H.F.: Use of fractal theory in neuroscience: methods, advantages, and potential problems. *Methods* 24, 309–321 (2001)
3. Jelinek, H.F., Cornforth, D.J., Roberts, T., Landini, G., Bourke, P., Bossomaier, T.: Image processing of finite size rat retinal ganglion cells using multifractal and local connected fractal analysis. In: *Complex Systems Conference*, Cairns, Australia (2004)
4. Nonnenmacher, T.F., Baumann, G., Barth, A., Losa, G.A.: Digital image analysis of self-similar cell profiles. *Int. J. Bio-Med. Comp.* 37, 131–138 (1994)
5. Smith Jr., T.G., Marks, W.B., Lange, G.D., Sheriff Jr., W.H., Neale, E.A.: A fractal analysis of cell images. *J. Neurosci. Meth.* 27, 173–180 (1989)

6. Jelinek, H.F., Fernández, E.: Neurons and fractals: how reliable and useful are calculations of fractal dimension? *J. Neurosci. Meth.* 81, 9–18 (1998)
7. Ristanović, D., Stefanović, B.D., Milošević, N.T., Grgurević, M., Štulić, V.: Fractal and nonfractal analysis of cell images: comparison and application to neuronal dendritic arborization. *Biol. Cybern.* 87, 278–288 (2002)
8. Mandelbrot, B.B.: *The fractal geometry of nature*. Freeman and Co., New York (1983)
9. Schroeder, M.: *Fractals, chaos, power laws*. W.H Freeman and Co., New York (1991)
10. Iannaccone, P.M., Khokha, M.: *Fractal geometry in biological systems: an analytical approach*. CRC Press, New York (1995)
11. Karperien, A.L., Jelinek, H.F., Buchan, A.M.: Box-counting analysis of microglia form in schizophrenia, Alzheimer's disease and affective disorder. *Fractals* 16, 103–107 (2008)
12. Voss, R.M., Wyatt, J.C.Y.: Multifractals and local connected fractal dimension: Classification of early chinese landscape paintings. In: Crilly, A.J., Earnshaw, R.A., Jones, H. (eds.) *Applications of Fractals and Chaos*. Springer, Berlin (1991)
13. Stanley, H.E., Amaral, L.A.N., Goldberger, A.L., Havlin, S., Ivanov, P.C., Peng, C.K.: *Statistical physics and physiology: Monofractal and multifractal approaches*. *Physics A*. 270, 309–324 (1999)
14. Fernández, E., Bolea, J.A., Ortega, G., Louis, E.: Are neurons multifractals? *J. Neurosci. Meth.* 89, 151–157 (1999)
15. Block, A., von Bloh, W., Schnellhuber, H.J.: Efficient box-counting determination of generalized fractal dimensions. *Phys. Rev. A*. 42, 1869–1874 (1990)
16. Milošević, N.T., Krstonošić, B., Gudović, R., Ristanović, D.: Fractal analysis of neuronal dendritic branching patterns in the human neostriatum: a revised classification scheme. In: Dumitrache, I. (ed.) *Proceedings CSCS-18, vol. 2*, pp. 871–876. Editura Politehnica Press, Bucharest (2011)
17. Ristanović, D., Krstonošić, B., Milošević, N.T., Gudović, R.: Mathematical modeling of transformations of asymmetrically distributed biological data: an application to a quantitative classification of spiny neurons of the human putamen. *J. Theor. Biol.* 302, 81–88 (2012)
18. Bayer, T.A., Buslei, R., Havas, L., Falkai, P.: Evidence for activation of microglia in patterns with psychiatric illness. *Neurosci. Lett.* 271, 126–128 (1999)
19. Radewicz, K., Garey, L.J., Gentleman, S.M., Reynolds, R.: Increase in HLA-DR immunoreactive microglia in frontal and temporal cortex of chronic schizophrenics. *J. Neuropathol. Exp. Neurol.* 59, 137–150 (2000)
20. Karperien, A., Jelinek, H.F., Milošević, N.T.: Multifractals: a review with an application in neuroscience. In: Dumitrache, I. (ed.) *Proceedings CSCS-18, vol. 2*, pp. 888–893. Editura Politehnica Press, Bucharest (2011)
21. Soltys, Z., Ziaja, M., Pawłński, R., Setkiewicz, Z., Janeczko, K.: Morphology of reactive microglia in the injured cerebral cortex. Fractal analysis and complementary quantitative methods. *J. Neurosci. Res.* 63, 90–97 (2001)
22. Ristanović, D., Milošević, N.T., Stefanović, B.D., Marić, D.L., Rajković, K.: Morphology and classification of large neurons in the adult human dentate nucleus: a qualitative and quantitative analysis of 2D images. *Neurosci. Res.* 67, 1–7 (2010)
23. Milošević, N.T., Ristanović, D., Jelinek, H.F., Rajković, K.: Quantitative analysis of dendritic morphology of the alpha and delta retinal ganglion cells in the rat: a cell classification study. *J. Theor. Biol.* 259, 142–150 (2009)

24. Jelinek, H.F., Milošević, N.T., Ristanović, D.: The Morphology of Alpha Ganglion Cells in Mammalian Species: a Fractal Analysis Study. *J. CEAI* 12, 3–9 (2010)
25. Chhabra, A., Jensen, R.V.: Direct determination of the $f(a)$ singularity spectrum. *Am. Phys. Soc.* 62, 1327–1330 (1989)
26. Vicsek, T.: *Fractal Growth Phenomena*. World Scientific, Singapore (1992)
27. Jęstczemski, F., Sernetz, M.: Multifractal approach to inhomogeneous fractals. *Phys. A.* 223, 275–282 (1996)
28. Berthelsen, C.L., Glazier, J.A., Skolnick, M.H.: Global fractal dimension of human DNA sequences treated as pseudorandom walks. *Phys. Rev. A.* 45, 8902–8913 (1992)
29. Braak, H., Braak, E.: Neuronal types in the striatum of man. *Cell Tiss. Res.* 227, 319–342 (1982)
30. Graveland, G.A., Williams, R.S., DiFiglia, M.: A Golgi study of the human neostriatum: Neurons and afferent fibers. *J. Comp. Neurol.* 234, 317–333 (1985)
31. DiFiglia, M., Pasik, T., Pasik, P.: Ultrastructure of Golgi-impregnated and gold-toned spiny and aspiny neurons in the monkey neostriatum. *J. Neurocyt.* 9, 471–492 (1980)
32. Dimova, R., Vuillet, J., Seite, R.: Study of the rat neostriatum using a combined Golgi-electron microscope technique and serial sections. *Neurosci.* 5, 1581–1596 (1980)

Comparison between Titan and Cobalt Hydroxyapatite-Coated Shoulder Prosthesis, during External and Internal Rotation

Mihaela Manisor¹, Cosmin Marcu¹, Gheorghe Tomoaia², and Liviu Miclea¹

¹ Department of Automation, Technical University of Cluj-Napoca, Romania

² “Tuliu Hatieganu” University of Medicine and Pharmacy, Cluj-Napoca, Romania
{Mihaela.Manisor,Cosmin.Marcu,Liviu.Miclea}@aut.utcluj.ro,
tomoia2000@yahoo.com

Abstract. A comparison study was conducted between titan and cobalt alloy, shoulder prosthesis, with hydroxyapatite coating. We simulated the internal and external rotation of the shoulder in order to compare the deformation, equivalent strain and equivalent mechanical stress appeared for different types of prosthesis covering layers. The results show that the maximum mechanical solicitations appear in similar humeral positions and the minimal solicitations were obtained for cobalt alloy prosthesis with composite coatings. This study is innovative due to the fact that it considers all muscle groups involved in shoulder movement and we obtain a realistic estimate of the mechanical solicitations appeared into the humerus, and gives an estimate of the optimal material to be used for prosthesis cementing. Possible fracture risk can be determined and prevented in case a prosthesis is implanted.

Keywords: Modeling, Finite element, Prosthesis, Shoulder, Collagen, Solicitations.

1 Introduction

Orthopedic implants are intended to support forces and must thereby be firmly attached to the rest of the skeleton [1]. Orthopedic implant devices are intended to restore the function of load-bearing joints which are subjected to high level of mechanical stresses, wear, and fatigue in the course of normal activity [2]. The implant is placed in the body either with an acrylic cement that gradually fails as regeneration of connecting bone tissue is proceeding, or without cement using an implant with an interface designed to provide the necessary attachment. However, a device fabricated from a single material usually cannot meet all physical requirements for successful implantation and function. Therefore, implants and prostheses usually consist of composites and mixtures or alloys [3].

The problem of the prosthesis material to be used still remains open. There is a need for high bonding inorganic surfaces to organic ones, and the development of

materials durable and resistant to various types of mechanical solicitations. The prostheses are usually built from biomaterials, which are synthetic or natural materials intended to function appropriately in a bio-environment. Titanium are very often used in building shoulder prosthesis, because in comparison with other metallic biomaterials, titanium and titanium alloys are more biocompatible, more corrosion resistant, lighter, more durable, and possess a reasonable balance of high strength and low elastic modulus and but support cell attachment as well [4].

Cobalt alloys have been used as hip implants due to their appropriate mechanical properties. However, they do not bond spontaneously to living bone. In order to improve bone bonding ability, the cobalt alloy implants can be coated with a layer of a bioactive material [5]. Cobalt-based alloys are quite resistant to fatigue and to cracking caused by corrosion, and they are not brittle, however, cobalt based alloys may fail because of fatigue fracture. The superior fatigue and ultimate tensile strength of the wrought CoCrMo alloy makes it suitable for the applications which require long service without fracture or stress fatigue, as for prosthesis implantation.

Efforts to increase bone in-growth of cementless prostheses have led to the application of hydroxyapatite (HA) coating. Hydroxyapatite is chemically similar to the mineral component of human bones and hard tissues. It is able to support bone in-growth and osteointegration when used in orthopedic applications. Moreover, HAp coatings have the capacity to shorten the healing process of metal based implants. The use of hydroxyapatite (HA) has been advocated to provide rapid and reliable attachment of bone to metal implants [6].

Because of the difficulty of performing implant tests *in vivo*, mathematical models have been developed to carry out the structural analysis of implants before application on a patient. Accordingly, bone-implant scapulohumeral prosthesis could be designed and studied with computer simulations. To design highly durable prostheses one has to take into account the natural processes occurring in the bone. In most cases, these models consider that the bone is anisotropic even though the reality shows otherwise. This approximation is imposed due to the fact that there is no database containing the mechanical properties of the bone according to the model topography [7].

Many physical or computer models of the scapulohumeral joint have been developed to improve understanding of its function. Some were developed to understand and analyze muscle action developed during certain movements of the upper limb, others, based on the concept of deformable bodies, were proposed and used to calculate the distribution of bone stresses, [8]. Favre et al [9] reviewed some models of the shoulder movement. One method was to force the muscle to pass or replace the complex bony geometry with simpler shapes. By using muscle force estimation techniques, muscular action and joint reaction forces addressing issues in joint stability and muscular rehabilitation or muscle transfer, were simulated.

Clavert et al, [10], made a 3D reconstruction of the humerus and the muscles involved in the movement used to display and map the distribution of stresses in

the major and minor humeral tubercles generated by muscle tendons in the rotator cuff muscles during abduction. Internal and external rotations of the shoulders were achieved by a displacement of the muscle active during the specific rotation (subscapularis for internal and infrapinatus for external rotation [11]).

Most models concentrate on force estimation and stability of the glenohumeral joint; there are few models that measure the mechanical solicitation on the humerus during the shoulder movement. It is important to determine the deformation, stress and strain into the humerus, in shoulder motion, by considering all muscle groups involved, in order to prevent possible fractures in case there is a prosthesis implanted. It is also important to determine the optimal material to be used in prosthesis construction.

The purpose of this work was to develop a 3D digital model of prosthetic humerus and the rotator cuff muscles, and to investigate shoulder movement effect on humerus and prosthesis. It was considered that for each movement, the muscles involved act with a specified force on the humerus along its insertion area. There were considered two types of prosthesis: one built from a Ti-6Al-4V alloy and the other from CoCrMo alloy. For each type of prosthesis there were simulated the internal and external rotation, for three types of biomaterials used to cover it, respectively: hydroxyapatite layers, hydroxyapatite and collagen layers and composite material hydroxyapatite and collagen. We studied and compared the mechanical solicitation obtained for both types of prosthesis. The motivations were to assist in prosthesis development, to determine the regions where the bone fracture is more likely to appear and to determine which type of prosthesis coating is more preferable to be used.

2 Finite Element Model

It is well known that the bone is not an isotropic material; therefore the development of a model is not an easy task. We can obtain a handy model of the humerus by 3D scan and digital reconstruction. By using the finite element modeling approach one can analyze the deformation, strain and stress appeared into the bone when external forces are applied. . Figure 1 shows the steps in achieving the finite element model of the humerus. A prosthesis developed by Solar [12] is used to analyze the mechanical behavior of the glenohumeral-prosthesis assembly.

In order to achieve the bone-prosthesis interface and to establish a contact between the interior surface of the bone and the prosthesis, it is necessary to cut up the bone such that the prosthesis shape is respected. For an accurate analysis, concentricity and concurrency between the contact surfaces were established. The bone prosthesis interface along with its covering layers were imported in Ansys Workbench. Six contact surfaces between the assembly components were considered: prosthesis-bone, prosthesis-first covering layer, first covering layer – second covering layer, second covering layer-bone, prosthesis – prosthetic humeral layer and prosthesis-second covering layer.



Fig. 1. a) Humerus b) Resulting mesh c) 3D reconstructed model

There were used two types of prosthesis: a Ti-6Al-4V alloy and a CoCrMo alloy prosthesis. For the titan alloy prosthesis were considered the following types of covering layers:

Hydroxyapatite 2 mm covering layer;

Hydroxyapatite 1 mm and Collagen 1 mm covering layers;

Composite material Hydroxyapatite and Collagen 2 mm covering layers

For the cobalt alloy prosthesis were considered the following types of covering layers:

Hydroxyapatite 1 mm and Collagen 1 mm covering layers

Composite material Hydroxyapatite and Collagen 2 mm covering layers

3 Simulation

We simulated two types of movement for the healthy and prosthetic shoulder: external and internal rotation. In order to simulate the muscle activity the insertion of each muscle was drawn on the finite element model of the humerus and humerus-prosthesis assembly according to Gray Anatomy [13].

The simulation performed using the following assumptions:

The muscles involved in the external rotation are: Infraspinatus, Teres minor, Deltoidus and Supraspinatus.

The muscles that participate in the internal rotation are: Subscapularis, Latissimus Dorsi, Teres major, Pectoralis major, Deltoidus and Brahialis.

In each insertion point on the humerus a force of 22N was applied.

For each individual muscle, the force direction was chosen according to its insertion angle on the humerus

4 Results

Total deformation, equivalent strain and the equivalent mechanical stress are investigated and analyzed for both shoulder rotation movements, for all types of prosthesis. Due to the fact that in many cases, the mechanical solicitations appear in similar humeral positions, there are presented only the representative cases.

4.1 *External Rotation for the Prosthetic Shoulder*

Deformation, stress and strain distribution for the humerus-prosthesis assembly is presented in figures (2), (3) and (4). As illustrated (fig 2), the maximum deformation for all types of prosthesis appears in the central diaphysis and is closer to the distal epiphysis. No deformations appear in the prosthesis area. For all types of considered prosthesis, (fig 3) the strain is distributed along the humerus, and a maximum appears in the proximal epiphysis, on the prosthesis head. The maximum stress appears in the prosthesis head (fig 4), usually in the second covering layer.

4.2 *Internal Rotation for the Prosthetic Shoulder*

If the internal rotation is simulated for the prosthetic glenohumeral joint, figure 5 shows that the total deformation appears close to the center of the diaphysis and decreases along the bone. The equivalent strain, (fig. 6 and 7), is not the same for all types of prosthesis. For the Ti-6Al-4V alloy prosthesis with: Hydroxyapatite 2 mm; composite material Hydroxyapatite and Collagen 2 mm covering layer and CoCrMo alloy with composite material Hydroxyapatite and Collagen 2 mm covering layer, the equivalent strain is distributed along the bone and reaches a maximum at the distal part of the epiphysis. For the Ti-6Al-4V alloy with: Hydroxyapatite and Collagen 2 mm covering layers and CoCrMo alloy Hydroxyapatite and Collagen 2 mm covering layers, the equivalent strain appears in the proximal part of the humerus, at the junction with the prosthesis. The equivalent stress is distributed along the prosthesis and has a maximum in the distal part of the prosthesis (fig. 8).

5 Discussion

Analyzing the simulation results one could see that for the external rotation, maximum mechanical solicitations appeared in similar humeral positions. In terms of materials used for prosthesis construction, the lowest total value was obtained for cobalt alloy prosthesis with composite coatings composed of hydroxyapatite and collagen, and the highest value is obtained for the titanium prosthesis coated with hydroxyapatite 1 mm and collagen 1 mm (fig 9). The titan alloy prosthesis

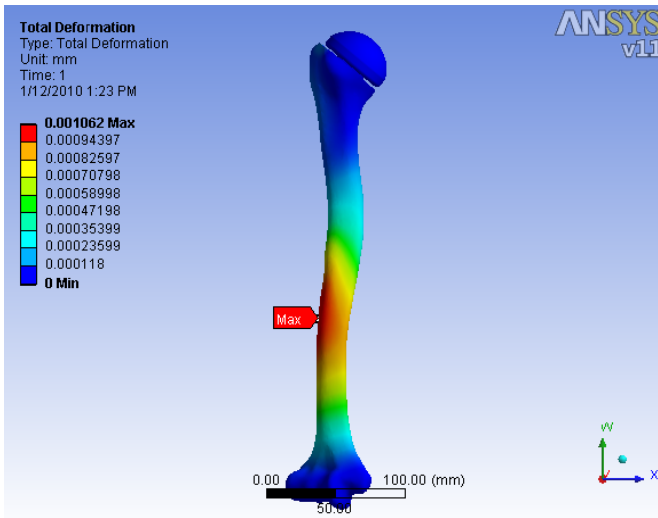


Fig. 2. Total deformation during external rotation for all types of prostheses

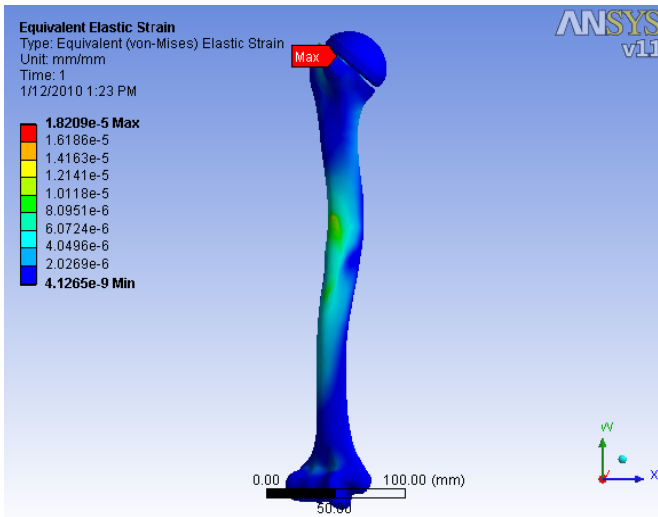


Fig. 3. Equivalent strain during external rotation for all types of prostheses

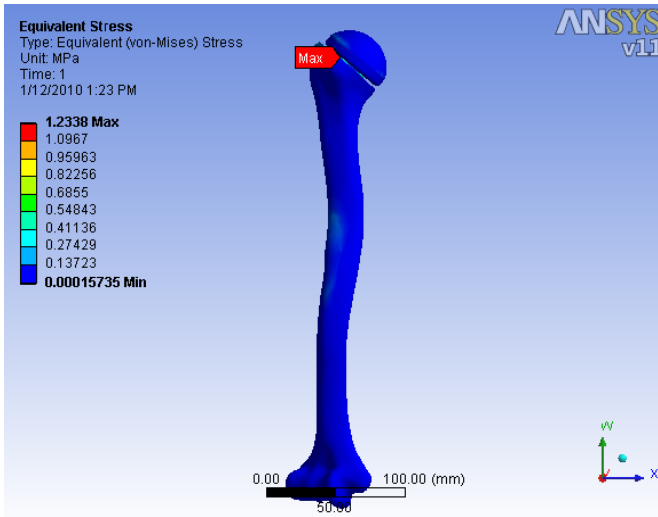


Fig. 4. Equivalent mechanical stress during external rotation for all types of prostheses

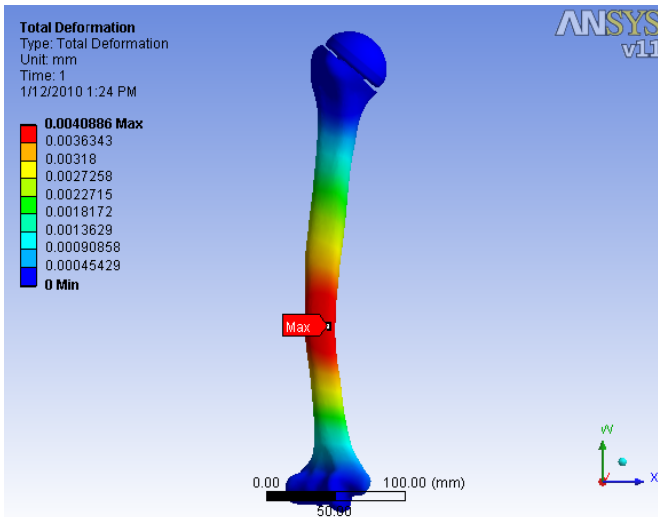


Fig. 5. Total deformation during internal rotation for all types of prostheses

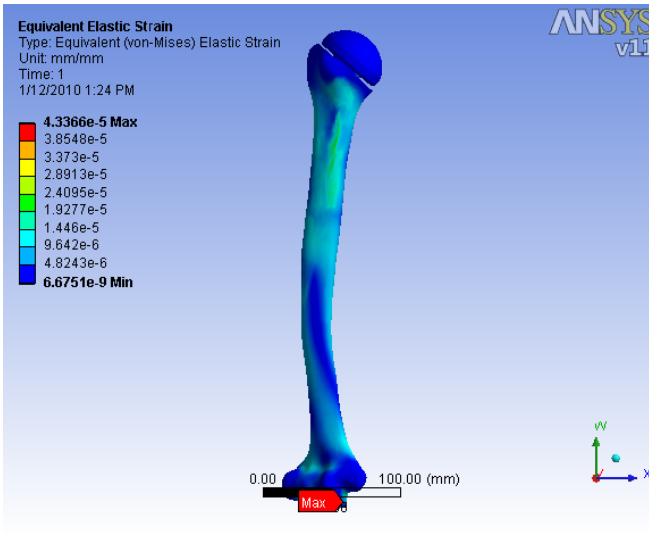


Fig. 6. Equivalent strain during internal rotation for: a Ti-6Al-4V alloy with: Hydroxyapatite 2 mm; composite material Hydroxyapatite and Collagen 2 mm covering layer and CoCrMo alloy with composite material Hydroxyapatite and Collagen 2 mm covering layer

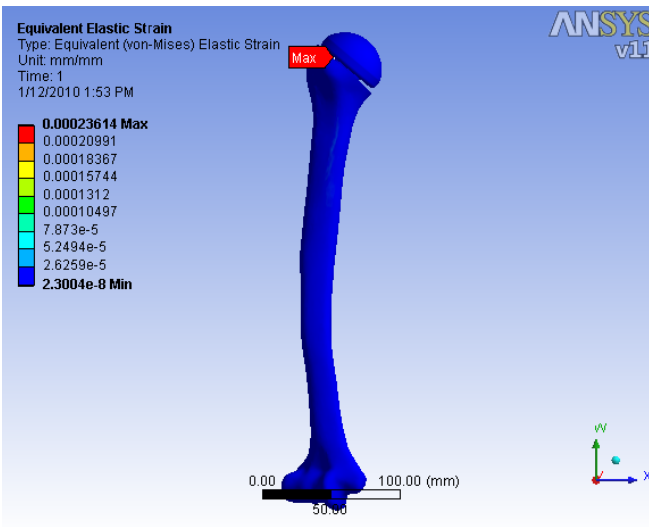


Fig. 7. Equivalent strain during internal rotation for: a Ti-6Al-4V alloy with: Hydroxyapatite and Collagen 2 mm covering layers and CoCrMo alloy Hydroxyapatite and Collagen 2 mm covering layers

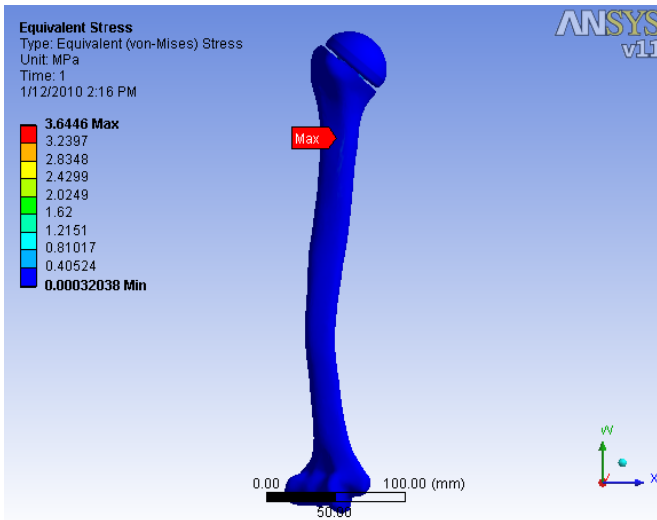


Fig. 8. Equivalent mechanical stress during external rotation for all types of prostheses

with hydroxyapatite 2 mm covering layer, also has a big value compared to the other types of prosthesis used. The minimum equivalent strain was obtained for implants with titanium and cobalt composite coatings composed of hydroxyapatite and collagen, and the maximum values for the prostheses of titanium or cobalt alloy coated with hydroxyapatite 1 mm and collagen 1 mm (fig 10). By analyzing the results from figure (10), one can see that there were obtained very small numerical values of the equivalent strain, for the titan alloy prosthesis with hydroxyapatite 2 mm covering layer. It can be seen that the difference between the values obtained for the prostheses of titanium or cobalt alloy coated with hydroxyapatite 1 mm and collagen 1 mm and the other three types of prosthesis used is very large. Mechanical stress had its minimum for titan alloy prosthesis with hydroxyapatite 2 mm coating and the maximum mechanical stress for the cobalt alloy prosthesis coated with hydroxyapatite 1 mm and collagen 1 mm (fig 11). It can be seen that the difference between the minimal and maximal value is large, and also the value of the strain for the titan alloy prosthesis with hydroxyapatite 2 mm coating is much smaller than the values obtained for all the other prosthesis.

If the internal rotation was simulated the maximum total deformation occurred in the diaphysis closer to the proximal epiphysis. The maximum equivalent strain was obtained in the distal part of the prosthesis at the junction between bone and prosthesis and distal epiphysis. The minimum total deformation for internal rotation was obtained when we used cobalt alloy prosthesis with composite coatings composed of hydroxyapatite and collagen, and the maximum for titanium alloy

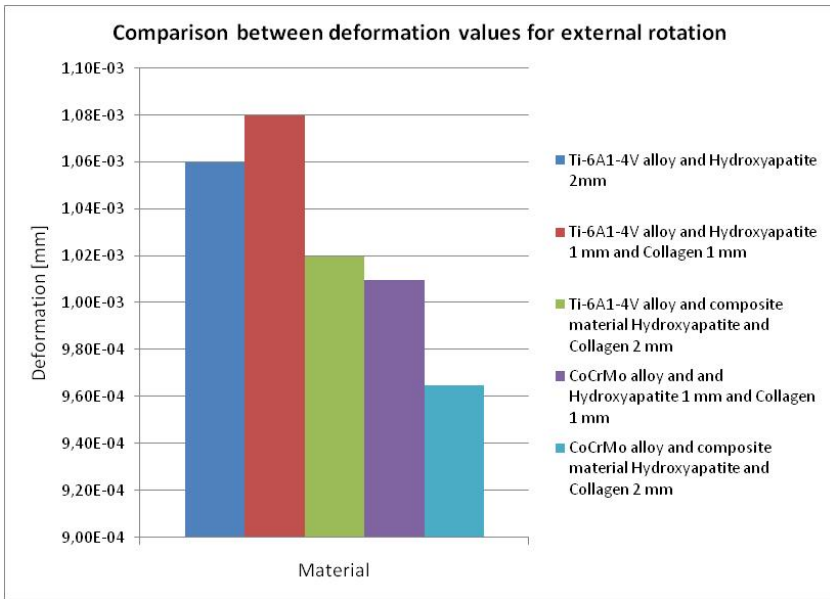


Fig. 9. Comparison between deformation values for external rotation

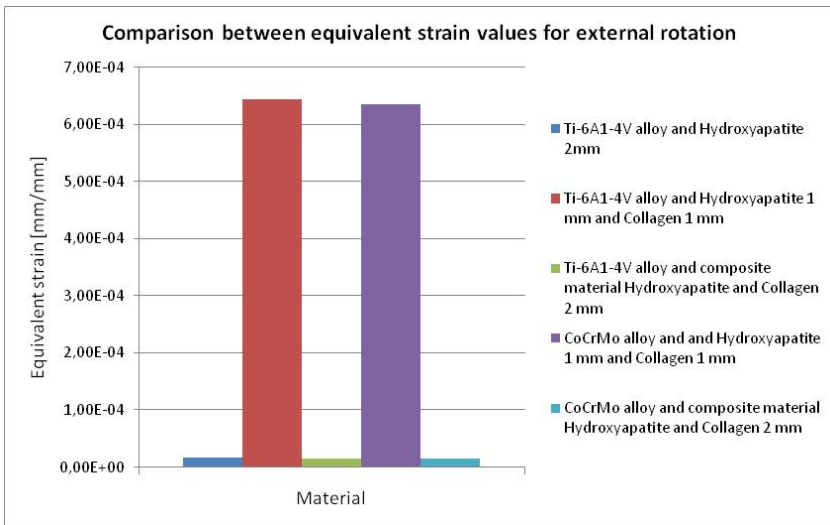


Fig. 10. Comparison between equivalent strain values for external rotation

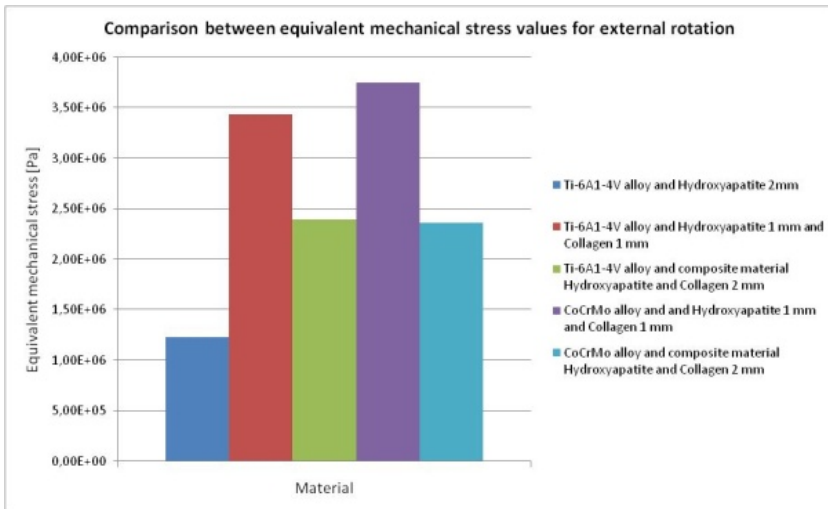


Fig. 11. Comparison between equivalent mechanical stress values for external rotation

prosthesis coated with hydroxyapatite 1 mm and collagen 1 mm (fig. 12). By analyzing figure (12), one can see that the numerical values obtained for the deformation have similar range. Equivalent strain was minimum for cobalt alloy prosthesis with composite coating and maximum if we used titanium alloy prosthesis with hydroxyapatite 1 mm and collagen 1 mm covering layers (fig. 13). It can be seen that, similar to the external rotation case, the difference between the values obtained for the prostheses of titanium or cobalt alloy coated with hydroxyapatite 1 mm and collagen 1 mm and the other three types of prosthesis used is very large. The minimum mechanical stress was obtained when we used a titanium alloy prosthesis coated with hydroxyapatite 2 mm and a maximum for cobalt alloy prosthesis with composite covering layers (fig. 14). It can be seen that the numerical values are very large for the rest of prosthesis if we compare to the numerical values obtained for the titanium alloy prosthesis coated with hydroxyapatite 2 mm.

Analyzing the data we can see that we obtain a maximum total deformation titanium alloy prosthesis coated with of hydroxyapatite 1 mm and collagen 1 mm, for both movements. For both rotational movements to obtain minimum values, cobalt alloy prosthesis with composite coatings, had to be used. One can say that in terms of total deformation it is preferable to use cobalt alloy prosthesis with composite coating. If we analyze the strain we can see that we obtain a maximum for titanium alloy prosthesis coated with of hydroxyapatite 1 mm and collagen 1 mm for both movements and we obtain a minimum value for titanium alloy prosthesis with composite material coating. If we look at the values obtained for the

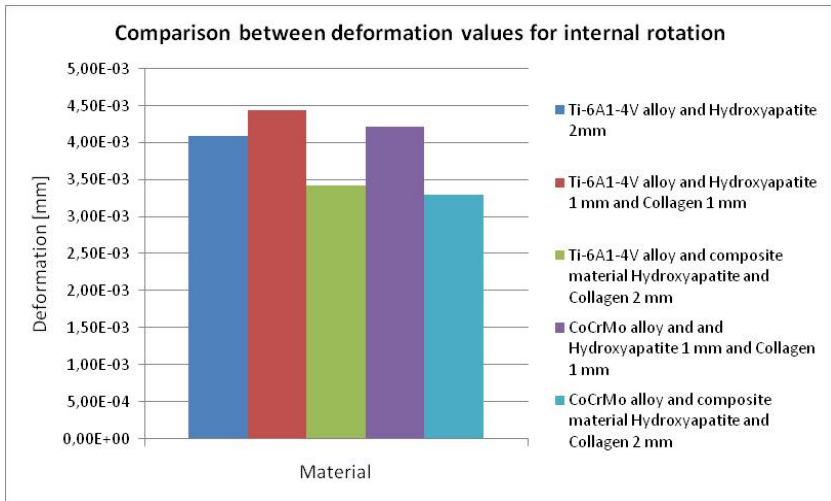


Fig. 12. Comparison between deformation values for internal rotation

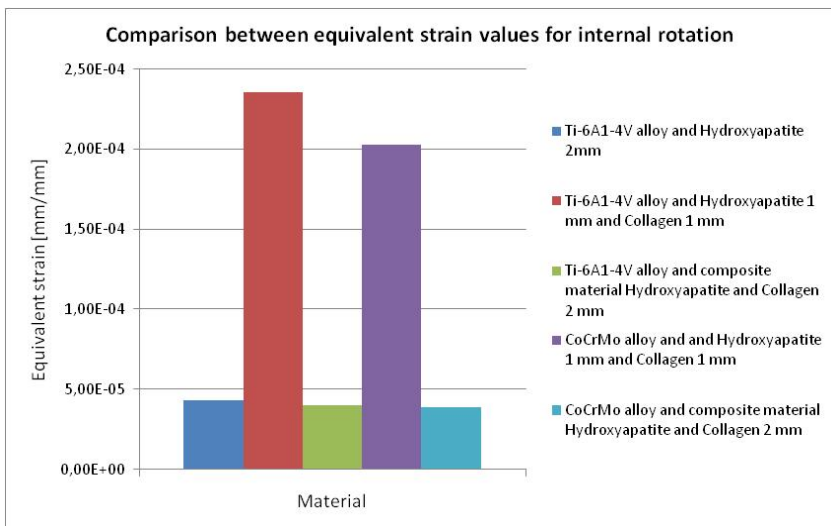


Fig. 13. Comparison between strain values for internal rotation

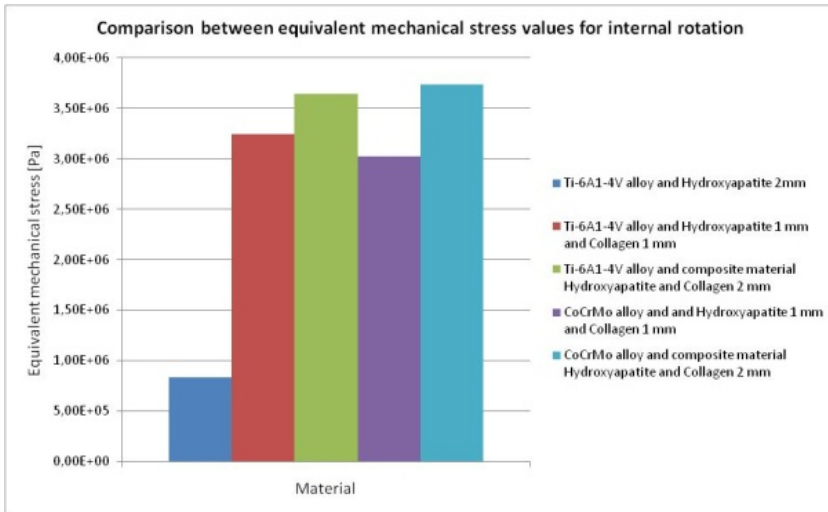


Fig. 14. Comparison between mechanical stress values for internal rotation

mechanical stress it be seen that the maximum value obtained for the external rotation is for the titanium alloy prosthesis with hydroxyapatite 1 mm coatings and cobalt 1mm and for the internal rotation for the cobalt alloy prosthesis with composite covering layers. Minimum value for the stress is obtained for titanium alloy prosthesis with hydroxyapatite 2 mm coatings for external rotation. For the internal rotation we obtain a minimum using titanium alloy prosthesis with hydroxyapatite 2 mm covering layers. By analyzing these data we can see that there are materials for which the deformation is minimal but the equivalent strain and stress are maximum. Although the mechanical stress gives us more data on fracture risk the total deformation and equivalent strain can not be ignored. It can be concluded that it is preferable to use cobalt alloy prosthesis with composite material coatings.

6 Conclusions

This study analyzed the load, stress and strain equivalent distribution for titan and cobalt alloy prosthesis covered with different types of biomaterials, during external and internal rotation of the shoulder.

3D models for the humerus, the prosthesis and the humerus-prosthesis assembly were obtained and the external rotation and internal rotations were simulated.

In order to simulate the muscle activity the insertion area of each muscle was drawn on the finite element model of the humerus and humerus-prosthesis. In each insertion point on the humerus a force of 22N was applied.

The simulation results showed that for all the movements the maximum mechanical solicitation appeared in similar areas for all types of prosthesis used.

Analyzing the data obtained it can be concluded that none of the materials used for the simulations did behave better than the rest materials, those who had low

strain and deformation, had higher values for mechanical stress, but none of the analyzed cases exceeded the maximum limits, which would destroy the materials. It can be concluded that it is preferable to use cemented cobalt alloy prosthesis with composite coatings. By considering this one has to take in consideration other properties of the material used: elasticity, yield stress, ductility, toughness, wear resistance, bonding to the bone. Also a real estimate of the muscle forces has to be obtained, by using EMG (electromyography) based methods [14].

Using this type of modeling one can study the material characteristics used to construct the prostheses and to perform complete analyses, due to the fact that there are other physical properties that can be studied. By incorporating the muscle insertions into the model, the maximum humeral fracture risk can be determined, for all shoulder movements. Due to the fact that the prosthesis material can be changed, one can say that these models are versatile and can be further used to test cemented prosthesis, and different cement materials, in order to determine their biomechanical behavior.

Although there are many models that analyze muscle action developed during certain movements; there are few models that measure the mechanical solicitation on the humerus during the shoulder movement. Only muscles inserted around the humeral rotator cuff are used in previous studies. This study is innovative due to the fact that it considers all muscle groups involved shoulder movement and it obtains a realistic estimate of the mechanical solicitations appeared into the humerus for many types of material used to build the prosthesis. Possible fracture risk can be determined and prevented in case a prosthesis is implanted.

References

1. Carter, D.R., Beaupre, G.S.: Skeletal function and form. Cambridge University Press (2001)
2. Paital, S.R., Dahotre, N.B.: Calcium phosphate coatings for bio-implant applications: Materials, performance factors, and methodologies. *Materials Science and Engineering: R: Reports*, 1-70 (2009)
3. Erli, H.J., Marx, R., Paar, O., Niethard, F.U., Weber, M., Wirtz, D.C.: Surface pre-treatments for medical application of adhesion. *BioMedical Engineering OnLine* (2003)
4. Hsu, H.C., Hsu, S.-K., Wu, S.C., Lee, C.J., Ho, W.F.: Structure and mechanical properties of as-cast Ti-5Nb-xFe alloys. *Materials Characterization* 61, 851–858 (2010)
5. Ortiz, J.C., Cortes, D.A., Escobedo, J.C., Almanza, J.M., Muniz, C.R., Luna, J.S., Rodriguez, N.A.: Bioactive coating on a cobalt base alloy by heat treatment. *Materials Letters*, 329–332 (2011)
6. Gadow, R., Killinger, A., Stiegler, N.: Hydroxyapatite coatings for biomedical applications deposited by different thermal spray techniques. *Surface and Coatings Technology* 205, 1157–1164 (2010)
7. Hollerbach, K., Hollister, A.M., Ashby, E.: 3-D Finite Element Model Development for Biomechanics: A Software Demonstration. In: *Sixth International Symposium on Computer Simulation Biomechanics* (1997)

8. Guldberg, R.E., Hollister, S.J., Charras, G.T.: The accuracy of digital imagebased finite element models. *Journal of Biomechanical Engineering*, 289–295 (1998)
9. Favre, P., Snedeker, J.G., Gerber, C.: Numerical modelling of the shoulder for clinical applications. *Phil. Trans. R. Soc. A.*, 2095–2118 (2009)
10. Clavert, P., Zerah, M., Krier, J., Mille, P., Kempf, J.F., Kahn, J.L.: Finite element analysis of the strain distribution in the humeral head tubercles during abduction: comparison of young and osteoporotic bone. *Surg. Radiol. Anat.*, 581–587 (2006)
11. Buchler, P., Ramaniraka, N.A., Rakotomanana, L.R., Iannotti, J.P., Farron, A.: A finite element model of the shoulder: application to the comparison of normal and osteoarthritic joints. *Clinical Biomechanics*, 630–639 (2002)
12. ****, Zimmer M/L taper hip prosthesis with kinectiv technology, Zimmer. Inc. (2007)
13. Gray, H.: *Anatomy of the Human Body*. Lea & Febiger, Philadelphia (1918)
14. Man, S., Herle, S., Cescon, C., Lazea, G., Merletti, R.: Online classification of electromyographic signals during finger isometric flexion and the role of visual feedback. In: *The 18th International Conference on Control Systems and Computer Science (CSCS18)* (2011)

Artificial Emotion Simulation Model and Agent Architecture: Extended*

Valentin Lungu

Faculty of Computer Science and Automatic Control, Polytechnics University of Romania,
Splaiul Independentei 313, sector 6, 060042 Bucharest, Romania
valentin.lungu.aimas@gmail.com

Abstract. Research in human emotion has provided insight into how emotions influence human cognitive structures and processes, such as perception, memory management, planning and behavior. This information also provides new ideas to researchers in the fields of affective computing and artificial life about how emotion simulation can be used in order to improve artificial agent behavior. This paper describes an emotion-driven artificial agent architecture based on rule-based systems that not attempts to provide complex believable behavior and representation for virtual characters, as well as attempts to improve agent performance and effectiveness by mimicking human emotion mechanics such as motivation, attention narrowing and the effects of emotion on memory. To this end, our approach uses an inference engine, a truth maintenance system and emotion simulation to achieve reasoning, fast decision-making intelligent artificial characters.

Keywords: affective computing, emotion simulation, agent architecture, artificial life, virtual character.

1 Introduction

The study of what is now called emotional intelligence has revealed yet another aspect of human intelligence. Emotions were shown to have great influence on many of human activities, including decision-making, planning, communication, and behavior. Emotion theories attribute several effects of emotions on cognitive, decision and team-working capabilities (such as directing/focusing attention, motivation, perception, behavior towards objects, events and other agents. This paper attempts to analyze some important aspects of emotions of human emotions and

* The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/61178.

model them in order to create more believable artificial characters and more effective agents and multi-agent systems. This paper mainly deals with the aspect of memory and emotional influences on memory, perception, and reasoning.

Autonomous artificial entities are required to be able to exist in a complex environment and respond to relevant changes in the environment, reason about the selection and application of operators and actions and be able to pursue and accomplish goals. This type of agent has applications in various fields, including character behavior in virtual environments, as used in e-learning applications and serious games.

Emotion integration is necessary in such agents in order to generate complex believable behavior. It has been shown that emotions influence human cognition, interaction, decision-making and memory management in a beneficial and meaningful way. Several studies in the field of artificial intelligence have also been conducted [1, 2, 3] and have shown the necessity of emotion simulation in artificial intelligence, especially when human-computer interaction is involved.

This paper presents an artificial emotion simulation model and agent architecture (section 2) that implements various human-inspired mechanics, such as emotions (section 3.), knowledge indexing (4.2.) and attention narrowing and memory management (section 5.) in an effort to provide artificial characters in virtual environments with believable behavior, as well as improve the functioning of artificial agent cognitive structures and procedures, improving agent performance, by mimicking human emotion mechanics such as motivation, attention narrowing and the effects of emotion on memory.

2 Architecture

The proposed agent architecture is presented in Fig. 1 with an expanded view of the inference engine in Fig. 2.

Facts and rules are fed into the inference engine according to their associated emotion and its intensity, by way of knowledge indexing, as described in section 4.

The inference engine uses two inference methods (forward and backward chaining) running in parallel and produce rule activations along with an associated emotion vector, which are added to an agenda and a plan respectively [4]. These are then merged into a conflict set and sorted according to associated emotion intensity and then triggered.

The trigger algorithm (Alg. 1) shows the steps taken when a rule instance is fired.

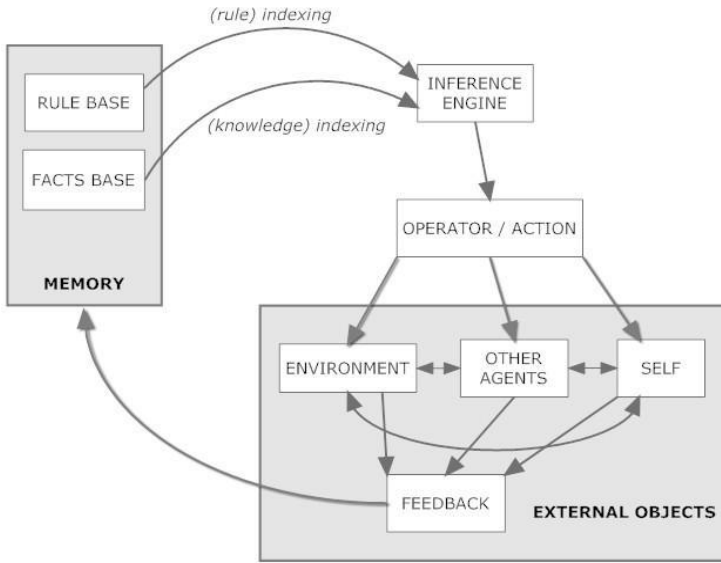


Fig. 1. System architecture

Algorithm 1 Trigger(δ)

Input: rule instance δ

Ensure: triggers rule instance δ

if $\neg triggered_\delta$ then

$triggered_\delta \leftarrow true$

 if $goal_\delta$ then

$Influence(\delta, \epsilon_{goal} \cdot \phi_{goal})$

 end if

 for all $expr \in \delta$ do

$Evaluate(expr, \delta)$

 end for

$rulesTriggeredThisTurn \leftarrow rulesTriggeredThisTurn \cup \delta$

end if

The architecture also uses a truth maintenance system [4] so that the agent maintains a consistent world view, each new inferred fact has a justification and we are able to propagate associated emotions up the inference chain (see section 4.1.).

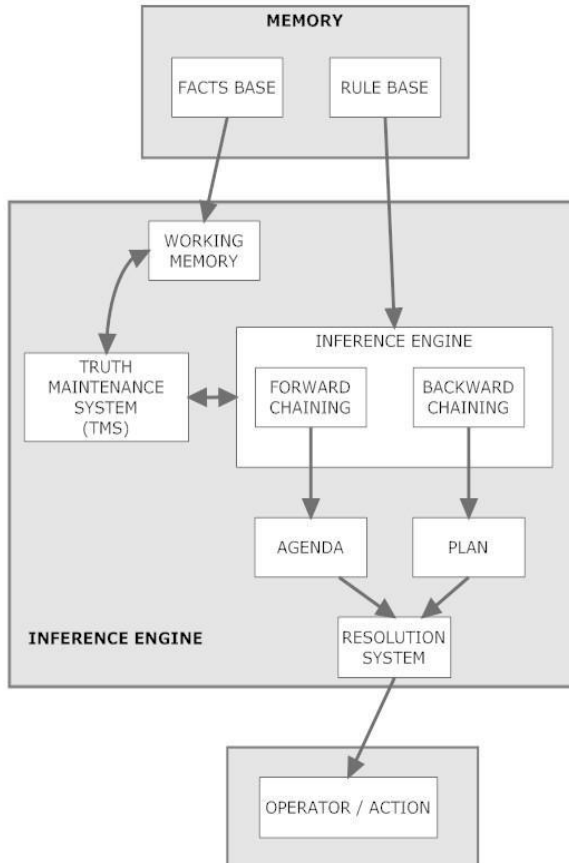


Fig. 2. Inference engine diagram

This algorithm (Alg. 2) forms the core of our framework. This is where all other algorithms are called from, and performs one behavior cycle for the agent.

Algorithm 2 Agent.Run(Plan p)

```

rulesTriggeredThisTurn  $\leftarrow$  {}
Forward.Update() {see Alg. 5}
p  $\leftarrow$  {}
for all  $g \in$  goals do
  Update( $g$ , p) {see Alg. 3}
end for
stepsTaken  $\leftarrow$  0
while  $p \neq \emptyset$  and  $stepsTaken < maxSteps$  do
   $\delta \leftarrow$  Pop(p)
  if  $\neg triggered_\delta$  then
    Trigger( $\delta$ ) {see Alg. 1}
    stepsTaken++
    Forward.Update() {see Alg. 5}
    p  $\leftarrow$  {}
    for all  $g \in$  goals do
      Update( $g$ , p) {see Alg. 3}
    end for
  end if
end while

```

2.1 Backward Chaining Engine

An agent needs to be able to accomplish its goals. Hierarchical goal decomposition is a powerful tool, allowing the agent to represent and solve complex problems. Backward chaining is a lazy inference method. It only does as much work as it has to. Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent that matches a desired goal. A backward chaining inference engine is best at breaking down goals into subgoals.

An important part of the agent's logic are the plan revision (Alg. 3) and plan pursuit (Alg. 4) algorithms. After all possible plan steps are added to the plan (which, as the event queue, uses a sorted insertion list), the most important among these (according to the associated emotion) is triggered and knowledge, events and the plan are then revised.

Algorithm 3 Update(Goal G, Plan p)

Input: Goal G**Ensure:** update G's subgoals

```

if accomplished(G) then
  return
end if
for all  $r \in relatedrules$  do
  {r}ules that contain a form of this goal as a consequent
  for all  $a \in antecedent_r$  do
     $g = new\ Goal(a)$  {bind all necessary variables so that g would lead to
    G and g still matches a}
     $\epsilon_g = \epsilon_G \cdot \phi_{goal-subgoal}$ 
     $subgoals_G \leftarrow subgoals_G \cup \{g\}$ 
  end for
end for
for all  $g \in subgoals_G$  do
  if  $depth_g < maxDepth$  then
    Update(g, p)
  end if
end for
Run(G, p) {see Alg. 4}

```

Algorithm 4 Run(Goal G, Plan p)

Input: Goal G, Plan p**Ensure:** adds expressions that lead to G to the global plan

```

if  $depth > maxDepth$  then
  return
end if
for all  $\delta \in agenda$  do
  if  $Match(G, \delta)$  then
     $p \leftarrow p \cup \{\delta\}$ 
  end if
end for
for all  $g \in subgoals_G$  do
  Run(g, p)
end for

```

2.2 Forward Chaining

Agents in a dynamic environment where multiple aspects of the world are currently changing must be able to infer new knowledge about the world. Also, said

agents should also be able to act in uncertain conditions (conditions of uncertain knowledge). A forward chaining inference engine is used to infer knowledge about the world that is not strictly goal-related, and a truth maintenance system is used to handle conflicting knowledge and maintain a consistent set of beliefs about the world.

The forward inference engine attempts to fire as many rule instances as it can (Alg. 5), subject to availability and algorithm parameter imposed restrictions.

Algorithm 5 Forward.Update()

```

stepsTaken ← 0
while perceptions ≠ ∅ and stepsTaken < maxSteps do
  p ← Pop(perceptions)
  TMSEngine.Evaluate(p)
  stepsTaken++
  p.Influence()
end while
for all r ∈ rules do
  if Repeatable(r) then
    Reactivate(r)
  end if
  agenda ← agenda ∪ agendar
  agendar ← ∅
end for

```

Algorithm 6 Forward.Run()

```

Update() {see Alg. 5}
stepsTaken ← 0
while agenda ≠ ∅ and stepsTaken < maxSteps do
  δ ← Pop(agenda)
  if ¬triggered(δ) then
    Trigger(δ) {see Alg. 1}
    stepsTaken++
    Update() {see Alg. 5}
  end if
end while

```

2.3 Conflict Set Resolution

We are not proposing a specific mechanic that influences the decision making process, however, knowledge retrieval and event handling are influenced through attention narrowing and knowledge indexing, thereby determining rule order in

the agent's agenda or plan. It should be clear that the agent adapts its behavior according to the dominant emotion and its context; for example, if its current state is fear, it will make efforts (introduce operators in the agenda or formulate plans, depending on the particularities of the agent) in order to eliminate the threat or get away from this situation. It will also make efforts in the future (if able) to avoid circumstances that led to the current state. Another example, if something causes the agent joy, the agent will be more likely to fire operators or formulate plans that lead to the joyous occurrence.

3 Emotion Simulation

Increased interest in emotions has revealed another aspect of human intelligence, showing that emotions have a major impact on the performance of many human tasks, including decision-making, planning, communication and behavior [5].

We intend to use our emotion simulation model in order to achieve the same effects that emotions have on human cognition, in artificial agents: catalyst in brain activity, motivation, direct attention and behavior, establish importance of events around us, therefore we follow a human cognitive emotion theory.

According to Lazarus [6], an emotion is a three-stage process:

1. cognitive appraisal *the subject assesses the event cognitively*
2. physiological change *biological changes occur as a reaction to cognitive appraisal (increased heart rate, adrenal response)*
3. action *the individual feels the emotion and chooses how to react*

This is the model we will be using in our implementation in order to appraise events and apply operators:

1. triggering event
2. cognitive assessment of event and associated emotional state
3. modify emotional state and physiological response (if any)
4. an action or series of actions are inserted into the action queue

We represent an emotion in a way similar to how primary colors combine, primary emotions could blend to form the full spectrum of human emotional experience. For example interpersonal anger and disgust could blend to form contempt. An emotion is encoded as a series of eight weights, one for each basic emotion category of Plutchik's taxonomy (Fig. 3).

Each weight represents the influence that emotion has on the current state [7].

The sum of emotion weights in our representation always equals one. The enforcement of this condition when certain emotions are modified by external stimuli allows gradual shifts from one dominant emotion to another, thereby avoiding sudden emotion changes (such as going from extreme anger (rage) to joy; the re-normalization would increase joy and decrease anger in order to keep the sum of emotion weights equal to one; however, the dominant emotion would not become

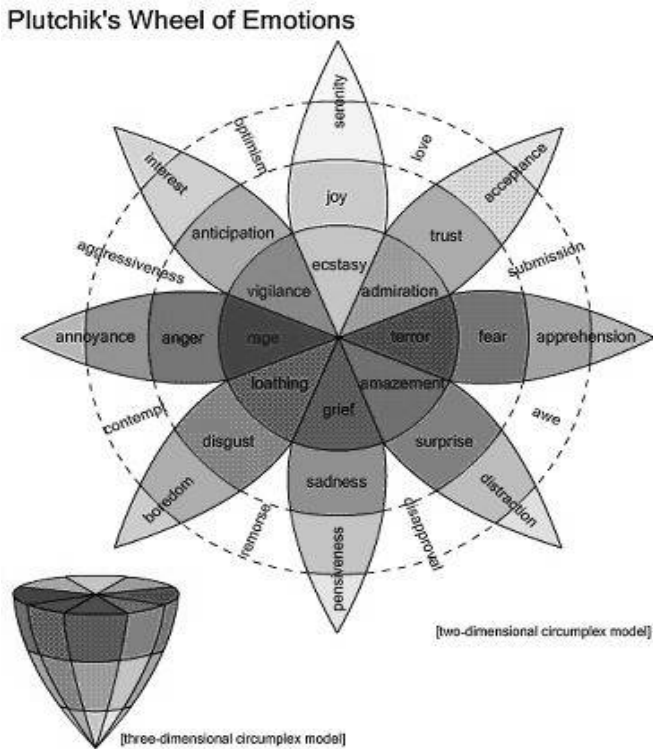


Fig. 3. Plutchik’s wheel of emotions

joy unless the stimulus was extremely powerful). The dominant (main) emotion is the emotion or combination of two adjacent emotions with the greatest weight / combined weights. Predisposition towards simple or complex emotions is left as the subject of future studies. For now, we will assume that characters / humans prefer simpler emotions as opposed to more complex ones.

Initially the agent is in a neutral, indifferent state, called the baseline state, where all emotion weights are equal to each other and equal to 0.125 (1/8). This representation was chosen because it provides an emotional state to relate to and because it is simpler to compute the effects of events on the emotional state as opposed to a percentage approach where it is possible for the sum of percentages to be lower than 100%. Emotion intensity (either one of the eight basic emotions or one of the eight compound emotions shown in Fig. 3) is calculated relative to this baseline, using the following formula:

$$\frac{\sum E_r}{n_r} \cdot \varphi,$$

where E_r is the value of the weight that belongs to each basic emotion that contributes to the emotion being evaluated minus the baseline weight, as modified by

internal and external events, based on the agent's preferences and expectations, as in the OCC model [1], n_r is the number of relevant emotions and φ is a normalization factor that scales the result into the $[0, 1]$ range.

$$\varphi = \frac{1}{1-0.125} = 1.142857.$$

The emotion with the highest intensity according to the above formula is chosen as the dominant (main) emotional state. If no clear emotion is dominant (most weights are within ε of each other), the character is considered to be in the baseline state.

We also need a way of compositing emotions and computing the influence of one emotion on another (the resulting emotion). A weighted average works best:

Compositing emotions according to factors ϕ_1 and ϕ_2 :

$$e_1 \circ e_2 = \frac{\phi_1 \cdot e_1 + \phi_2 \cdot e_2}{\phi_1 + \phi_2}$$

if we want both emotions to have the same influence, then $\phi_1 = \phi_2$:

$$e_1 \circ e_2 = \frac{\phi \cdot e_1 + \phi \cdot e_2}{2 \cdot \phi} = \frac{e_1 + e_2}{2}$$

if we want one emotion e_2 to have an influence $\theta < 1$ on another emotion e_1 ,

$$e_1 \circ e_2 = \frac{(1-\theta) \cdot e_1 + \theta \cdot e_2}{(1-\theta) + \theta} = (1-\theta) \cdot e_1 + \theta \cdot e_2$$

Computing emotion similarity:

$$similarity(e_1, e_2) = - \sum_{b \in e} (b_{e_1} - b_{e_2})^2$$

The weight vector representation is very well suited to graphical representation, as we can have one graphical representation for each base emotion, and use the emotion vector weights to interpolate them. This resulting representation may in turn be interpolated with animation sequences such as speaking or breathing. We may also vary activity levels by speeding up or slowing down the animation according to emotion intensity. The baseline state yields the agent's base movement speed. This is increased or decreased according to emotion intensity [7].

4 Knowledge Representation

The artificial agent will be able to learn about its environment (and other agents) through the emotions and emotion intensities associated to its knowledge items (facts and rules).

4.1 Ontology

The agent's knowledge is organized in an inheritance-based hierarchical manner called an ontology. We are using an ontology because we want to be able to share common understanding of the structure of information among people or software agents and separate domain knowledge from operational knowledge.

Learning about the environment, human users and other agents is key to providing believable behavior in artificial characters [3]. Each class and individual in the ontology has an associated emotion vector and intensity, which allows the agent to learn how to interact with a certain individual, as well as a class of individuals, as these associated emotions are propagated up the ontology hierarchy.

The emotion e associated with an ontological class is

$$e = \frac{\sum_{n_{subclasses}} e_{subclass}}{n_{subclasses}}$$

or, if an individual's or a subclasses' emotion has been modified by an emotion vector e' , we modify its superclasses

$$e_{superclass} = e_{superclass} \circ e'(\theta)$$

where $\theta = 1 / n_{individuals}$ is used as the influence factor in the emotion composition formula.

In this way, we can also take a look at the emotion associated with the top-level root node in the ontology *Thing* in order to get an idea of the agent's feelings about its environment / universe.

Initially, a new individual's associated emotion will be

$$e_{individual} = \frac{\sum e_{superclass}}{n_{superclasses}}$$

4.2 Knowledge Indexing

Emotions have a big impact on memory. Many studies have shown that the most vivid memories tend to be of emotionally charged events, which are recalled more often and with greater detail than neutral events. This can be linked to human evolution, when survival depended on behavioral patterns developed through a process of trial and error that was reinforced through life and death decisions. Some theories state that this process of learning became embedded in what is known as *instinct*.

A similar system may be used to decide the importance and relevance of events and knowledge and manage memory elements in artificial agents. This process is described in section 5.

5 Perception and Memory Management

5.1 Perception

It has been discovered in humans that an increase in arousal levels leads to attention narrowing [8]. Attention narrowing is a decrease in the range of stimuli from the environment that the subject is sensitive to, in other words, perception will be focused on more arousing events and details first and these will be processed sooner than non-arousing details and events [8]. This means that highly emotional events are more likely to be processed when attention is limited, leading experts to attest that emotional information is processed according to priority [9].

In fast-changing environments an agent may not have enough resources in order to react to every event that occurs, therefore a way of prioritizing important / significant events. This may be achieved through attention narrowing. Events are stored into an event queue and sorted according to emotional arousal associated with the event, ensuring more important events are processed first. Emotion preference is a domain-specific problem, for example fear (usually signaling danger) may be considered more important than disgust, and therefore should be processed with a higher priority. Events will be queued according to emotion intensity.

Algorithm 7 Perceive(Event e)

Input: Event e

Ensure: adds event e to the perception queue based on its importance

$perceptions \leftarrow perceptions \cup e$ {perceptions is an ordered insertion list}

5.2 Memory Management

It has been determined that humans are more likely to remember bits of knowledge in similar emotional contexts that they learned or have used said knowledge in the past. In order to replicate this mechanic in artificial agents, we propose a system of indexing knowledge based on dominant emotion (including secondary emotions) and emotion intensity, similar to a hash-table, because there is a high probability for the same emotional context to be triggered in similar situations which may be solved by similar types of reasoning.

We have devised a way of indexing facts and rules based on dominant emotion and emotion intensity similar to a hash-table. Both facts and rules have an associated emotion and intensity, which will be updated by a fraction of the agent's emotional state every time they are used to infer new data. At each new inference stage, possible activations are sorted according to the associated emotional intensity.

The main differences between this model and other emotion simulation approaches are the perception and memory management mechanics that allow emotions to indirectly influence the agent's decision and planning processes. Another difference from other approaches is that in this model, emotions serve a purpose, and are not merely a goal in and of themselves.

Algorithm 8 Emotion influence on rule instances

Input: rule instance δ , emotion ϵ **Ensure:** propagation of emotion influence through the system (learning)

```

 $\epsilon_{\delta+} = \epsilon \cdot \phi_{event-\delta}$ 
for all  $expr \in \gamma_{\delta}$  do
   $\epsilon_{rule_{expr}+} = \epsilon \cdot \phi_{event-\delta} \cdot \phi_{\delta-rule}$ 
   $\epsilon_{expr+} = \epsilon \cdot \phi_{event-\delta} \cdot \phi_{\delta-expr}$ 
  for all  $\sigma \in expr$  do
     $\epsilon_{\sigma+} = \epsilon \cdot \phi_{event-\delta} \cdot \phi_{\delta-expr} \cdot \phi_{expr-\sigma}$ 
  end for
end for
for all  $g \in goals_{\delta}$  do
   $\epsilon_{g+} = \epsilon \cdot \phi_{event-\delta} \cdot \phi_{\delta-goal}$ 
end for

```

Algorithm 9 Event influence

Input: Event event**Ensure:** propagation of emotion influence through the system (learning) $\Delta = \{\text{set of rule instances triggered this inference cycle}\}$

```

for all  $\delta \in \Delta$  do
  Influence( $\delta, \epsilon_{event}$ ) {see Alg. 8}
end for

```

5.3 Learning

Our model uses emotion intensity as the main means to learn by. Associated emotion intensity weighs each element in the model (rules, facts) and reinforces some while suppressing others, however, by using the similarity function to weigh elements instead of the emotion intensity itself, the elements being enforced / suppressed change dynamically depending on the agent's emotion state. This means that knowledge relevant to the agent should be able to access knowledge more relevant to its state faster.

We want the agent to adapt its behavior to a dynamic environment, so whenever an event changes the agent's (emotional) state, we influence all inferences (and knowledge structures that took part in those inferences) so that their associated emotion matches a little bit more with the agent's current state, increasing the likelihood that an agent will trigger the same rules in similar emotional contexts.

As shown in Fig. 4, the agent learns based on feedback from the environment and other agents. When the agent perceives an event, it assumes that it is a consequence of all rules triggered that turn, therefore the emotion associated with the perceived event influences all rule instances triggered that inference turn, which in turn influence the rule that created them, the goal that the rule helps achieve (if any) and all expressions part of the rule, by a different factor F for each influence type.

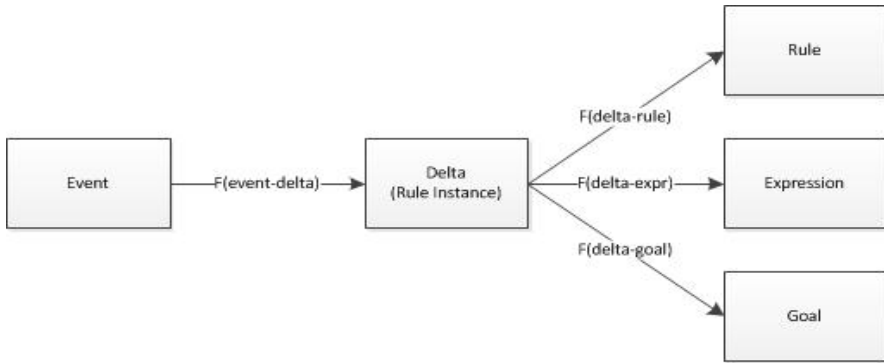


Fig. 4. Learning through emotion influence

6 Conclusion

One of the goals of this work is to demonstrate how emotions can affect agent attention focus, performance, learning, memory management and decision making and how the effects can vary according to the personality of the agent. Another is to provide a flexible and scalable emotion representation model that reproduces the effects of emotion observed in humans and that allows for a general dynamic approach to emotion in artificial agents.

Another goal of this work is to develop an emotion simulation model and agent architecture that provides artificial characters in virtual environments with believable behavior in order to enhance virtual environment experiences for human users.

The final goal of this paper, as presented in this paper, is to use artificial emotion simulation as motivation for agent behavior, as well as improve reasoning capabilities and memory management by simulating mechanics tied to emotions found in humans (see section 4.2.).

The main contributions of this model are that emotion is not only tied to the agent's learning and reasoning process, but it is a key component, providing agents with the underlayer for the simulated emotion, making the reasoning process more akin to humans, achieving complex, believable behavior for virtual agents. Another important contribution of this model is the improvement of agent performance and effectiveness through emotion simulation as it is currently believed emotions do in humans.

7 Future Work

There are at least two directions in which the model can be extended, both exploring the multi-agent capabilities of the model. First of all, the model can be extended to be used as a trust and reputation model among agents, based on the eight emotions. Another direction for future research, still in the field of multi-agent

systems, would be to extend the model so that it supports the theory of reciprocal altruism (where an agent acts in a manner that temporarily reduces its fitness while increasing another agent's fitness, with the expectation that the other agent will act in a similar manner at a later time).

References

1. Ortony, A., Clore, G., Collins, A.: *The cognitive Structure of emotions*. Cambridge University Press, Cambridge (1988)
2. Chwon, E., Jones, R.M., Henniger, A.E.: *An Architecture for emotional decision-making agents*. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy (July 2002)
3. El-Nasr, M.S., Ioerger, T.R., Yen, J.: *PETEEI: A Pet with Evolving Emotional Intelligence*. In: *Proceedings of the Third Annual Conference on Autonomous Agents*, Seattle, Washington, United States, pp. 9–15 (April 1999)
4. Lungu, V.: *Rule-based System for emotional decision-making agents*. In: *Sisteme distribuite*, Suceava (2009) (online) ISSN 2067-5259
5. de Sousa, R.: *Emotion*. *Stanford Encyclopedia of Philosophy* (September 2010), <http://plato.stanford.edu/entries/emotion/>
6. Lazarus, R.S.: *Emotion and Adaptation*. Oxford University Press, New York (1991)
7. Lungu, V.: *Artificial Emotion simulation model*. In: *Agents for Complex Systems Workshop on SYNASC 2010*, Timisoara (2010)
8. Sharot, T., Phelps, E.A.: *How arousal modulates memory: Disentangling the effects of attention and retention*. *Cognitive, Affective & Behavioral Neuroscience* 4, 294–306 (2004)
9. Kensinger, E.A.: *Remembering emotional experiences: The contribution of valence and arousal*. *Reviews in Neurosciences* 15, 241–251 (2004)
10. Jones, R.M.: *An introduction to cognitive architectures for modeling and simulation*. In: *Interservice/Industry Training, Simulation and Education Conference, I/ITSEC* (2004)
11. Langley, P., Laird, J.E., Rogers, S.: *Cognitive Architectures: Research issues and challenges*. Technical Report (2002)
12. Scherer, K.: *What are emotions and how can they be measured?* *Social Science Information* 44(4), 695–729 (2005)

Remote Monitoring and Control System for Environment Applications

Alexandru Dumitrașcu, Dan Ștefănoiu, and Janetta Culiță

Faculty of Automatic Control and Computers, University Politehnica of Bucharest,
313 Spl. Independenței, district 6, 060042 Bucharest, Romania
dumalex@ecosys.pub.ro, danny@indinf.pub.ro, jculita@yahoo.com

Abstract. This article treats the issue of remote monitoring and control of greenhouse environment through a distributed system that interconnects several subsystems. Thus, a wireless data acquisition subsystem, a fully automated subsystem made of PLCs and industrial communication networks and an irrigation subsystem consists of two water tanks, sensors and actuators are interconnected. According to data provided by sensors, the automatic control subsystem decides whether the irrigation cycle of plants should be performed. In this case, it provides corresponding commands to irrigation subsystem. Moreover, two graphical interfaces (*eKo-View* and *eKo-GreenHouse*) are added of the process and presented in detail in this paper.

Keywords: remote monitoring, remote control system, PLCs and HMI communication.

1 Introduction

This study of greenhouse microclimate is intended to complete the process of irrigation to keep plants at the appropriate level of development. In our process cycle, the first phase is represented by the acquisition and primary processing of data collected from a wireless sensor network. Central equipment of the sensors network is the eKo-gateway, which collects data and runs a visual monitoring application of the ecological phenomena. Thus, there is a graphical interface that allows the management of parameters and sensors network configuration, called *eKo-View* interface.

eKo-gateway equipment is connected via MPI (MultiPoint Interface) industrial communication bus with PLC S7-300 Simatic class, produced by Siemens[®]. This is the interconnection between first two subsystems: wireless sensors subsystem and automation control subsystem. The automation control subsystem includes multiple devices (PLCs S7-300 and LOGO!, OP 177B operator board for the local interface with the process parameters). These devices are connected together by different types of dedicated communication bus: Profinet, MPI, AS-I (Actuator and Sensor Interface) and I/O Link. In turn, the automation control subsystem

provides commands to the irrigation subsystem to perform proper watering plants. At this level, the actuators such as pump and electro-valves are controlled.

The distributed system can be managed remotely for monitoring and control via a second interface, named *eKo-Greenhouse*. This interface was implemented to create access to the process parameters, actuators, PLC S7-300, export data and log files of events. The two interfaces – *eKo-View* and *eKo-Greenhouse* – offer a series of highly useful capabilities that will be described in section 2. Also, in the next section will be detailed automation solution with all components of the three subsystems. The experimental results of the automated control system are presented in section 3.

2 The Automation Solution

For the purposes of an efficient implementation the chosen automation solution consists in three subsystems: the primary acquisition and processing of data is performed by a wireless sensors network manufactured by Crossbow® - The *eKo Pro Series* sensors network; the control tasks are performed by a network of programmable logic controllers and interface equipment manufactured by Siemens®. The final set of tasks, the irrigation itself is performed by a third subsystem.

The *eKo Pro Series* wireless sensors network presents itself as a set of nodes (Fig. 1) that wirelessly transmit data collected from sensors attached to them to a radio base. From here, the data gets stored into a single board computer: the eKo-gateway.



Fig. 1. The wireless node attached to a plant

This data acquisition subsystem is composed of 6 wireless nodes, one for each location with plants in the greenhouse. The eKo-node integrates an IRIS processor/radio board and antenna that are powered by rechargeable batteries and

a solar cell. The nodes themselves form a wireless mesh network that can be used to extend the range of coverage. By simply adding an additional eKo-node, it is easy to expand the coverage area. The nodes are pre-programmed and configured with XMesh low-power networking protocol, which provides plug-and-play network scalability for wireless sensor networks.

An eKo-node has four ports to connect the sensors. The system contains a variety of sensors (Fig. 2) such as soil moisture and temperature, ambient temperature and humidity, leaf wetness, soil water content and solar radiation. This solution provides an efficient wireless system for monitoring and acquisition data from multiple locations.



Fig. 2. The sensors of the eKo Pro wireless subsystem

In Table 1 are listed all the ecological parameters that can be monitoring using existing sensors. It is specified for everyone the variation range of measured values, measurement units, and an acronym, which will be used further.

Table 1. The ecological parameters of the sensors network

Soil	Leaf	Air
Moisture (Mo) 0 ... 240 [cbar]	Leaf Wetness (LeWe) 0 ... 1024 [CntS]	Humidity (Hu) 0 ... 100 [%]
Temperature (Te) -40 ... +65 [°C]		Temperature (Te) -40 ... +65 [°C]
Water Contents (WaCo) 0 ... 100 [%wfv]		Dew Point (DwPo) -10 ... 50 [°C]
		Solar Radiation (SoRa) 0 ... 1800 [W/m ²]

The base radio (Fig. 3a) is a fully integrated package that provides the connection between eKo-nodes and sensors and the eKo-gateway. The base radio integrates an IRIS processor/radio board, antenna and USB interface board which is pre-programmed with *XMesh* low-power networking protocol for communication with eKo-nodes. The USB interface is used for data transfer between the base radio and the *eKo-View* interface application running inside the eKo-gateway.

The eKo-gateway (Fig. 3b) runs the *Debian Linux* operating system and comes preloaded with sensor network management and data visualization software packages, *eKo-View* and *XServe*. These programs are automatically started when the gateway is turned on.

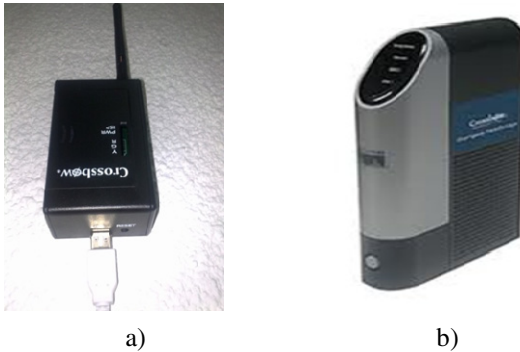


Fig. 3. The base radio (a) and the eKo-gateway (b)

The data from each sensor is sampled every 15 minutes. The software located in the eKo-gateway then computes average values for every hour, day and month for long term statistics and stores this data into a *SQLite* database that can be exported for processing by third-party applications or simply for backup.

The *eKo-View* application (Fig. 4) offers a familiar and intuitive web browser based interface for sensor network data visualization. In this interface is easy to start monitoring and data acquisition from anywhere in the world via a computer after the sensor network was configured. Through *eKo-View* interface, it is possible to setup and configure the wireless network to display only the data that are interested. The *eKo-View* web interface allows users to make various settings:

- create user-defined map view of sensor nodes across overall network;
- manage user-defined chart configurations;
- create trend charts of multiple sensors across customized time spans;
- view data real-time, which gives users the control needed to manage and maintain crop health;
- view details of individual sensor data;
- monitor performance of network and health of individual nodes;
- set alert levels, run report and get notifications via SMS or email;
- assign custom names to nodes and sensors.

Besides the default functionality presented, additional software was developed to allow the system to communicate the relevant data to the automation equipment: binaries for low level communication via the MPI interface (attached to an USB port of the eKo-gateway) and high level scripts for selecting and sending the data. Therefore, at the level of eKo-gateway equipment have been implemented many applications [1] such as a communication protocol to facilitate data transmission between the eKo-gateway and S7-300 PLC, and PHP scripts that represent the development base of the eKo-Greenhouse interface for remote monitoring and control the process parameters and systems devices.

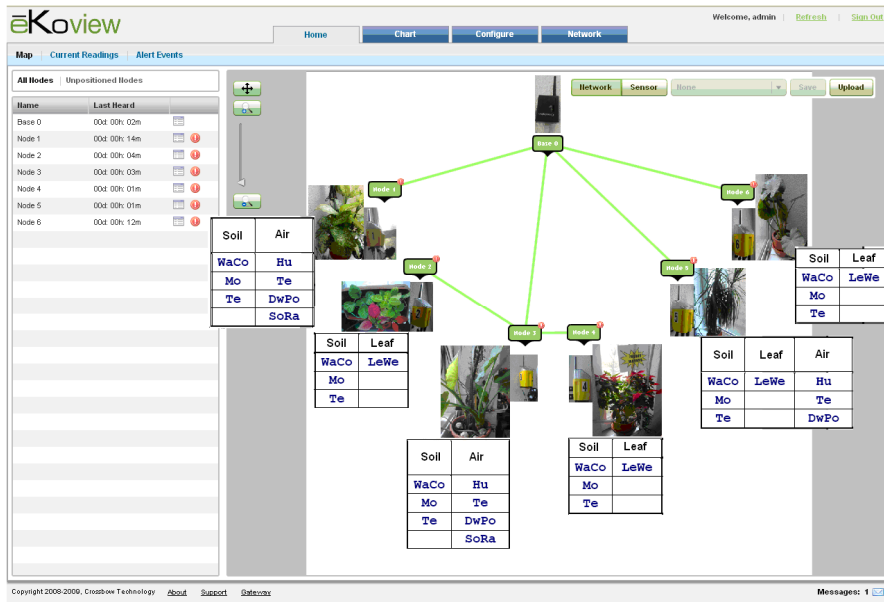


Fig. 4. The wireless sensor network and measured parameters

The control functions of the distributed system reside in the PLC network. An overview of this subsystem can be observed in figure below (Fig. 5).

The central unit in the control subsystem is the CPU315F-2DP/PN of the S7-300 Programmable Logic Controller. Here all the data from the eKo Pro data acquisition subsystem is collected via the MPI network.

Another industrial communications network used in the control subsystem is the Profinet / Industrial Ethernet network. It is used to accomplish two very important tasks: one is the programming of the CPU315F PLC and the other to transfer data between the CPU and the OP 177B HMI device.

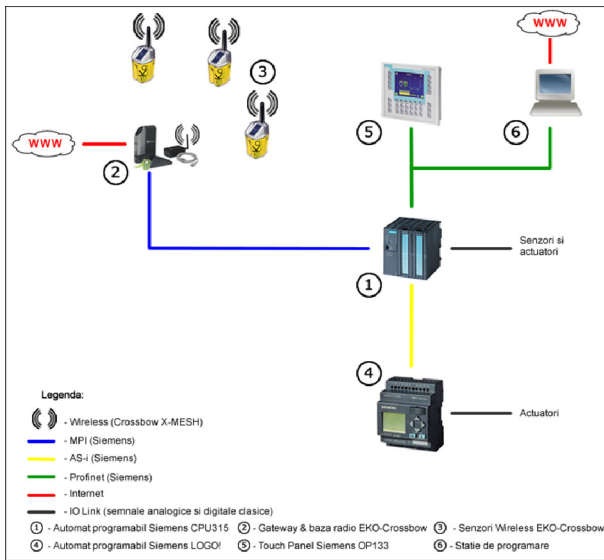


Fig. 5. The control system communication

Finally, the interactions with the next subsystem (the irrigations subsystem) are done with the help of a second PLC – a Siemens[®] LOGO! PLC – that receives its instructions directly from the CPU315F.

The two PLCs interact with the irrigations subsystem with basic I/O digital signals: the water levels in the two irrigation tanks are detected by digital sensors – low, middle and high level for each tank – that are connected directly to the CPU315F. Actuators – the three electro-valves and the pump – are controlled by the LOGO! PLC according to the instructions received from the CPU315F on the AS-interface network.

The most important piece of software in this subsystem is the control software for the CPU315F PLC. This program computes the average values across all the sensors and makes decisions on which of the valves to open so that the humidity values stay within preset limits. Another function of this program is to detect abnormal conditions and defects in the system and take corrective measures. The last of the tasks performed by the central PLC is to log its own actions so that problems are detected and corrected by the human operator [3].

The next important software resource is the program that runs on the OP177B touch panel. This allows the human operator to have a full view of the system and also to change important parameters in the system like the limits which trigger the opening of valves and the timings for the valves.

To improve system performance, at the level of the control subsystem some measures of reliability are considered. From this point of view, the causes that produce system failure are eliminated.

3 The Application's Results

The three subsystems proposed above – data acquisition subsystem, control subsystem, and irrigation subsystem – together form the complete architecture of the process, illustrated in the figure 6. This architecture is based on the measurement and data acquisition, management of database, monitoring and data processing, and then the *Master* device (S7-300 PLC) take decisions to control the irrigation system by sending commands to actuators via *Slave* device (LOGO! PLC).

In this architecture a webcam is used 24 hours a day for video monitoring of the process. For this task, another LOGO! PLC is used to control lighting enclosure. Thus, a lamp will be lit at night and goes off next morning.

For the irrigation process two tanks of water are used. The first tank into the water flow circuit is a buffer tank (T1), which is fed directly from the mains water supply. The second tank (T2) is used for the irrigation process of plants.

Before the first water tank a normal-open (NO) type electro-valve is installed to interrupt the general water supply in an emergency. Also, there is a manual tap with a mesh water filter, which is used to retain impurities. Inside the buffer tank are mounted some elements such as a float switch for water supply, three sensors for detecting water level in the tank (“min”, “middle” and “max” levels), and a mini-submersible pump. Also, the second tank has three floating level sensors that determine the minimum, middle and maximum levels of water. To fill the second tank with water the mini-pump is used. The irrigation process is performed because the second tank is located at a height of about 3 meters above the floor.

The irrigation process begins when a normal-closed (NC) type electro-vane receives command to open. This command is sent by the AS-Interface bus from S7-300 PLC to LOGO! PLC after the Master processes the sensors data. Water flows through the pipeline in a time set by user. At the end of the irrigation cycle, the electro-valve switches in the closed position again (default position).

If an extreme condition determines an emergency state of the system, the NO type electro-valve is closed in order to stop the water supply and three LEDs are lit.

The web interface to remotely control the automatic irrigation system was built using common web technologies: *HTML* (Hyper Text Markup Language), *JavaScript*, *AJAX* (Asynchronous JavaScript and XML), and *PHP* (PHP Hypertext Processor). This interface is protected. The access is done using a username and a password. Also, the web interface contains the image of process area provided by the webcam. Moreover, in the left side of the interface the last 10 events are precisely shown (date and hour). The remote control interface of the distributed system is presented in figure 7. The user has the possibility to observe and control the process parameters and system's devices.

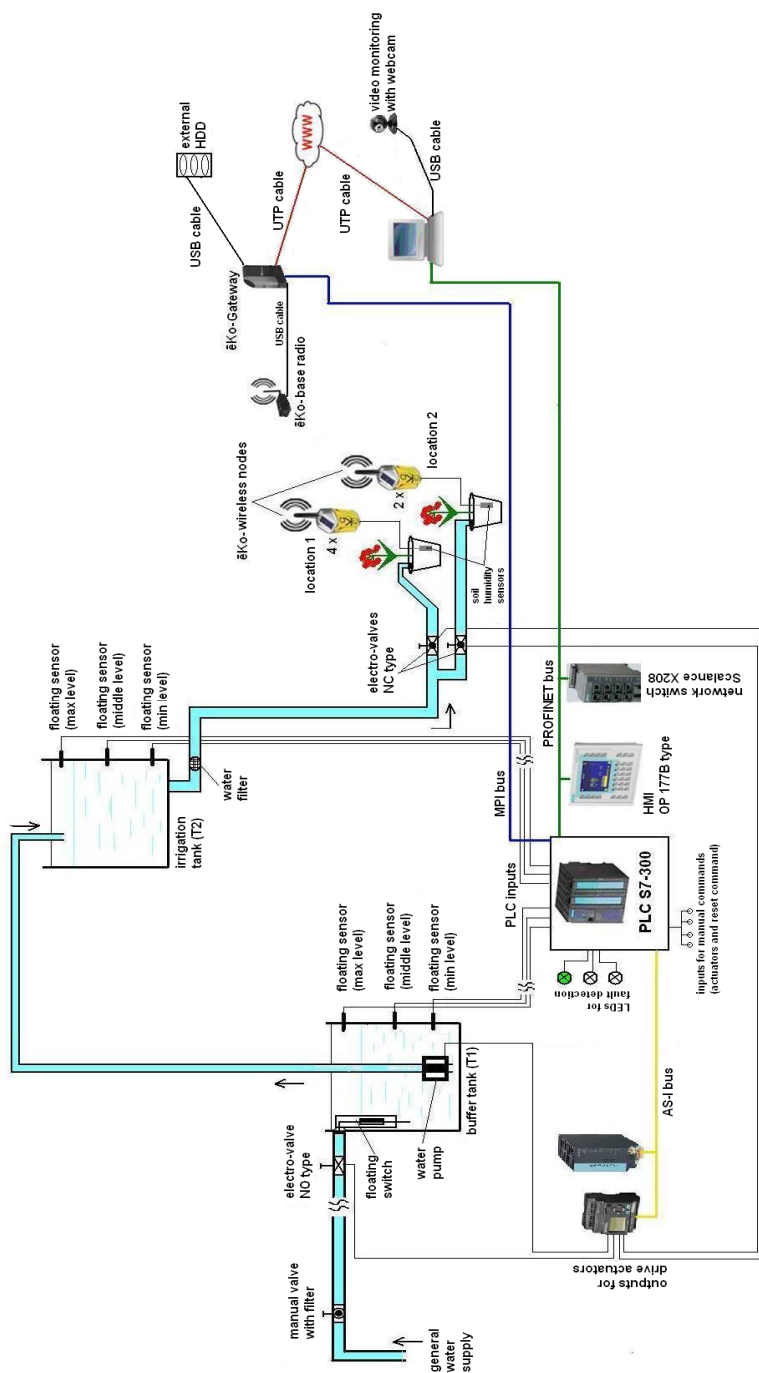


Fig. 6. The structure of the monitoring and control distributed system



Fig. 7. The web interface for remote process control

In this web interface the control panels are organised into 4 sections, each section containing a set of commands. The first section is shown in figure 8; it contains the commands of S7-300 PLC.

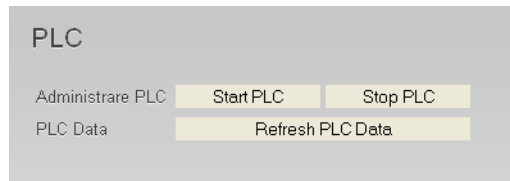


Fig. 8. The S7-300 PLC commands

The second section (Fig. 9) includes remote commands for manual control of actuators (two NC type electro-vanes and one pump) in irrigation process.

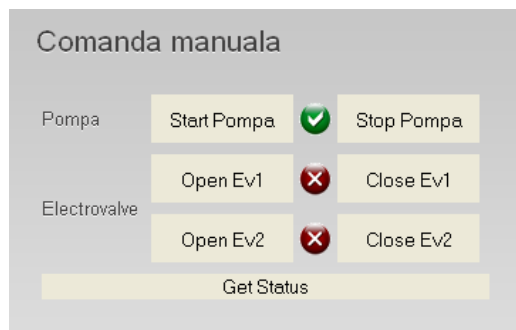


Fig. 9. The commands for manual control of actuators

The third section (Fig. 10) allows user to view and set the application parameters.

The screenshot shows a web interface titled "Variabile" with a table of parameters. The table has four columns: "Curent", "Prag", "Timp(s)", and "Comanda".

	Curent	Prag	Timp(s)	Comanda
Soil Moisture (cbar) SM				
Sala 1	4.58	12.00	Sala 1	<input checked="" type="radio"/> SM <input type="radio"/> SWC
Sala 2	9.17	9.00	25	
Soil Water Content (%WFV) SWC				
Sala 1	22.44	20.00	Sala 2	Set Get
Sala 2	19.87	22.00	20	

Fig. 10. The interface of application's parameters

The last section (Fig. 11) represents an easy modality to export data from the eKo-gateway in .csv type file. The data obtained in this section is useful for prediction algorithms [2] which consider data as input series.

The screenshot shows a web interface titled "Export". It contains a dropdown menu with "Soil Moisture & Temperature" selected, another dropdown with "Nod 1" selected, a "Start date:" field with a placeholder "aaaa-mm-dd hh:mm:ss", an "End date:" field with a placeholder "aaaa-mm-dd hh:mm:ss", and an "Export" button.

Fig. 11. The section for export data

The system had a great influence over the evolution of the relevant parameter (soil moisture), as best observed in figure 12 that shows the evolution graph for the soil moisture as captured by the sensors planted near the root of the six monitored plants.

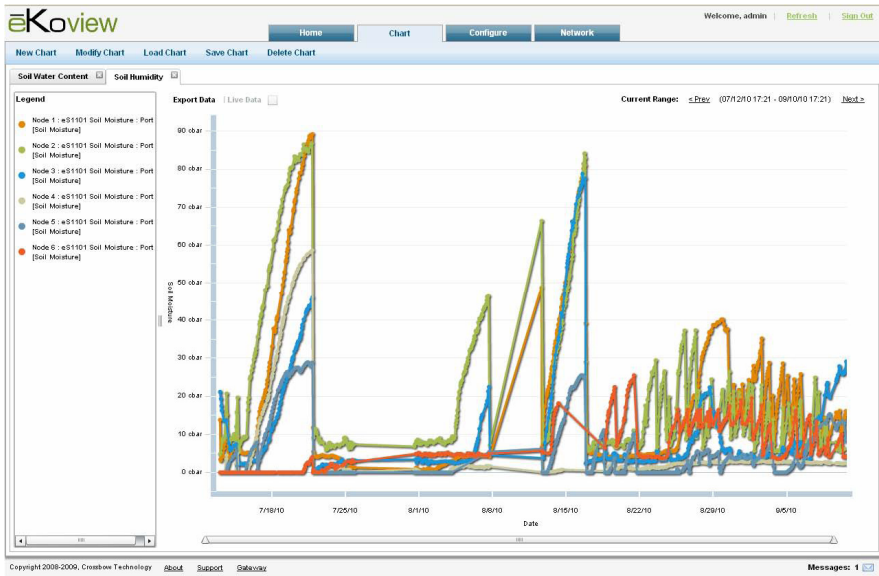


Fig. 12. The experimental results of the irrigation system

4 Conclusions

This paper addresses a very topical subject in the current issue of environment resources, distributed systems, data acquisition and data processing, process control etc. It is therefore submitted a comprehensive application that wants to be an efficient control system of the irrigation process in greenhouse environmental conditions. The implemented application provides a series of operations on process parameters. The first operation is acquisition of data using a wireless sensor network. Another important task is represented by the irrigation process control that is performed in two ways: locally of the process using HMI operator panel and remotely using a web interface that manage equipments and interested parameters' values. All the characteristics of the efficient process control are based on modern equipments, which are current in automation field. Also, regarding the aspect of communication between devices, have been implemented and configured line buses and protocols current used in industry.

The graph is provided by the original software found in the eKo-gateway. The first half represents the period of time in with the irrigation system was turned off. After switching the system on, the variation limits for the soil moisture were severely reduced, translating in better health and plants comfort, which is the purpose of the system, too.

References

- [1] Dumitraşcu, A.: Contributions to industrial computer networks in process control. PhD thesis, p. 218. "Politehnica" University of Bucharest (2010)
- [2] Ştefănoiu, D., Dumitraşcu, A., Culiţă, J.: Monitoring and prediction system for ecological phenomena, research report, p.166, CNMP.UPB-P4.31050-2007.IV/DS.AD.JC-11.2010 (2010)
- [3] Nitu, C., Dumitraşcu, A., Vinătoru, M.: Automatica, vol. 1, ch. 17. Editura Academiei Române, Bucureşti (2009)

H^∞ Control of an Induction Heating Inverter

Tibor Szelitzky and Eva-Henrietta Dulf

Technical University of Cluj-Napoca, Memorandumului str.28,
400114 Cluj-Napoca, Romania
{Tibor.Szelitzky,Eva.Dulf}@aut.utcluj.ro

Abstract. The industry strongly relies on PI controllers for induction heating inverters due to their simple design. Because correct load parameter estimation is almost impossible and their effect over electrical variables is not negligible, as it is presented in the paper, it is necessary a *variation tolerant control* algorithm. The present work proposes such advanced algorithms, based on different augmented plant models and on H^∞ design method. The paper also focuses on the practical approach of the controller implementation on an experimental pilot. The advantages of the robust controller are justified by simulation results and experimental data, as compared to the classical PI controller solution.

Keywords: induction heating, robust H^∞ control, pilot plant, embedded control.

1 Introduction

Induction heaters are modern, economic and environment friendly equipments. The heated work-piece is placed in an inductor, which generates high intensity electromagnetic field inducing eddy currents. Through Joule-Lenz effect of these currents heat is generated directly in the working material. For a given inductor and work-piece geometry, the heating profile strongly depends on the skin and proximity effects, which are influenced by the physical properties of the heated material [1].

For low power (<25kW) and high frequency (>10kHz) applications, inverters with series resonant load are preferred due to their simple design.

The output inverter variables (load current, developed power, capacitor voltage or phase shifts) can be controlled through inverter supply voltage, or through driving strategies of the power transistors of inverters.

For the present study the second approach was adopted, output variables being controlled by the inverter bridge, in order to avoid complex and expensive power supplies.

The industrial applications strongly relies on PI controllers, for frequency controlled inverters [2, 3, 4], and variable frequency PWM (Pulse Width Modulation) inverters [5].

Although PI controller studies are very common, are also known methods based on Fuzzy controllers [6] or adaptive controllers [7]. They have small impact due to the high amount of calculus implied by these advanced control algorithms.

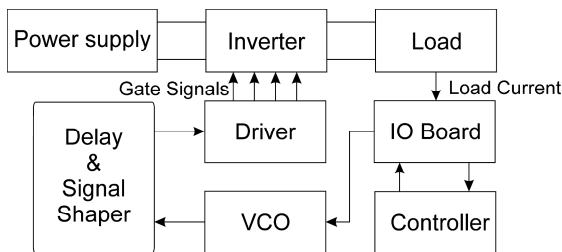


Fig. 1. Inverter block diagram

The general scheme of the induction heater is exemplified in figure 1. The inverter uses a DC power supply. The inverter generates a variable frequency square wave, which supplies the load. The values of the resulted load current are acquired and delivered to the controller which drives the power transistors through a VCO (Voltage Controlled Oscillator) and proper transistor drivers.

The present work conceives, design and implement a new version of a H_∞ robust controller dedicated to the above described process. After a short introduction, the paper justifies the difficulty of correct load parameter estimation and emphasizes the effect upon electrical variable, highlighting the necessity of a *variation tolerant control* algorithm. Based on conventional models, using the first harmonic of Fourier expansion, the next section develops an “upgraded” model with uncertain parameters, enabling the robust controller synthesis. Based on the augmented plant model, on the weighting functions and on the mixed sensitivity algorithm, the H_∞ controller is designed and implemented. This new controller will improve general performance of the heating equipment, compared to the usual, well-known controllers. Are emphasized the advantages of the H_∞ controller by comparing the simulation results with the experimental data of the laboratory induction heating equipment. The last section of the paper presents concluding remarks.

2 Load Parameter Estimation, and Their Influence on System Performance

2.1 Parameter Effects on Electric Variables

Inductor analytical design methods were developed in the first half of the XX century and became most popular in the 1960-1970s. The procedure is based on the similarities between the inductor and high leakage transformers. The design method is based on the use of correction factors, which asks for a strong knowledge and experience of the design engineer.

The development of the actual computing power leads to newer and easier circuit parameter estimation methods. The current possibilities of existing numerical methods and of the computers hardware enable the development of some advanced algorithms, the most used for inductor modeling being [8]:

- finite element method;
- boundary element method;
- mutual impedance method, etc.

These methods can be used together or separately in order to model electromagnetic fields and heat flow. Some dedicated programs for this kind of problems are: Ansoft’s MAXWELL, Consol, Flux, Elta and FEMM. For this study the open source FEMM-method (Finite Element Method in Magnetics) was used, which is dedicated to solve the steady-state electromagnetic and heat flow problems [9].

The major disadvantage of this approach relies in the appropriate mesh sizing. For this particular application, a dependency between mesh size and circuit impedance is shown in figure 2.

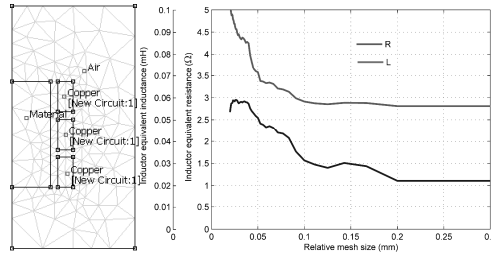


Fig. 2. FEMM meshing and equivalent circuit impedance

It can be observed a high variation of circuit parameters as function of mesh size, making difficult an accurate parameter estimation, which will make the controller design more unreliability. To minimize the possibility of system instability while ensuring fast dynamics, robust control of induction heating systems is considered a viable solution.

2.2 Parameter Effects on Load Current

It is a general observation that, during the heating evolution, the load’s circuit parameters vary. These variations are generated by the changes with temperature of the work-piece physical parameters, with effect on the skin and proximity phenomena. These parameter changes can affect current, power and phase shifts in the circuit. A secondary interdependency is established between the switching

frequency of the power supply and the load parameters. With the increase of the switching frequency the load's equivalent resistance grows while its inductivity decreases in accord to the inverse relation between frequency and skin depth. This influence is relative weak, but must be considered in the case of frequency controlled inverters.

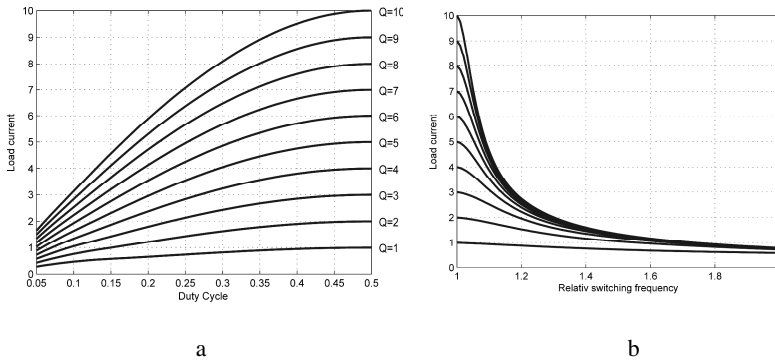


Fig. 3. Square wave duty ratio and switching frequency effects over load current for various loads

Almost all dependencies listed above can be incorporated in one single variable: circuit quality factor, which make possible the study of current, power, voltage and phase displacement evolutions.

Figure 3 presents the effects of the duty ratio and of the switching frequency on load current for different values of the quality factor, the data being normalized. If quality factor fluctuations arise (due to new batch or unequal temperature distribution), inductor current variations can occur. These can be compensated by changing the main input variables: square wave duty cycle and switching frequency, since these variables are controllable through the inverter. The increase of duty ratio increases the load current, while the frequency increase has a diminution effect. The circuit has a band pass filter characteristic, which implies electric variable attenuation in each direction of frequency deviation from resonant frequency. For a given resonant frequency f_0 , by frequencies greater than $1.7 \cdot f_0$, the current variations with frequency are insignificant. For desired lower current values, a secondary control strategy is required.

3 Inverter Modeling

The series RLC circuit with a quality factor over two and powered by an AC source presents band-pass filter properties. Therefore, if the circuit is excited with

a rectangular signal (signal with a infinite harmonic spectrum), the effects located in the pass band will be more pronounced than the effects of harmonics outside this band. Hence for model construction purposes the rectangular voltage power supply with variable frequency and duty cycle can be replaced by a sinusoidal voltage source with variable amplitude and frequency.

A demonstrative example is given in Figure 4. The series RLC circuit was excited using a rectangular wave and the first (sinusoidal) harmonic. Measuring the load current, acceptable differences can be observed for the two excitation cases. If load quality factor is further increased, the differences between the effects of the two excitations will decrease. This property maintains as far as the rectangular signal's frequency and band pass of the filter (load) coincide or are close to each other [10]. Based on this observation, several models have been deduced. The most flexible [11] and most convenient will be used to augment with uncertainties enabling the design of robust controllers.

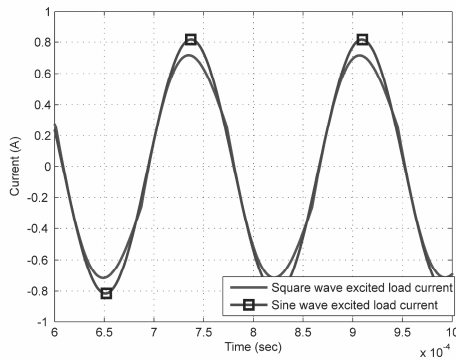


Fig. 4. Differences between square wave and sinusoidal wave excitation

For easier manipulation, the variable duty cycle rectangular signal is centered before Fourier decomposition to accomplish the anti-symmetry property, which enables the approximation of the original signal only through sine functions of form:

$$u_{AB} = \frac{4}{\pi} U_{dc} \cdot \sin\left(\frac{\pi}{2} \cdot \beta\right) \sin(\omega t) \tag{1}$$

Applying Kirchoff's law, it can be obtained the following equations:

$$\begin{aligned} u_{AB} &= L \frac{di_{AB}}{dt} + u_C + i_{AB}R \\ i_{AB} &= C \frac{du_C}{dt} \end{aligned} \tag{2}$$

Considering the load current and the capacitor voltage:

$$\begin{aligned} i_{AB}(t) &= i_{AB,\cos}(t) \cos(\omega t) + i_{AB,\sin}(t) \sin(\omega t) \\ u_C(t) &= u_{C,\cos}(t) \cos(\omega t) + u_{C,\sin}(t) \sin(\omega t) \end{aligned} \quad (3)$$

the matrix form of the model is:

$$\dot{x} = \begin{pmatrix} 0 & \frac{1}{C} & \omega & 0 \\ -\frac{1}{L} & -\frac{R}{L} & 0 & \omega \\ -\omega & 0 & 0 & \frac{1}{C} \\ 0 & -\omega & -\frac{1}{L} & -\frac{R}{L} \end{pmatrix} \cdot x + \begin{pmatrix} 0 \\ \frac{4}{\pi L} \sin(\frac{\pi}{2}\beta) \\ 0 \\ 0 \end{pmatrix} \cdot u \quad (4)$$

Choosing the system variables as:

$$x = [u_{C,\sin} \quad i_{AB,\sin} \quad u_{C,\cos} \quad i_{AB,\cos}]^T \quad (5)$$

and the system output variable the load current, defined as:

$$y_{MAX} = \sqrt{i_{AB,\sin}^2 + i_{AB,\cos}^2}, y_{RMS} = \sqrt{i_{AB,\sin}^2/2 + i_{AB,\cos}^2/2} \quad (6)$$

and taking into account the possible disturbances for all variables, results the following linearized system:

$$\begin{aligned} \dot{x} &= \underbrace{\begin{pmatrix} 0 & \frac{1}{C} & \omega & 0 \\ -\frac{1}{L} & -\frac{R}{L} & 0 & \omega \\ -\omega & 0 & 0 & \frac{1}{C} \\ 0 & -\omega & -\frac{1}{L} & -\frac{R}{L} \end{pmatrix}}_A \cdot x + \underbrace{\begin{pmatrix} 0 \\ \frac{4}{\pi L} \sin(\frac{\pi}{2}\beta) \\ 0 \\ 0 \end{pmatrix}}_{B_1} \cdot \hat{U} + \underbrace{\begin{pmatrix} 0 \\ \frac{2U}{L} \cos(\frac{\pi}{2}\beta) \\ 0 \\ 0 \end{pmatrix}}_{B_2} \cdot \hat{\beta} + \underbrace{\begin{pmatrix} U_{C,\cos} \\ I_{AB,\cos} \\ -U_{C,\sin} \\ -I_{AB,\sin} \end{pmatrix}}_{B_3} \cdot \hat{\omega} \quad (7) \\ y &= \underbrace{\begin{bmatrix} 0 & I_{AB,\sin}/\sqrt{I_{AB,\sin}^2 + I_{AB,\cos}^2} & 0 & I_{AB,\cos}/\sqrt{I_{AB,\sin}^2 + I_{AB,\cos}^2} \end{bmatrix}}_C x \end{aligned}$$

Since the paper is focused on frequency controlled inverter, the matrixes B_1 and B_2 , which are useful in PAM (Pulse Amplitude Modulation) and PWM (Pulse Width Modulation) control, respectively, are neglected.

To compute the system parameters in the static operational point, the Cramer rule can be applied as:

$$x_{\text{SOP}} = \frac{1}{A} \cdot B_3 \quad (8)$$

The linearization point is commonly chosen based on general usage of the equipment. Mathematically, for higher accuracy of the model over a wider range of switching frequencies and duty factors, it is recommended to set the linearization point for a duty factor of 25% and the switching frequency in the range (1.02-1.06*f₀).

4 Robust Control

The rapid development of computers in the last decades led the introduction of new concepts in system engineering. In the modern automation, are considered models which approximate the plant, but are taken into account also the differences between the model and the real plant. These models have to be simple to facilitate the analysis and controller design, but complex enough to include all plant behavior. [12] The differences between the mathematical model and the process are called "modeling errors" or "uncertainties" [13]. The presence of uncertainty is imminent, resulting from voluntary or accidental neglect of high frequency dynamics and un-modeled nonlinearities, from parameter variations and component tolerances, etc [14].

The aim of the robust control synthesis is to design a fixed structure controller which ensures stability and acceptable performances in the presence of disturbances and noise for the nominal system but also for the system family in the considered range of uncertainties. The easiest method to ensure this is to take into account the "size" of system family signals, which is represented by the norm of signals.

The norms are defined in Hilbert space, and if the system is linear and the output signals are bounded, the norms represent gains, amplifications [15]:

$$\|G\|_{(\infty, \infty)} = \sup_u \frac{\|y\|_{\infty}}{\|u\|_{\infty}} \quad (9)$$

Figure 5 represents the infinity norm of a frequency controlled inverter with RLC load, considering as output variable the load current. The maxima of this norm, indifferent of batch temperature, are obtained at the resonant frequency, where the circuit amplification is maxim. The maximum of the infinity norm is (1.032e⁻³) at the melting point of the work piece for a switching frequency of ~7.4kHz, while the minimum is obtained at room temperature at 15 kHz. The separate group of values from the right side of figure represents the system norms without work-piece where the maximum (4.129⁻³) was obtained at 6.9 kHz, and the minimum at 15 kHz.

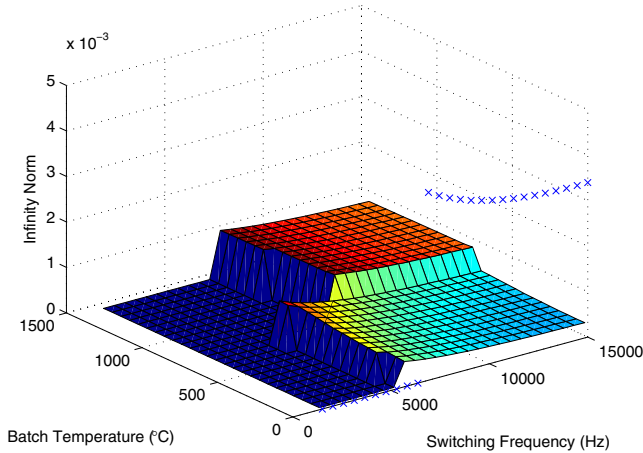


Fig. 5. Infinity norm of the uncontrolled system

4.1 Robust Controller Synthesis

Robust control was firstly developed for the defense industry, mainly for airplane and rocket guidance. These systems vary their flight characteristics and dynamics with the change of weight (cargo), altitude, speed etc. The controllers minimize the norm of the closed loop system, minimization which takes into consideration the system uncertainties. To design the controller, the uncertainties have to be introduced in the model in order to form the augmented plant model.

In the case frequency controlled inverters with load current as output variable the nominal model is based on equation 7 and is presented as block diagram in figure 6. The character X denotes all auxiliary parameters which depend by the working (linearization) point. For the augmented model the number of uncertainties can rise up to 9, depending on the model complexity: three variable parameters (L, R and ω) and six parameters depending on operation point, located in matrixes B and C of equation 7. [16]

To keep system order as low as possible, three different cases will be studied:

- system with two uncertainties: L and R
- system with three uncertainties: L, R and ω
- system with all nine uncertainties: L, R, ω and six linearization uncertainties.

The easiest way to introduce the uncertainties is by replacing the concerned parameters with upper LFT in uncertainty, figure 7.

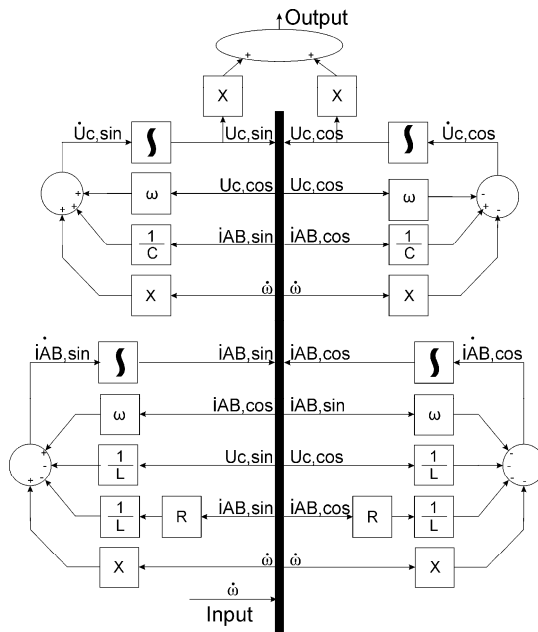


Fig. 6. Inverter block diagram

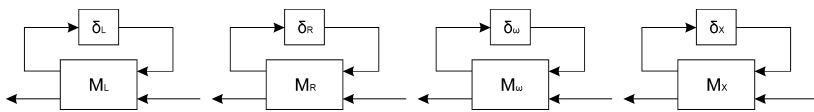


Fig. 7. Introduction of parameters with uncertainties

Two different uncertainty representations were used (additive and multiplicative) in each case in order to compare the performances of the controlled plants. The structures of these additive and multiplicative uncertainties are presented in figure 8.

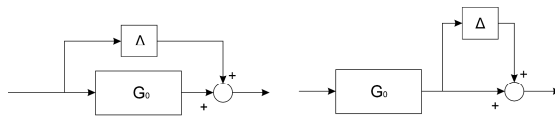


Fig. 8. Additive and multiplicative uncertainties

Depending on the used modeling approach, the uncertainty matrixes from figure 7 are the following:

- For additive uncertainties:

$$M_L = \begin{bmatrix} 0 & 1 \\ 1 & 1/L \end{bmatrix}; M_R = \begin{bmatrix} 0 & 1 \\ 1 & R \end{bmatrix}; M_\omega = \begin{bmatrix} 0 & 1 \\ 1 & \omega \end{bmatrix}; M_X = \begin{bmatrix} 0 & 1 \\ 1 & X \end{bmatrix} \quad (10)$$

- For multiplicative uncertainties:

$$M_L = \begin{bmatrix} -p_L & 1/L \\ -p_L & 1/L \end{bmatrix}; M_R = \begin{bmatrix} 0 & R \\ p_R & R \end{bmatrix}; M_\omega = \begin{bmatrix} 0 & \omega \\ p_\omega & \omega \end{bmatrix}; M_X = \begin{bmatrix} 0 & X \\ p_X & X \end{bmatrix}. \quad (11)$$

The augmented model of the process in all cases takes the standard form used in robust control:

$$\begin{aligned} \dot{x} &= Ax + B_1u + B_2w \\ y &= C_1u + D_{11}u + D_{12}w \\ z &= C_2u + D_{21}u + D_{22}w \end{aligned} \quad (12)$$

Two weighting functions are chosen (W_p and W_u) to represent the frequency characteristics of some external (output) disturbance d and performance requirement (including consideration of control-effort constraint) level, which modify the minimization problems to [17]:

$$\min \left\| \begin{array}{l} w_p (I + GK)^{-1} \\ w_u K(I + GK)^{-1} \end{array} \right\| \quad (13)$$

This approach is known in the literature as mixed sensitivity problem or S-KS problem. The performance weighing function is defined as a low pass filter amplifying the low frequency errors, while, W_u , the control signal weighting function, is defined as a high pass filter, ensuring the control signal limits.

$$w_p = \frac{K_p}{T_p s + 1}; w_u = \frac{K_u s}{T_u s + 1} \quad (14)$$

The controller design follows the algorithm described in [14]:

1. Define D_{1*}, D_{*1}, R_n and \tilde{R}_n as:

$$\begin{aligned} D_{1*} &= \begin{bmatrix} D_{11} & D_{12} \end{bmatrix} & D_{*1} &= \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} \\ R_n &= D_{1*}^T D_{1*} - \begin{bmatrix} \gamma^2 I_{m1} & 0 \\ 0 & 0 \end{bmatrix} & \tilde{R}_n &= D_{*1} D_{*1}^T - \begin{bmatrix} \gamma^2 I_{p1} & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \quad (15)$$

2. Solve the two Riccati equations

$$S = (I + GK)^{-1} \quad (16)$$

3. Construct the feedback and the observer gain matrixes

$$\begin{bmatrix} F_{11} \\ F_{12} \\ F_2 \end{bmatrix} = -R_n^{-1}(D_{1*}^T C_1 + B^T X) \tag{17}$$

$$[L_{11} \ L_{12} \ L_2] = -(B_1 D_{*1}^T + Y C^T) \tilde{R}_n^{-1}$$

4. Compute $Z, \hat{A}, \hat{B}_1, \hat{B}_2, \hat{C}_1, \hat{C}_2$

$$Z = (I - \gamma^{-2} Y X); \hat{A} = A + B F - \hat{B}_1 (C_2 + F_{12})$$

$$\hat{B}_1 = -Z L_2 + Z (B_2 + L_{12}) \hat{D}_{11}; \hat{B}_2 = Z (B_2 + L_{12}) \hat{D}_{12} \tag{18}$$

$$\hat{C}_1 = F_2 - \hat{D}_{11} (C_2 + F_{12}); \hat{C}_2 = -\hat{D}_{21} (C_2 + F_{12})$$

5. Build the central controller

$$K = \begin{bmatrix} \hat{A} & \hat{B}_1 \\ \hat{C}_1 & \hat{D}_{11} \end{bmatrix} \tag{19}$$

For each of the six cases (two uncertainty representation for each considered case), the weighting functions parameter are selected in accord to Table 1.

Table 1. Weighting function parameters

Nr of uncertainties	Type of uncertainties	Kp	Tp	Ku	Tu
2	Additive	8536.59	7.44	0.002	0.002
2	Multiplicative	4146.34	24.39	0.003	0.010
3	Additive	51219.5	5.61	0.000	0.009
3	Multiplicative	975.61	28.54	0.001	0.000
9	Additive	4512.19	91.46	0.007	0.439
9	Multiplicative	1524.390	280.488	0.001	1.000

The resulting structure of the H_∞ controllers is

$$H_{reg\infty} = \frac{b_5 z^5 + b_4 z^4 + b_3 z^3 + b_2 z^2 + b_1 z^1 + b_0 z^0}{a_5 z^6 + a_4 z^5 + a_3 z^4 + a_2 z^3 + a_1 z^2 + a_0 z^1 + a_0 z^0}, \tag{20}$$

with the parameters presented in Table 2.

Table 2. Controller parameters

Nr of un-cert.	Type of un-cert.	Numerator						
		b_5	b_4	b_3	b_2	b_1	b_0	
2	Ad.	-30.05	127.3	-229.3	219.32	-111.4	24.06	
2	Mult.	-215.70	477.2	-396.5	169.14	-38.329	3.700	
3	Ad.	-19.594	92.4	-176.59	170.61	-83.412	16.505	
3	Mult.	-7.495	-4.235	2.953	3.197	0.988	0.106	
9	Ad.	-0.828	3.915	-7.492	7.253	-3.553	0.705	
9	Mult.	-0.543	-0.209	0.349	0.304	0.089	0.009	
		Denominator						
	Ad.	a_6	a_5	a_4	a_3	a_2	a_1	a_0
2	Ad.	1	-4.32	7.890	-7.63	3.906	-0.85	0.000
2	Mult.	1	-2.23	1.870	-0.8	0.180	-0.02	0.000
3	Ad.	1	-5.376	12.15	-14.8	10.22	-3.80	0.596
3	Mult.	1	0.288	-0.68	-0.49	-0.11	-0.001	0.002
9	Ad.	1	-5.71	13.69	-17.6	12.89	-5.06	0.836
9	Mult.	1	-0.61	-1.03	0.083	0.396	0.147	0.017

5 Simulations and Experimental Results

In order to test each designed controllers, an experimental low power prototype induction heating inverter was built, figure 9. The block diagram of the control loop is presented in figure 10.



Fig. 9.The experimental plant

For this particular application the two BSM75GB120DN2 modules mounted in an H bridge connection is fed from a 30V DC power supply. Each power transistors is driven by a UCC37321, which receives the signals through two optocouplers.

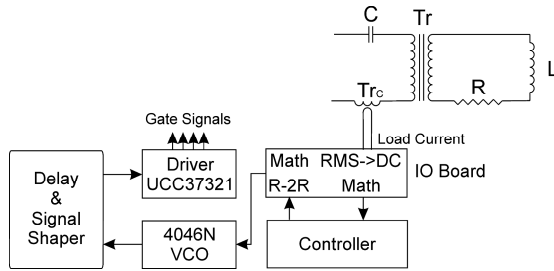


Fig. 10. The block diagram of the control loop

The two optocoupler approach was selected to ensure safe operations by requiring two signals (enable and gate signals) to open a transistor, minimizing the accidental short circuits. Prior to galvanic separations, a delay and signal shaper circuit was designed, which ensure the safe transition between open and blocked state of the power transistors. The pulse generator receives the driving signals from a 4046N's VCO (Voltage Controlled Oscillator) part, which obtains the control signals from a PIC32MX440F512H through an IO (Input Output) board. This interface converts the digital signals provided by the microcontroller through an R-2R ladder, and passes through a series of op-amps for convenient VCO control. The load circuit consists of three elements: matching transformer, capacitor and the inductor. The capacitor is mounted in series with the matching transformer, which has a 16:1 transformation ratio. The inductor is a three turn solenoid with an internal diameter of 60mm. The work piece is a 40mm iron rod inserted in the middle of the coil. In series with the primary winding, a current transformer is mounted, with a transformation ratio of 5:1. The feedback is fed to an AD736 RMS-DC converter, which output is introduced - after a few mathematical operations - in the analog to digital converters of the microcontroller. The mathematical operations are required due to AD736's negative output and the microcontroller's analog to digital converter low input range (0-3.3V).

The designed robust controllers are tested using Matlab® simulations. In figure 11.a are presented the step response of the closed loop system using the augmented models with nominal values and of the closed loop with PI controller and classical model, while in figure 11.b are presented the same step responses in worst case. It can be observed that in nominal case the PI controller is even better than the robust controller, but in the worst case simulation the robust controller remains stable, the performances depending on the used uncertainty model, while the system with PI controller became unstable.

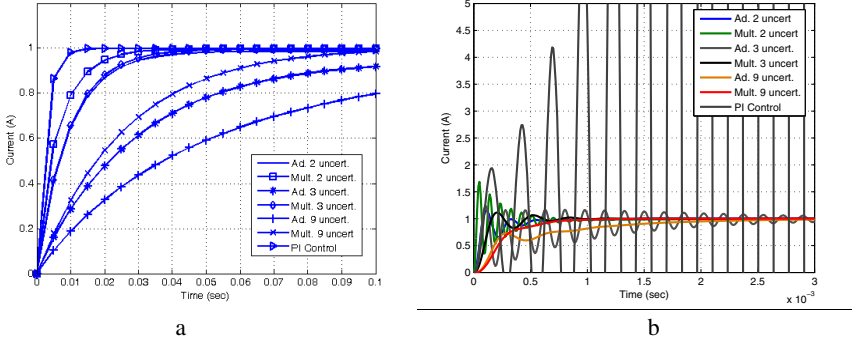


Fig. 11. Step response of the closed loop using augmented models with nominal values (a) and in worst case (b)

The same good performances of PI and robust controllers can be observed from figure 12 in the nominal case with the experimental plant.

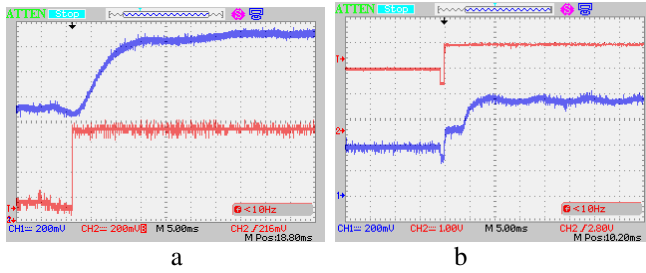


Fig. 12. Experimental step response of the closed loop using robust (a) and PI (b) controller with nominal parameters

Figure 13 presents the experimental worst case scenario. The experimental result obtained with the best robust controller is plotted in figure 13.a, highlighting the same stability as in simulation results.

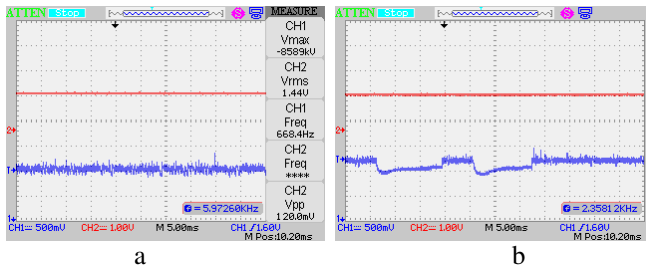


Fig. 13. Experimental step response of the closed loop using robust (a) and PI (b) controller in worst case

The output using PI controller has an oscillatory characteristic, due to electric protection equipment, figure 13.b. Without these safety features the output would be unstable, as in the simulation result.

6 Conclusions

In the literature are detailed different PI controller designs for the frequency controlled induction heating inverters with good results. Advanced control strategies are studied only in a few article and presented as control structures with difficult implementation possibilities. The present work proposed to overcome this gap and to design a series of robust H ∞ controller, based on augmented models with different complexity and two type of uncertainty (additive and multiplicative). The validation of the controllers is done using Matlab® simulations. The realized pilot scale experiments prove the viability of the proposed advanced control and ask for future research in the domain.

Acknowledgement. The authors are grateful to eng. Peter Bela for his support and contribution in the experimental part of the present work.

References

1. Zinn, S., Semiatin, S.L.: Elements of Induction Heating – Design, Control and Applications. ASM International (1988)
2. Espi, J.M., Navarro, A.E., Maicas, J., Ejea, J., Casans, S.: Control Circuit Design of The L-LC Resonant Inverter For Induction Heating. In: 31st Annual Power Electronics Specialists Conference, vol. 3, pp. 1430–1435 (2000)
3. Cui, Y.-L., He, K., Fan, Z.-W., Fan, H.-L.: Study on DSP-Based PLL-Controlled Superaudio Induction Heating Power Supply Simulation. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, vol. 2, pp. 1082–1087 (2005)
4. Calpe, J., Sanchis, E., Martinez, M., Esteve, V., Rosado, A.: DSP-Based Control For A Series Resonant Heating Generator. In: 28th Annual Conference of the Industrial Electronics Society, vol. 3, pp. 1960–1965 (2003)
5. Khan, I., Naylor, E., Tapson, J.: Optimised Control for High Frequency Induction Heating Power Sources. In: International Conference on Power Electronics and Drives Systems, vol. 2, pp. 1015–1018 (2005)
6. Tomșe, M., Pașca, P.D.: Fuzzy Control of Resonant Inverter for Induction Heating. In: 11th International Conference on Microwave and High Frequency Heating, University of Oradea, pp. 136–139 (2007)
7. Kwon, Y.-S., Yoo, S.-B., Hyun, D.-S.: Half-Bridge Series Resonant Inverter For Induction Heating Applications With Load-Adaptive PFM Control Strategy. In: 14th Annual Applied Power Electronics Conference and Exposition, vol. 1, pp. 575–581 (1999)
8. Rudnev, V., Loveless, D., Cook, R., Black, M.: Handbook of Induction Heating. CRC Press, New York (2002)

9. Finite Element Method Magnetics : Documentation: FEMM Reference Manual, <http://www.femm.info/Archives/doc/manual142.pdf>
10. Kelemen, A., Kutasi, N.: Induction-heating voltage inverter with hybrid LLC resonant load, the D-Q model. In: Pollack Periodica, vol. 2(1), pp. 27–37. Akadémiai Kiadó, Budapest (2007)
11. Grajales, L., Lee, F.C.: Control System Design And Small-Signal Analysis of a Phase-Shift-Controlled Series-Resonant Inverter For Induction Heating. In: 26th Annual IEEE Power Electronics Specialists Conference Record, vol. 1, pp. 450–456 (1995)
12. Festila, C., Both, R., Dulf, E.H., Cordos, R.: Adaptive Robust Stability for Extremum Control System with a modified Implementation. In: 18th International Conference on Control Systems and Computer Science, vol. 1, pp. 227–331. Politehnica Press, București (2011)
13. Dulf, E.H.: Robust Control – Case Studie (Romanian). Mediamira, Cluj Napoca (2007)
14. Gu, D.-W., Petrov, P.H., Konstantinov, M.M.: Robust Control Design with MATLAB®. Springer, London (2005)
15. Zhou, K., Doyle, J.C.: Essentials of Robust Control. Prentice Hall, Englewood Cliffs (1997)
16. Szelitzky, T., Dulf, E.H., Inoan, I., Festila, C., Neaga, A.O.: Robust Control in Frequency Controlled Induction Heating Inverters. In: 18th International Conference on Control Systems and Computer Science, vol. 1, pp. 298–301. Politehnica Press, București (2011)
17. Sánchez Peña, R., Sznaiier, M.: Robust Systems: Theory and Applications. John Wiley & Sons Inc., New York (1998)

Forecasting Energy Consumption in Dwellings

Nicoleta Arghira¹, Stéphane Ploix², Ioana Făgărășan¹, and Sergiu Stelian Iliescu¹

¹ Automatic Control and Computers Faculty, University Politehnica of Bucharest,
313 Splaiu Independentei, 060042 Bucharest, Romania

² Grenoble Institute of Technology,
46 Avenue Félix Viallet, 38031 Grenoble Cedex 1, France
arghira.nicoleta@gmail.com, stephane.ploix@grenoble-inp.fr,
{ioana,Iliescu}@shiva.pub.ro

Abstract. Energy consumption is a major issue nowadays. The importance of forecasting energy consumption from end-user to power system operator becomes more obvious than ever. The consumption in the residential sector represents a significant percentage in the total electricity demand in Europe and all over the world and it is expected to grow. So, the prediction of energy consumption becomes a key component in the management (e.g. power flow) of the electrical grid. This paper presents different methods for prediction of energy consumption of electrical appliances used in dwellings. A stochastic approach is used since forecasting the consumption for a single appliance is more difficult than predicting the overall consumption. Different basic predictors are presented and a stochastic predictor is proposed and tested according to a prediction precision criterion. The enhancement of forecast precision is done by segmentation and aggregation of data. Several experiments are conducted for different appliances in the house and the results are discussed.

Keywords: energy consumption, stochastic process, energy forecast, smart home.

1 Introduction

Energy consumption is a major issue nowadays. The biggest concern of power system operators is to maintain the balance between generation and load. Power grids today are controlling generation to match load at any particular time. If until now in peak demand periods the equilibrium was kept by cutting loads, the development of communication and technology gives the possibility of controlling the energy demand also. The load-following strategy becomes more difficult as more renewable generation is added to the grid. Intermittent renewable energy sources like wind and solar generation can't be scheduled and can't be predicted with certainty. So, in the last decade, the concept of demand dispatch gains importance all around the world. Demand dispatch is the capability to aggregate and precisely control (or dispatch)

individual loads on command. Unlike traditional demand response, demand dispatch is active and deployed at all the time, not just at peak times, [1].

The importance of forecasting energy consumption from end-user to power system operator becomes more obvious than ever. The control system used by the electrical grid operator has always had a forecast function for the demand at a large scale. But with the involvement of the demand side resources and taking into account the consumers comfort, the prediction has to be done at a smaller scale (e.g. electricity provider).

It was noticed that in terms of energy consumption, the residential sector represents an important part of the total electricity demand. In this context, a proper prediction of energy demand in dwellings sector is very important. A bottom-up approach [2] can be used: first, the prediction is done for each appliance in a home, then the forecast will be made for the total energy consumed in a home and, finally, a prediction can be made regarding the households supplied by a certain energy provider. It is more difficult to predict the consumption of each appliance than the overall consumption, but there can be a great save of energy when considering a dynamic demand side management. The energy savings depend on the type of appliance: some can be shut down, some can be postponed and some cannot be changed, [3].

The purpose of this paper is to predict the energy consumption in houses for the next 24 hours, as the energy price in the day-ahead market is set for each hourly interval with one day in advance. The prediction of the next day energy consumption for different services in a house is an important part of a home automation system as seen in [3]. [4] presents a household energy control system with three layers: anticipative layer, reactive layer and device layer. The anticipative layer is mainly concerned with the predictions of energy consumption.

Several papers propose methods for energy prediction, but few of them consider the energy consumption at a house scale and even less papers are regarding the forecast for some electrical devices in dwellings. In [5] a short term energy prediction at house level is done using with support vector machines. In [6] the forecast for some residential consumers is done for 24 hours or a week using neural networks, but still at house level. A prediction based on Bayesian Networks for a single appliance in dwellings is considered in [7]. An enriched learning algorithm which proposes a general way to take expert knowledge into account is shown in [8]. Although some conclusions can be drawn from this papers concerning the forecast of energy consumption for different appliances, a more general method has to be found.

This paper presents a stochastic method which predicts the consumption for electrical services in dwellings. Since the learning period is critical for the forecast, the best historical data interval is searched also. The method was tested for all the appliances for which data was available.

2 Energy Consumption in Smart Grids

This paragraph presents the smart grid and smart home concepts in order to better understand the importance of energy prediction for appliances in dwellings.

2.1 Smart Grid

The concept of smart grid appeared as an answer to the new power system challenges. Better control systems which include advanced protection functions [9], new measurement systems, improved communication and modern technology have to be proposed. Smart grid integrates the use of sensors, communications, computational ability and control in order to enhance the overall functionality of the electric power system, [10]. Smart grid initiatives seek to improve operations, maintenance and planning using modern technology in order to better manage energy use and costs, [11]. Many governments sustain modern networks in the global context of energy saving and environment issues. United States Department of Energy have defined the functions required for smart grids in [12]: the ability to heal itself; to motivate consumers to actively participate in operations of the grid, to resist attack, to provide higher power quality, to accommodate all generation and storage options, to enable electricity markets to flourish, to manage more efficiently the assets and costs. Figure 1 depicts the important components of the smart grid, [13].

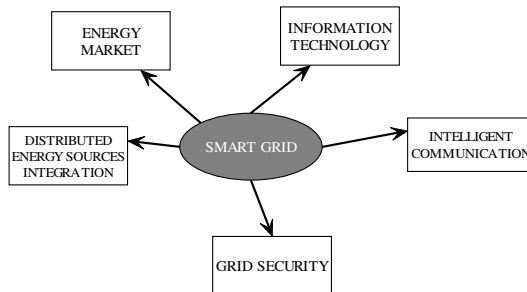


Fig. 1. The smart grid components

Considering the fact that energy resources are limited, there are many policies for energy saving on European and world scale. A good method to limit or change the energy consumption habits is the energy price. The energy market is a powerful instrument that sets the prices between the energy producers and energy suppliers and consumers (C1...C4), figure 2. It has an important role in the power system nowadays, but it is a complex bidding rules mechanism which sometimes is difficult to follow. The energy market is divided into different categories, but

the Day Ahead Market or Spot Market is of great interest. This type of energy market involves bidding the energy consumption of the next day. It is a very complex mechanism, which requires a very good knowledge of the demand for the power suppliers. The participation in the day ahead energy market imposes a dynamic energy management (hour by hour changes in the energy production/consumption ratio).

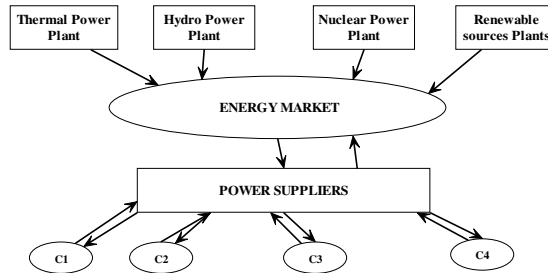


Fig. 2. Energy market in the power system

2.2 *Smart Home*

Smart homes are buildings equipped with a home automation system (HAS) able to optimize the energy management in a dwelling. A home automation system basically consists of household appliances linked via a communication network allowing interactions for control purposes. Thanks to this network, a load management mechanism can be carried out taking into account the users preferences. This system consists of a set of appliances fitted with micro-controllers able to communicate two-way via standard protocols.

The home automation system should be able to take the best decisions in order to meet the energy consumption needs of the consumers in an economical manner. Energy management problem can be formulated as scheduling problem where energy is considered as a resource shared by appliances, and device energy demands considered as tasks. [3] presents a three-layer household energy control system capable both to satisfy the maximum available electrical energy constraint and to maximize user satisfaction criteria. The proposed architecture contains: an equipment layer, a protection layer and an anticipation layer. The equipment layer, composed by existing control systems, is responsible of adjusting equipment controls in order to reach given set points in spite of perturbations. The protection layer is responsible of decision making in case of violation of predefined constraints dealing either with energy or with comfort. The anticipation layer is responsible of managing predicted events dealing with electric sources or with electric loads in order to avoid the use of protection layer.

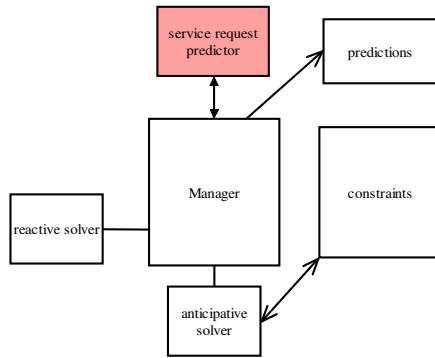


Fig. 3. Architecture of a power manager

2.3 Challenges for Energy Providers - Importance of Energy Prediction

Energy providers/retailers have the difficult task of dealing directly with the end-users and the electricity market. They have to take decisions in real time for a reliable and profitable operation of the grid (no congestions or load shedding). One method of getting the consumers controlled at all times is by changing the tariffs during the day. Figure 4 shows the interactions between the power retailers, consumers - smart homes and the energy market in the spirit of cost control.

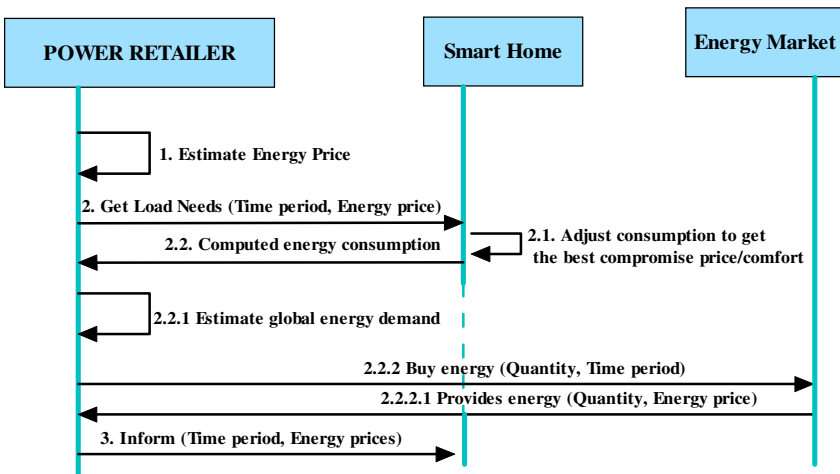


Fig. 4. Interactions power retailer - smart home - energy market

Considering the actions between the actors of the energy market at the load supply level, it is obvious the importance of load prediction in the smart homes. If dwellings are equipped with home automation systems capable of taking decisions, the estimation of energy demand will be easier. The prediction function of the HAS must give accurate forecast for the appliances in the house in order to build a reliable model and estimation on a larger scale.

3 Load Forecasting

Load forecasting occupies a central position in terms of planning and operation of electric utilities. Load forecasts are extremely important for energy suppliers, Transport System Operators (TSOs), financial institutions, and other participants in electric energy generation, transmission, distribution, and markets,¹³. This section presents a classification for load forecasting and the challenges of energy prediction for electrical appliances in dwellings.

3.1 Classification of Load Forecasting

When considering the time period of forecast a classification into three categories can be done: short-term forecasts which are usually from one hour to one day, medium forecasts which are usually from one day to a year, and long-term forecasts which are longer than a year. The forecasts for different time horizons are important for different operations within a utility company.

A review and categorization of electric load forecasting techniques is presented in [14-15]. There are explained several methods for energy forecasting, but none of them concerns a single appliance in a home. This paper tries to find a method to predict energy consumption for each device in a dwelling.

3.2 Issues Concerning Energy Forecast in Dwellings

Predicting the energy consumption in the housing sector is a difficult task because it varies a lot, depending on season, weather conditions and user behaviour. In the context of the home automation system, the forecast for each electrical equipment has to be done.

The energy consumption of electrical devices varies a lot during a year or even smaller periods of time. It is difficult to find similarities in terms of consumption between intervals of time. In order to find patterns to help with predicting energy consumption, the Euclidian distances are computed between each 2 days of the evaluated period data. Several equipments in houses were tested, but figure 5 shows the differences for the freezer in house 2000949. Some additional knowledge has to be produced for a proper forecast.

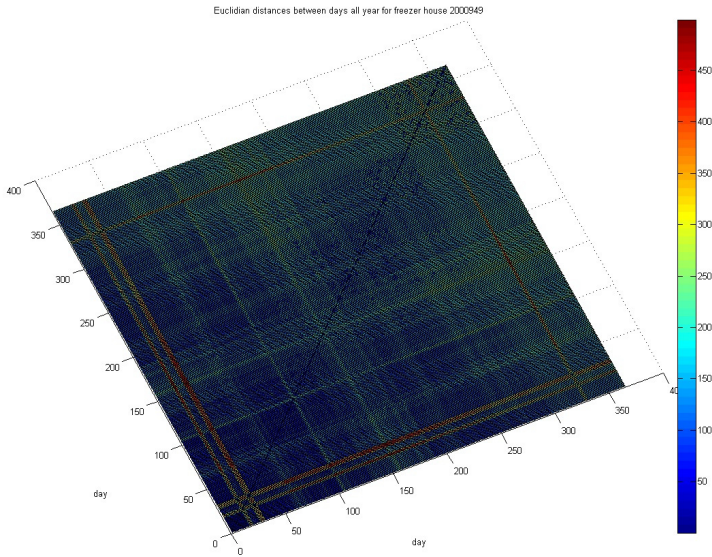


Fig. 5. Euclidian Distances between energy consumption values – freezer house 2000949

4 Forecasting Energy Consumption of Services in Dwellings

As seen in previous section, predicting the exact value of consumed energy is difficult, so the aim of this paper is predicting whether one service will consume energy or not during each hour of the next 24 hours.

The predictor performance is based on recorded data, which concern the energy consumption of appliances in 100 households in France during a full year. The used database comes from Residential Monitoring to Decrease Energy Use and Carbon Emissions in Europe (REMODECE), which is a European database on residential consumption. This database stores the characterization of residential electricity consumption by end-user and by country. The information for each house is recorded each 10 minutes and concerns the energy consumption for the appliances and also the weather conditions (temperature, wind strength, wind direction, humidity).

4.1 Computing the Prediction Precision

Before designing a predictor, it is important to set up a method for assessing the performance of a predictor because it clarifies the objectives. For computing the precision of a predictor, test data are first to be considered. Because of the predictions required by an anticipative energy management system, the test data are the hourly energy consumption for an appliance over a full year. Figure 6 shows a set of test data.

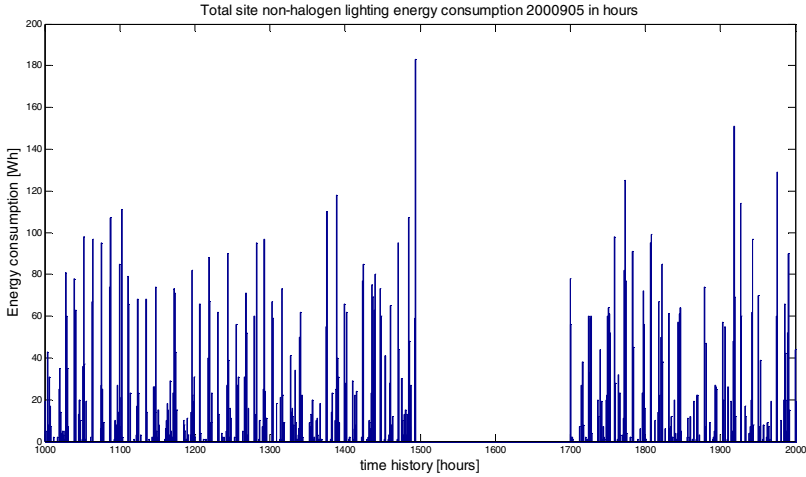


Fig. 6. Historical data for the energy consumption - non-halogen lighting in house 2000905

In order to evaluate the performance of predictors, some concepts have to be defined.

Let h be the current hour and $e(h)$ be a binary value which is equal to 0 if the considered appliance is actually consuming energy during the hour h and 1 otherwise. Let $p_a(h)$ be a prediction provided by the predictor a , which is equal to 1 if the considered appliance is predicted as consuming power during the hour h and 0 otherwise. The precision of the predictor is then expressed by:

$$\pi_a(h) = \sum_{i=1}^{24} \frac{(25-i) |e(h+i) - p_a(h, h+24+i)|}{275} \quad (1)$$

Any predictor a relies on an historical sliding time window of n hours used to predict the $d+1$ predictions. It can be denoted: $a^{h-n \rightarrow h}$. The number n has to be adjusted because if it is too large, seasonal phenomena may disappear, and if it is too short, data set will not be sufficient to yield a precise prediction.

The proposed algorithm for assessing a predictor a involves the following steps:

1. Set the time window dimension to n hours within the period for which the historical data was registered where n goes from 24 to $364 \cdot 24$;
2. Compute the predictions for the data corresponding to the historical sliding time window;
3. Compute the predictor precision $\pi_a(h)$ based on the “next day” data for all possible hours h and compute an average precision for the predictor.

4.2 Prediction with Basic Predictors

Since the information regarding the energy consumption is very dependent on user behavior, a stochastic approach will be tried. Two trivial predictors are tested: one

that considers that the service will consume all the time in the future and one which considers the service will never consume.

4.2.1 The “Will Always Consume” Predictor

This type of predictor involves considering that the appliance will consume energy permanently. The prediction is computed based on a set of test data and refers to the probability of the service to consume energy. The prediction $p_a(h, h+24)$ is expressed:

$$p_a(h, h + 24) = 1, \quad h = \{1, 2, \dots, 24\} \tag{2}$$

Figure 7 shows the prediction precision $\pi_a(h)$ for each time window of the test data considered in days (the prediction precision for a sliding window of 1 day, 2 days... etc.). This curve was obtained using the previously presented algorithm for assessing the predictor.

4.2.2 The “Will Never Consume” Predictor

This predictor assumes the service will not consume at all in the next day. The prediction is computed based on a set of test data and refers to the probability of the service not to consume). The prediction $p_a(h, h+24)$ is computed:

$$p_a(h, h + 24) = 0, \quad h = \{1, 2, \dots, 24\} \tag{3}$$

It can be denoted that the value of precision for this predictor is the complement of the “will always consume” predictor precision. Figure 8 shows the prediction precision for each time window of the test data considered in days for this predictor. Best precision is reached for a historical interval of approximately 100 days.

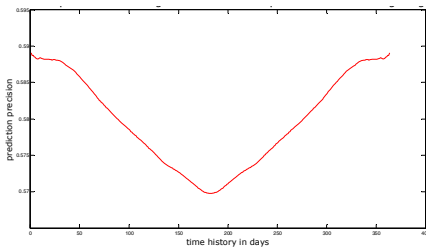


Fig. 7. Prediction precision assuming the service will consume continuously

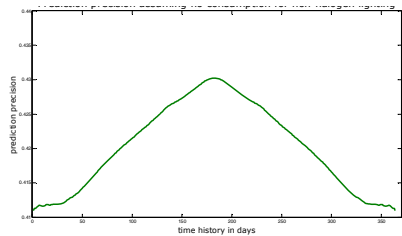


Fig. 8. Prediction probability assuming the service will never consume

4.3 The Proposed Predictor

An inhabitant in the house interacts with various electrical devices as part of his routine activities. Thus, energy consumption can be modeled as a stochastic process.

In this context, the proposed predictor specifies the probability of the appliance to consume on an hourly base. We consider the following prediction formula:

$$p_a(h, h+24) = \begin{cases} \frac{n_1(h, h+24)}{n(h)}, p_a(h, h+24) > p_{a,t} & , h = \{1, 2, \dots, 24\} \\ 1 - \frac{n_1(h, h+24)}{n(h)}, p_a(h, h+24) \leq p_{a,t} \end{cases} \quad (4)$$

Where $n(h)$ is the considered number of hours h in the test period, $n_1(h, h+24)$ is the number of times the service did consume during hour h of the historical data and $p_{a,t}$ is a set threshold. Figure 9 shows the prediction precision of the proposed predictor related to the basic predictors in the previous subsections.

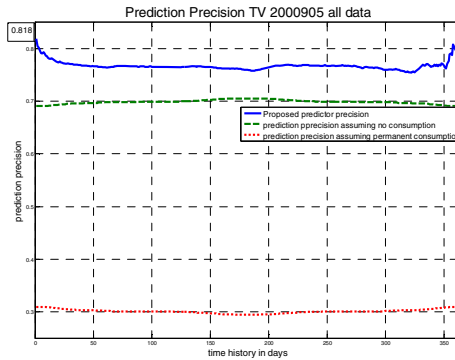


Fig. 9. Prediction precision using the proposed predictor

4.4 Enhancing the Forecast Precision

In order to increase the precision, some similarities between data are considered and clustering methods are applied.

4.4.1 Segmentation of Data

While mining the available data, some pattern of recurrence is searched in order to improve the prediction. A temporal segmentation can be used to introduce knowledge in the predictor, for instance, the use of the oven may be different for rainy Saturdays. The segmentation of data can be made considering different aspects such as the season, month, period of the day (day/night), type of day (weekday / weekend). The objective of this operation is to reduce the average dispersion in order to improve the prediction. After the segmentation is done, we will merge the segments that are similar using a clustering algorithm in order to gather the non-meaningful segments.

A temporal segmentation, that considers each day of the week as a partition was done. For each segment, the hourly predictions are made considering the proposed predictor. A k-Means clustering algorithm is applied in order to group the similar consumption days.

The K-Means algorithm assumes a fixed number of clusters, specified in advance. Each cluster is defined by its cluster center and clustering proceeds by assigning each of the input data to the cluster with the closest centre. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. This algorithm is based on the euclidean distance:

$$ED(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \tag{5}$$

Where X, Y are vectors, n is the vector length and x_i, y_i are their components.

The center of each cluster is then re-estimated as the centroid of the points assigned to it. The process is then iterated until a convergence criterion is accomplished, ED_t is a set threshold:

$$ED(X, Y) \leq ED_t \tag{6}$$

4.4.2 The Prediction Precision after Clustering

After applying the iterative k-Means algorithm, two clusters are obtained. In the presented case, cluster C_1 groups week days data and cluster C_2 gathers Saturday and Sunday data. After the clusters are obtained, the initial data corresponding to the energy consumption is divided into 2 sets according to the number of clusters and the considered segments. The prediction precision is computed for the proposed predictor for each of the clusters. Figure 10 *a and b* show the prediction precision for the new test data. The blue curve represents the precision for the proposed predictor and the other two are the curves for the basic predictors.

Figures 9 and 10 a, b present the prediction precision in 3 situations: for initial energy consumption data, for merged segments in cluster C_1 data and for merged segments in cluster C_2 data. When comparing the obtained curves for the proposed predictor, the precision of the predictor increases after the clustering is done. This proves that the segmentation of data and then the merging of similar partitions is a good method for increasing the prediction performance.

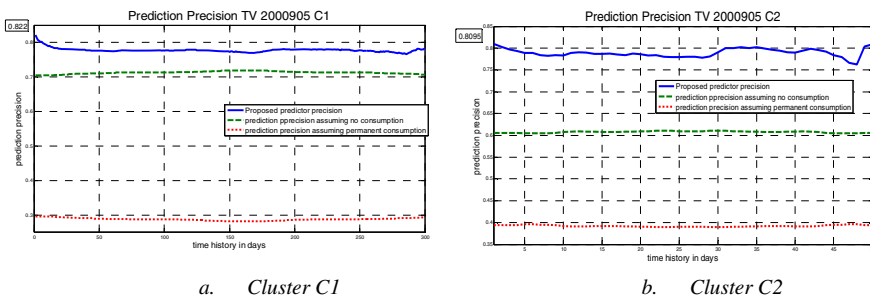


Fig. 10. Prediction precision comparison for the three predictors – clustered data

5 Experiments and Discussion

This section presents the results for different electrical appliances in the house when the proposed predictor is used in comparison with the basic predictors “will never consume” and “will consume continuously”. There are several services which were tested, but this paper will show the results for appliances representative for their class: the fridge, as it consumes all the time, the boiler, as its consumption can be delayed and the lighting, as it has a regular usage in a house.

The prediction precision is computed as explained in subsection 4.1 for a sliding time window between 1 and 364 days, covering all the historical data available. The tests show that the prediction acts in a special manner depending on the type of the electrical appliance, [16].

The prediction precision with the proposed predictor for the refrigerator (fig. 13) is lower than the precision assuming permanent consumption for time windows higher than 2 days. This implies that the prediction for the fridge should be done considering a short period of historical data (e. g. two weeks) in order to get a high precision. This conclusion was expected since the energy consumption for this appliance is dependent on the season, so a short period of time is significant for prediction.

The performance of prediction for the overall lighting consumption (figures 11) is higher than the performance of the basic predictors for the entire recorded data interval. For the gas boiler (figure 12), a good precision is obtained for almost all the considered periods of historical data.

As a general observation, the maximum value for prediction accuracy is obtained for a short period of historical data – almost in all studied cases the best precision was reached for one day of recorded data. Using this result will decrease the computation time since the learning time is short.

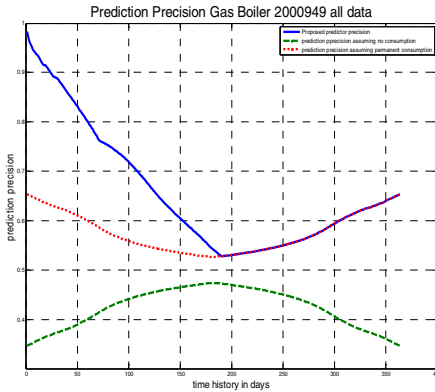


Fig. 11. Prediction precision of the boiler electricity consumption in house 2000949

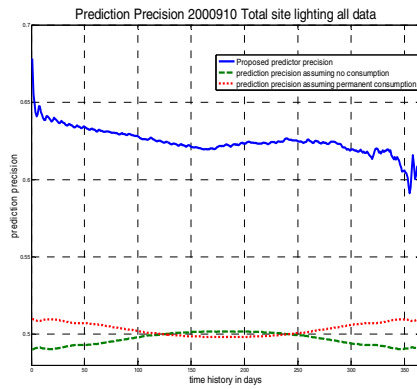


Fig. 12. Prediction precision of lighting consumption in house 2000910

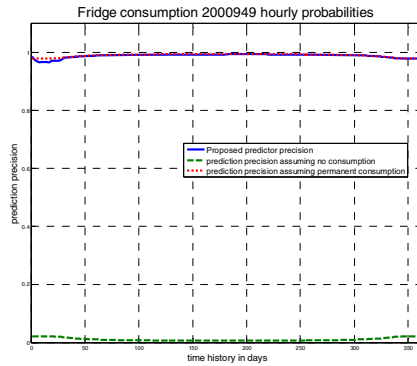


Fig. 13. Prediction precision of the refrigerator consumption in house 2000949

6 Conclusions

Predicting the energy consumption in dwellings is an essential part in the power management of the grid, as the consumption in the residential sector represents a significant percentage in the total electricity demand. The development of the smart grid is not possible without a good prediction of energy consumption. The trend nowadays is to get the prediction of energy consumption not only at house level, but at household appliance level.

The prediction of energy consumption in housing is very dependent on inhabitants' behavior, so a stochastic method for prediction has been presented in this paper. The paper discusses about how to evaluate the precision of a predictor in the *day+1* power management context. Different basic predictors are presented and tested for the available historical data. A relevant predictor is presented.

Segmentation of data is done considering the patterns in energy consumption. Also, the historical data is divided according to the results of the k-means clustering algorithm. After testing the predictor on the new clustered data, the precision of the predictor improves.

Further work involves testing the proposed predictor for all the appliances in a house in order to decide the proper way for prediction at the equipment level.

References

1. Brooks, A., Lu, E., Reicher, D., Spirakis, C., Wehl, B.: Demand Dispatch. IEEE Power and Energy Magazine 8(3), 20–29 (2010) ISSN 1540-7977
2. Iliescu, S.S., Fagarasan, I.: Modern approaches in power system control. In: IEEE Int. Conf. on Automation, Quality and testing, Robotics. Cluj, vol. I, pp. 41–44 (2008)
3. Long Ha, D., Ploix, S., Zamai, E., Jacomino, M.: Realtimes dynamic optimization for demand-side load management. International Journal of Management Science and Engineering Management 3(4), 243–252 (2008) ISSN 1750-9653

4. Abras, S., Ploix, S., Pesty, S., Jacomino, M.: A multi-agent design for a home automation system dedicated to power management. In: IFIP Conference on Artificial Intelligence Applications and Innovations, Greece (2007)
5. Jain, A., Satish, B.: Clustering based Short Term Load Forecasting using Support Vector Machines. In: IEEE PowerTech, Bucharest (2009)
6. Jigoria-Oprea, D., Kilyeni, S., Dan, F.: Electric energy forecast for residential users. *Journal of Sustainable Energy II*(2) (2011)
7. Hawarah, L., Ploix, S., Jacomino, M.: User Behavior Prediction in Energy Consumption in Housing Using Bayesian Networks, vol. 113, pp. 372–379 (2010), <http://www.SpringerLink.com>, doi:10.1007/978-3-642-13208-7_47
8. Basu, K., Guillame-Berty, M., Joumaa, H., Ploix, S., Crowley, J.: Predicting home service demands from appliance usage data. In: International Conference on Information and Communication Technologies and Applications ICTA 2011, USA (2011)
9. Popescu, V., Oprea, L., Costianu, D.R.: Transmission Network Capacity Enhancement by Special Protection Schemes. In: 18th Int. Conf. Control Systems and Computer Science, Bucharest, vol. 1, pp. 205–210 (2011) ISSN 2066-4451
10. Gellings, C.W.: The smart grid: enabling energy efficiency and demand response, USA (2009) ISBN 0-88173-624-4
11. Ipakchi, A., Albuyeh, F.: Grid of the future. *IEEE Power and Energy Magazine* 8(3), 20–29 (2009), doi:10.1109/MPE.2008.931384, ISSN 1540-777
12. ***, National Energy Technology Laboratory: A Vision for the Modern Grid (2007)
13. Arghira, N., Hossu, D., Făgărășan, I., Iliescu, S.S., Costianu, D.R.: Modern SCADA philosophy in power system operation - A survey. *Scientific Bulletin of University POLITEHNICA Bucharest, Series C: Electrical Engineering* 73(2), 153–166 (2011)
14. Feinberg, E.A., Genethliou, D.: Load Forecasting, Applied mathematics for restructured electric power systems, pp. 269–285 (2005), <http://www.SpringerLink.com>, doi:10.1007/0-387-23471-3
15. Alfares, K.H., Nazeeruddin, M.: Electric load forecasting: literature survey and classification of methods. *International Journal of Systems Science* 33(1), 23–34 (2002)
16. Arghira, N., Ploix, S., Făgărășan, I., Iliescu, S.S.: Prediction of energy consumption in homes. In: 18th Int. Conf. Control Systems and Computer Science, Bucharest, vol. 1, pp. 211–217 (2011) ISSN 2066-4451

Modelling and Composed Recursive Model Free Control for the Anaerobic Digestion Process

Haoping Wang¹, Boyko Kalchev², Yang Tian¹,
Ivan Simeonov², and Nicolai Christov³

¹ Automation School, Nanjing University of Science and Technology,
200 Xiao Ling Wei St, 210094 Nanjing, China

² The Stephan. Angeloff Institute of Microbiology – Bulgarian Academy of Sciences,
bl. 26, Acad. G. Bonchev St, 1113 Sofia, Bulgaria

³ LAGIS UMR CNRS 8219, Université Lille1 Sciences et Technologies,
59655 Villeneuve d'Ascq, France
hp.wang@njust.edu.cn, kalchev@microbio.bas.bg

Abstract. This paper presents a modelling and new composite recursive model free controller for trajectory tracking and disturbance compensation for the Anaerobic Digestion Process of cattle dung. The used model is on the basis of a fifth-order continuous anaerobic digestion model. And the proposed controller comprises a recursive model free controller based stabilization component and a time delay control based compensation component with recursive calculation structure which does not require any knowledge of the model parameters. Computer simulation examples illustrate the performance and robustness of the proposed approach.

Keywords: Anaerobic digestion, composed recursive controller, piecewise continuous systems, recursive model free controller.

1 Introduction

Environmental problems (ex. air and water pollution) and energy shortage are nowadays recognized worldwide issues. Several possibilities are available to treat these difficulties, from nowadays different treatment ways, biological processes are surely among the most used, sustainable and efficient systems. Anaerobic digestion (AD) which is a biotechnological process is widely used in life sciences, wastewater treatment and a promising method for solving energy shortage and ecological protection problems in agriculture and agro-industry [1]. In such kind of processes usually carried out in Continuously Stirred Tank bio-Reactors (CSTR), the organic matter is de-polluted by microorganisms into biogas (mainly methane CH_4 and carbon dioxide CO_2) and compost in the absence of oxygen [2], [3].

AD is a very unstable process with regard to the bioreactor operation. This is due to the complicated interactions between different microbial species, as well as to the

complex organic matter transformations affected by a variety of environmental factors [31]. In this context, mathematical models and sophisticated control algorithms are powerful tools for investigations and optimisation of the AD [19], [20], [32], [33], [34], [35].

Various control algorithms such as feed rate feedback stabilization method [5], methane-maximized-production optimal method [6], linear parameterizations based adaptive control [7]-[9] or nonlinear PI set-point regulation control [10] have been developed for control of this complex and strongly nonlinear process. For the plant limiting available information, different estimators or observers such as neural network model based method [11], [12] or Takagi-Sugeno fuzzy Observer [13] have been integrated with the control algorithms for plant coefficients estimation or state observation. Despite a long history practical experience [14] and decades of academic study [15]-[18], the control of these biological AD process seems still open. Many mathematical models of these processes in CSTR are known of nonlinear ordinary differential equations with a great number of coefficients that are hard to be estimated [2], [19], [20]. And a limited number of indexes characterizing the process are measurable online. Quite often one obtains only local solutions and it is impossible to validate the model in a large domain of experimental conditions [2].

In this paper a Composed Recursive Model Free Controller (CRMFC) is proposed for controlling a nonlinear five-order AD process and to study its performance by simulations taking into account a realistic stochastic environment of the process. The proposed control uses only the output measurement - the methane outflow rate, and is composed of a Recursive Model Free Controller (RMFC) based stabilization sub-control and a Time Delay Controller (TDC) based compensation sub-control.

The referred RMFC is developed by using hybrid systems called Piecewise Continuous Systems (PCS) which are characterized by autonomous switchings and controlled impulses [21]. Thus, based on PCS theory, a Piecewise Continuous Controller (PCC) was firstly developed, enabling sampled trajectory tracking of linear systems by undertaking a continuous time state feedback or a delayed and sampled output feedback. Then for an improved tracking performance, a Derived PCC (DPCC) with shorter sampling period close to zero was derived in [22]. PCS can be equally used in designing a piecewise continuous observer for visual servo control of non-minimum phase mechanical systems [23], [24].

The used TDC which is firstly proposed in [25]-[26] and then developed in [27] employs past observations of the system output response and control inputs to directly compensate the controlled plant unknown dynamics and disturbances.

The paper is organized as follows. Section 2 describes the used AD experimental platform description, experimental data studies and mathematical model used to represent the dynamics of the AD bio-process. This is followed, in Section 3, by the development of the composed recursive model free controller by using only the system output measurement. Then, numerical results are discussed in Section 4 and finally some concluding remarks are given in section 5.

2 Experimental Platform Description and Mathematical Modelling

2.1 Experimental Platform

Experimental studies of AD of cattle dung have been performed during the start-up of a pilot anaerobic plant (with full volume of the bioreactor of 100 m^3) at mesophilic temperature ($34 \text{ }^\circ\text{C}$). The scheme of this pilot plant with a system for monitoring and control is shown on Fig. 1.

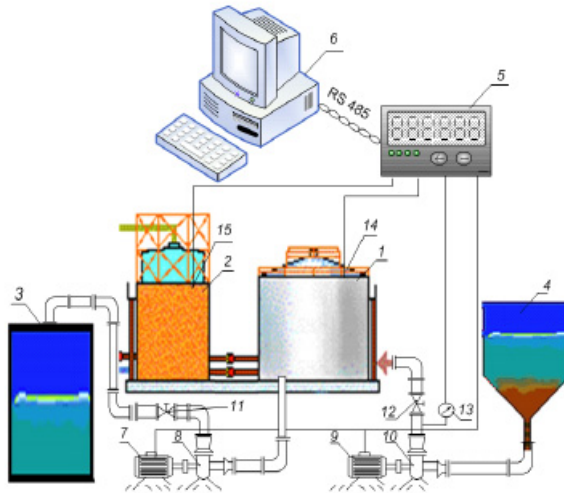


Fig. 1. Scheme of the pilot plant with a system for monitoring and control (1 - bioreactor (BR); 2 - gasholder; 3 - reservoir for digestate, 4 - vessel for substrate; 5 - controller Beckhoff; 6 - personal computer; 7, 9 - drives; 8, 10 - pumps; 11, 12 - stop valves; 13, 14, 15 - sensors for t_o , flow rate of biogas (Q), contents of CH_4 and CO_2 in the biogas).

The pilot anaerobic BR has been started and operated in continuous mode with different values of the dilution rate (D) and of the concentration of dry matter in the influent (S_{in}). Some experimental data from a very particular experiment with pulse-wise changes of D for different values of S_{in} (described in Table 1), are presented on Fig. 2 (with Q the daily biogas flow rate per 1 dm^3 of the working volume of the BR) and Fig. 3.

From Fig. 2 one may conclude that the community of microorganisms in the BR reacts immediately on pulse-wise changes of the control input (D) and different values of the concentration of dry matter in the influent (S_{in}). From the evaluations of CH_4 and CO_2 concentrations in the biogas, presented on Fig. 3, one may conclude that the ratio CH_4/CO_2 in the biogas for the above described experiment is practically not sensible to pulse changes of the control input (D) and different values of the concentration of dry matter in the influent (S_{in}).

Table 1. Experiment change conditions for inputs of D and S_{in}

Time day	$D \text{ day}^{-1}$	$S_{in} \text{ g/dm}^3$
0	0.01	69
1	Pulse ₁ =0.025	69
9	Pulse ₂ =0.02	69
16	Pulse ₃ =0.0225	86
30	Pulse ₄ =0.025	40

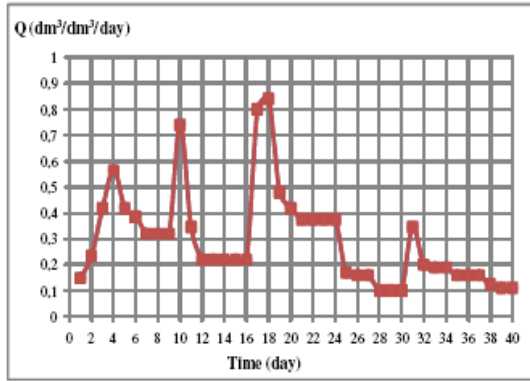


Fig. 2. Specific daily biogas flow rate evaluation during the experiment described in Table 1

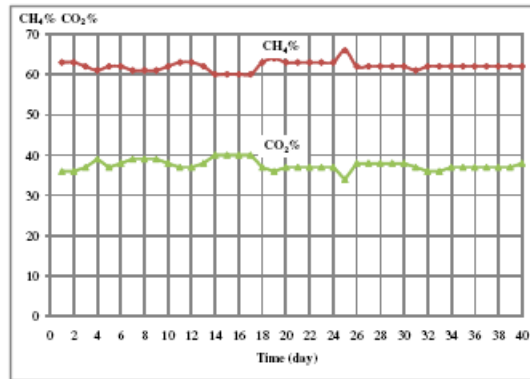


Fig. 3. Evaluations of CH_4 and CO_2 in the biogas during the experiment described in Table 1

2.2 Mathematical Modelling of the Process

AD processes can be divided in four main stages:

- hydrolysis of undissolved high-molecular weight compounds (proteins, sugars, fats) to soluble low-molecular weight compounds (monosugars, aminoacids, long - chain fatty acids, glycerol);
- acidogenesis – fermentation of low-molecular weight compounds from the previous stage to VFA (propionate, butirate, acetate), hydrogen and carbon dioxide;
- acetogenesis – transformation of VFA to acetate, hydrogen and carbon dioxide;
- methanogenesis mediated by acetoclastic methanogens (converting acetate to methane and carbon dioxide) and hydrogenotrophic methanogens (producing methane from hydrogen and carbon dioxide).

The key metabolites for methanogenesis – hydrogen, carbon dioxide and acetate, could be utilized as carbon sources not only from methanogenic Archae but also from other microorganisms. For example, hydrogen and carbon dioxide are used from homoacetogenic bacteria for acetate synthesis; alternatively, the acetate can be transformed to methane via syntrophic acetate oxidation [33]. The syntrophic acetate oxidation is a two-stage process mediated by two different phylogenetic microbial groups, living in syntrophic consortia: acetate oxidizing Eubacteria which convert acetate to carbon dioxide and hydrogen, and hydrogenotrophic methanogenic Archae which use carbon dioxide and hydrogen for methane production. Including this phenomenon in an AD model explains some process failures and restarts [36]. ADM1 [33] is the most complex and powerful AD model. However, such kind of models are very complex for control algorithm design. That is why model-free control algorithms are preferable in practice.

On the basis of the above-presented experimental investigations and according to the relatively simple three-stage biochemical scheme of the AD, the following 5th order Barth-Hill nonlinear model with one control input is further used for simulation purposes:

$$\frac{dS_0}{dt} = -bX_1S_0 + DY_pS_{in} - DS_0 \quad (1)$$

$$\frac{dX_1}{dt} = (\mu_1 - D)X_1 \quad (2)$$

$$\frac{dS_1}{dt} = -k_1\mu_1X_1 + bX_1S_0 - DS_1 \quad (3)$$

$$\frac{dX_2}{dt} = (\mu_2 - D)X_2 \quad (4)$$

$$\frac{dS_2}{dt} = -k_2\mu_2X_2 + Y_b\mu_1X_1 - DS_2 \quad (5)$$

$$Q = Y_g\mu_2X_2 \quad (6)$$

in which

$$\mu_i = \frac{\mu_{mi}S_i}{k_{si} + S_i} \quad i = 1, 2. \quad (7)$$

In this mass balance model, equation (1) describes the hydrolysis in a very simple way, where the first term reflects the hydrolysis of the diluted organics by acidogenic bacteria, the second term - the influent flow rate of liquid with concentration of the diluted organics S_{in} , g/dm^3 (presenting a sum of a constant component S_{io} and a disturbance $w(t)$) and the third one - the effluent flow rate of liquid. Equation (2) describes the growth and changes of the acidogenic bacteria (with concentration X_1 , g/dm^3), consuming the appropriate substrate (with concentration S_1 , g/dm^3). The mass balance for this substrate is described by (3), where the first term reflects the consumption by the acidogenic bacteria, the second term - the substrate S_0 , g/dm^3 , formed as a result of the hydrolysis, the third and the last one - the substrate S_1 , g/dm^3 , in the effluent flow rate of liquid. Equation (4) describes the growth and changes of the methane producing (methanogenic) bacteria (with concentration X_2 , g/dm^3), consuming acetate (with concentration S_2 , g/dm^3). The mass balance equation for acetate in (5) has three terms in his right side. The first one reflects the consumption of acetate by the methanogenic bacteria, the second one - the acetate formed as a result of the activity of acidogenic bacteria, and the third one -the acetate in the effluent liquid.

The algebraic equation (6) describes the formation of biogas with flow rate Q ($\text{dm}^3\text{gas/dm}^3$ medium/day) measurable with error (measurement noise) $v(t)$. The relations (7) present the specific growth rate of the acidogenic bacteria μ_1 , day^{-1} , and the specific growth rate of the methanogenic bacteria μ_2 , day^{-1} , both of Monod type. b , Y_p , k_1 , k_2 , Y_b , Y_g , μ_{m1} , μ_{m2} , k_{s1} , k_{s2} are coefficients. D , day^{-1} , is the dilution rate - the control input.

3 Composed Recursive Model Free Controller Design

3.1 Controller Design

In this section a new model free controller is proposed to solve the nonlinear AD control problem. The proposed controller is defined as

$$D(t) = G^{-1}(D_s(t) + D_c(t)) \quad (8)$$

where $D_s(t)$ is a stabilization sub-controller, $D_c(t)$ is a sub-controller for compensation of unknown system nonlinearities and $G \in R^+$ is an imposed constant. Both $D_s(t)$ and $D_c(t)$ are determined using recursive calculation loops. Since the model parameter informations are not needed, the proposed controller is called Composed Recursive Model Free Controller (CRMFC). The CRMFC realization diagram is given in Fig. 4. As shown in Fig. 4, the stabilization control component $D_s(t)$ is computed by

$$\lambda(t) = |e(t)|e(t) + \zeta(t)\lambda(t) \tag{9}$$

$$u(t) = \gamma\lambda(t) \tag{10}$$

where $\lambda(t) \in R$ is the stabilization sub-controller state, $e(t) = Q_r(t) - Q(t)$ is the output trajectory tracking error with $Q_r(t) \in R$ the desired output flow rate, $C_c \in R^+$ is the sub-controller output gain and $\zeta(t)$ is the sub-controller tracking

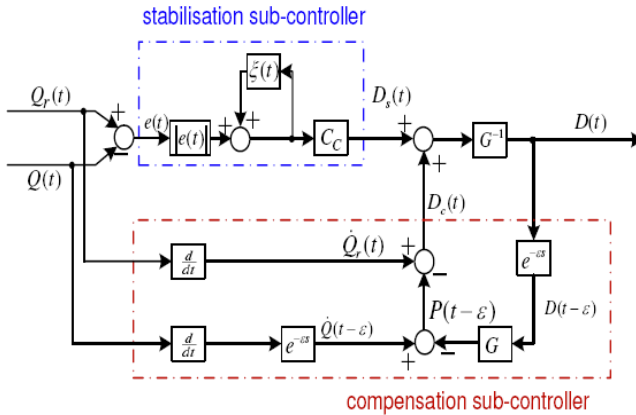


Fig. 4. Composed Recursive Model Free Controller

coefficient. To realize $e(t) \rightarrow 0$, the value of $\zeta(t)$ is tuned as

$$\zeta(t) = \exp(-e(t)^T e(t) / (2\sigma^2)) \tag{11}$$

with $0 < \sigma \leq 1$.

The compensation sub-control $D_c(t)$ is determined by using time delay estimation techniques [25]-[27] as

$$D_c(t) = \dot{Q}_r(t) - P^*(t) \tag{12}$$

where $P^*(t) = P(t - \varepsilon)$ is an estimate of

$$P(t) = f(x, t) + (g(x, t) - G)D(t) \tag{13}$$

and $f(x, t)$ and $g(x, t)$ are defined from the derivative equation of (6) which is written under the following form:

$$\frac{dQ}{dt} = f(x, t) + g(x, t)D(t) \tag{14}$$

For $\varepsilon \rightarrow 0$ one has

$$P(t) \cong P^*(t) = P(t - \varepsilon) = GD(t - \varepsilon) - \dot{Q}(t - \varepsilon). \tag{15}$$

3.2 Stability Analysis

It can be shown that the nonlinear closed-loop system (1) - (6) and (8) is input-to-state stable [28] and ensure the tracking of desired output trajectory. We have:

$$\begin{aligned} e(t) &= \dot{Q}_r(t) - \dot{Q}(t) \\ &= \dot{Q}_r(t) - (f(x, t) + g(x, t)D(t)) \\ &= \dot{Q}_r(t) - \underbrace{(f(x, t) + (g(x, t) - G)D(t))}_{P(t)} - GD(t) \\ &= \dot{Q}_r(t) - P(t) - GG^{-1}(D_s(t) + D_c(t)) \\ &= \dot{Q}_r(t) - P(t) - D_s(t) - \underbrace{D_c(t)}_{\dot{Q}_r(t) - P^*(t)} = -D_s(t) + P(t) - P^*(t) \end{aligned} \tag{16}$$

Since $\lim_{\varepsilon \rightarrow 0} (P(t) - P^*(t)) \rightarrow 0$, the following approximation of $\dot{e}(t)$ is valid:

$$\dot{e}(t) \cong -D_s(t). \tag{17}$$

In turn, $D_s(t)$ can be approximated as [29]

$$D_s(t) \cong \frac{2C_c \sigma^2}{|e(t)|} e(t). \tag{18}$$

Replacing (18) into (17), one gets

$$\dot{e}(t) \cong -\frac{2C_c \sigma^2}{|e(t)|} e(t). \tag{19}$$

which is stable.

4 Numerical Results

The previously refereed model parameters values are: $b = 3$, $Y_p = 0.144$, $S_{i0} = 40$ g/dm³, $\mu_{m1} = 0.4$ day⁻¹, $k_{s1} = 1.9$ g/dm³, $k_I = 6.67$; $\mu_{m2} = 0.25$ day⁻¹, $k_{s2} = 0.37$ g/dm³, $Y_b = 5$; $k_2 = 4.17$, $Y_g = 3.1$. The fundamental sampling time in the simulator of Matlab/Simulink is selected as 0.01 day. And the desired piecewise continuous output trajectory reference Q_r , has the following time profile, dm³/day: 0.29 - up to day 100; 0.80 - day 100 to 150; 0.29 - day 150 to 250.

The model initial conditions are chosen to be $S_0(0) = 10$, $X_1(0) = 0.36$, $S_1(0) = S_2(0) = 0.18$ and $X_2(0) = 15.66$ g/dm³ according to [30]. The parameters of the proposed controller are $Cc = 3$, $\sigma = 0.12$, $G = 1$ and $\epsilon = 0.01$ day – a value chosen to be equal to simulator sampling time.

The additive measurement noise $v(t)$ is considered as zero-mean Gaussian white noise with a sample time of 1 day and with realistically chosen covariance parameter values corresponding to average relative error of 5%. The additive disturbance $w(t)$ of S_{i0} is considered of two components: first, a zero-mean Gaussian white noise with the same sample time and with realistically chosen covariance parameter values corresponding to average relative error of 20%; second, a step-wise parameter disturbance of 20% S_{i0} at simulation day 140 and of -20% S_{i0} at day 170. The two disturbances $v(t)$ and $w(t)$ are shown on Fig. 5 and Fig. 6.

The simulation of the CRMFC performance is shown on Fig. 7, under the above chosen great value relative parameter errors and the step-wise disturbance. It can be concluded that the proposed control demonstrating robust performance in output reference tracking. The corresponding control action D is on Fig. 8 which shows in a zoom interval. The proposed controller has a very simple structure, moreover because of the used time unit is day, this type of controller can be implemented easily with nowadays actuators and embedded computer technologies. Under the proposed controller, the system states are illustrated in Fig. 9 – Fig. 13 which are stable.

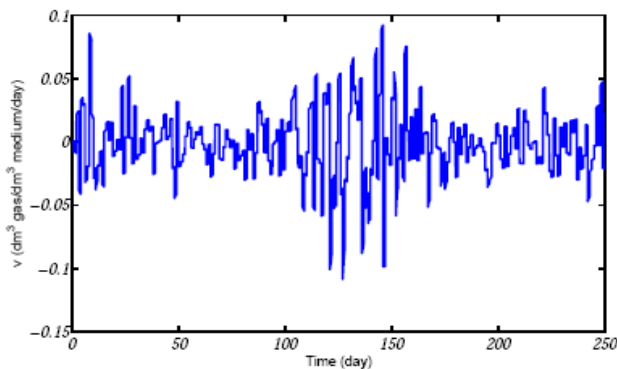


Fig. 5. Realistic disturbances $v(t)$ for the simulation test

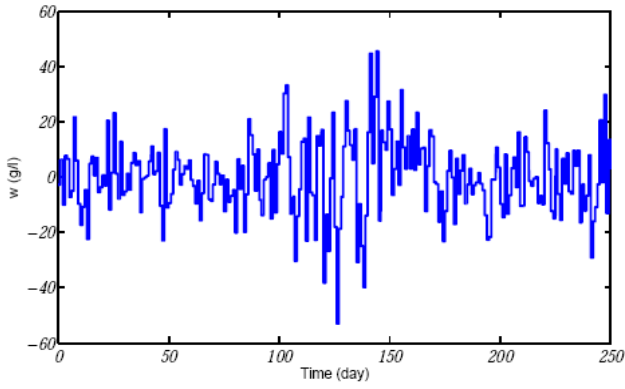


Fig. 6. Realistic disturbances $w(t)$ for the simulation test

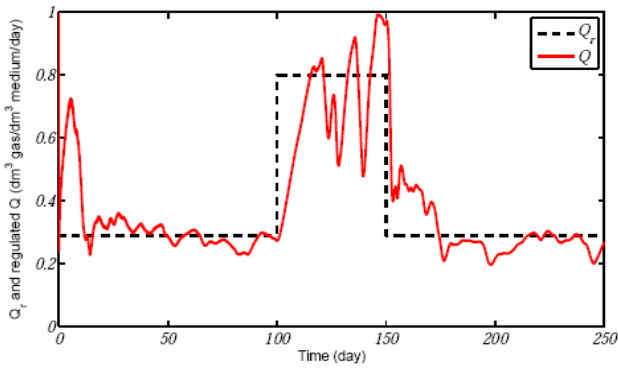


Fig. 7. Output regulation under CRMFC with different noises

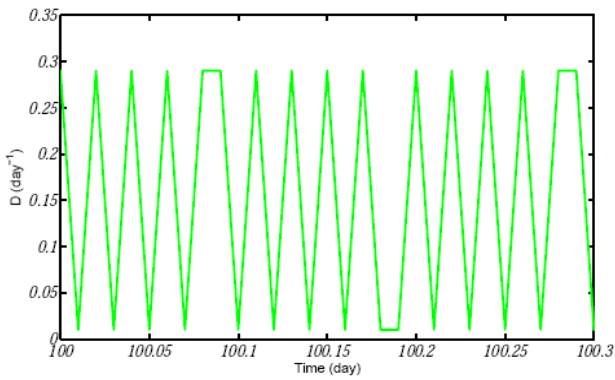


Fig. 8. CRMFC input signal in zoom for clear illustration

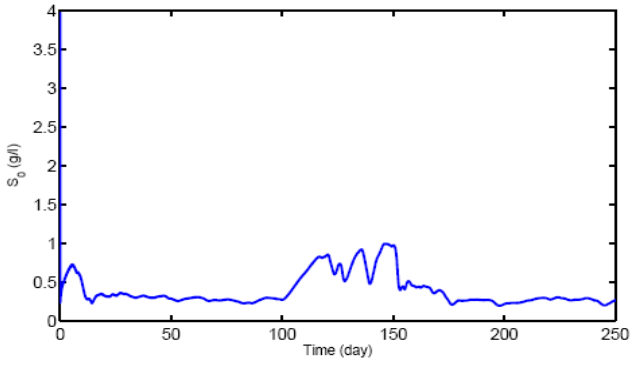


Fig. 9. Substrate S_0 evolution

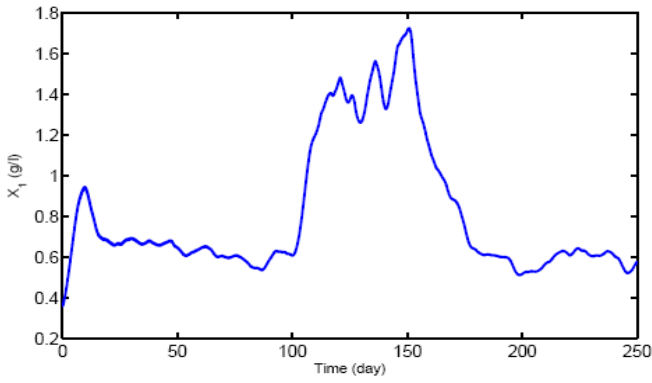


Fig. 10. Acidogenic bacteria concentration X_1 evolution

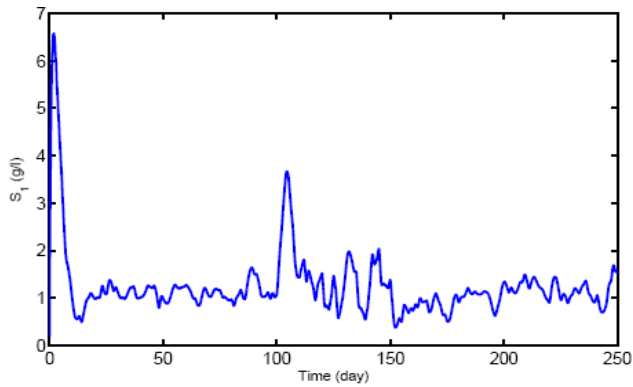


Fig. 11. Substrate S_1 evolution

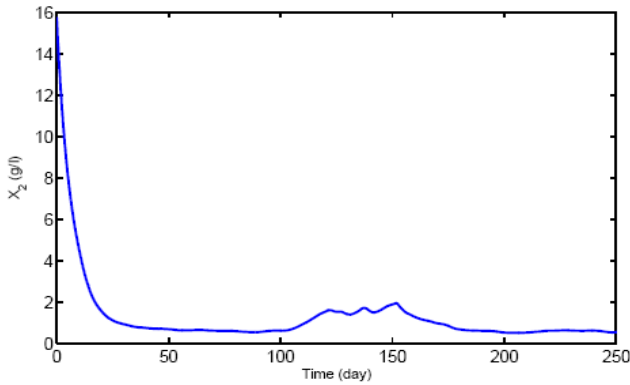


Fig. 12. Methanogenic bacteria concentration X_2 evolution

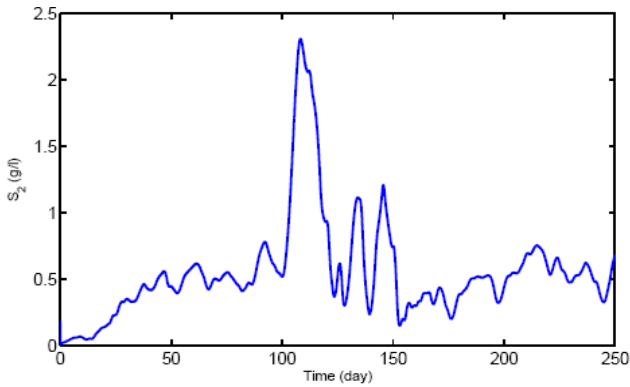


Fig. 13. Substrate S_2 evolution

5 Conclusions

A composite recursive model-free controller with a recursive model free controller based stabilization component and a time delay control based compensation component with a recursive calculation structure has been proposed for an experimental AD process based five-order nonlinear system. The proposed controller which does not require any knowledge of the model parameters is robust due to its adaptive mechanism and exhibits excellent performance in tracking a multi-step-wise reference trajectory under stochastic and step-wise disturbances of the inlet substrate concentration and measurement noise in the methane outflow rate. The proposed controller is planned to be implemented and mastered further on the referred real-time anaerobic digestion process.

Acknowledgements. This work was supported by contract No. DO 02-190/08 of the Bulgarian National Science Fund, the European project INTERREG IVA – SCODECE (Smart Control and Diagnosis for Economic and Clean Engine) and the Picardie region of France.

References

1. Steyer, J.P., Estaben, M., Polit, M.: Fuzzy Control of an Anaerobic Digestion Process for the Treatment of an Industrial Wasterwater. In: Proc. 6th IEEE Inter. Conf., Fuzzy Systems, Barcelona, pp. 1245–1249 (1997)
2. Simeonov, I., Queindec, I.: Linearizing control of the anaerobic digestion with addition of acetate (control of the anaerobic digestion). *Control Eng. Practice* 14, 799–810 (2006)
3. Diop, S., Simeonov, I.: A differential algebraic approach to anaerobic digestion estimation problems. In: Proc. 14th Mediterranean Conf. on Control and Automation, Ancona, pp. 1–5 (2006)
4. Deublein, D., Steinhauser, A.: Biogas from waste and renewable resources. An introduction. Wiley-VCH, Weinheim (2008)
5. Harmon, J., Pullamanappallil, P., Lyberatos, G., Svoronos, S.A.: Stabilization of a Continuous Glucose-Fed Anaerobic Digester via Feed Rate Control. In: Proc. Amer. Contr. Conf., pp. 2156–2160 (1990)
6. Pullammanappallil, P., Svoronos, S.A., Lyberatos, G.: Optimal Response of Anaerobic Digesters to Inhibitors Entering with the Feed. In: Proc. Amer. Contr. Conf., Boston, pp. 1341–1342 (1990)
7. Bastin, G., Dochain, D.: Non linear adaptive control algorithms for fermentation process. In: Proc. Amer. Contr. Conf., Atlanta, pp. 1124–1128 (1988)
8. Dochain, D.: Multivariable Adaptive Control of Nonlinear Completely Mixed Bioreactors. In: Proc. Amer. Contr. Conf., San Diego, pp. 2661–2666 (1990)
9. Petre, E., Selisteanu, D., Sendrescu, D.: Adaptive control strategies for a class of anaerobic depollution bioprocesses. In: Proc. IEEE Int. Conf. Automation, Quality and Testing, Robotics, Cluj-Napoca, pp. 159–164 (2008)
10. Antonelli, R., Harmand, J., Steyer, J.P., Astolfi, A.: Set-point regulation of an anaerobic digestion process with bounded output feedback. *IEEE Trans. Contr. Syst. Technol.* 11, 495–504 (2003)
11. Simeonov, I., Chorukova, E.: Neural networks modelling of two biotechnological processes. In: Proc. Second IEEE Int. Conf. Intelligent Systems, pp. 331–336 (2004)
12. Galvan-Guerra, R., Baruch, I.S.: Anaerobic digestion process identification using recurrent neural network model. In: Proc. Sixth Mexican Int. Conf. Artificial Intelligence, Aguascalientes, pp. 319–329 (2008)
13. Carlos-Hernandez, S., Mallet, G., Beteau, J.F., Sanchez, E.N.: A TS Fuzzy Observer for an anaerobic fluidized Bed. In: Proc. 2006 IEEE Inter. Conf. Fuzzy Systems, Vancouver, pp. 2299–2304 (2006)
14. Garrett, M.T.: Hydraulic control of activated sludge growth rate. *Sewage & Ind. Wasters* 30, 253–261 (1958)
15. Beck, M.B.: Identification, estimation and control of biological wasterwater treatment processes. *IEE Proc.* 133, 254–264 (1986)
16. Loeser, M., Redfern, M.A.: Overview of biomass conversion and generation technologies. In: Proc. 43rd Int. Universities Power Eng. Conf., Padova (2008)
17. Loeser, M., Redfern, M.A.: How small can micro-scale generation be? In: IEEE Electrical Power & Energy Conf. on Size Analysis of a Novel Biomass Power Plant, pp. 1–6 (2009)

18. Curry, N., Pillay, P.: Converting food waste to usable energy in the urban environment through anaerobic digestion. In: Proc. 2009 IEEE Electrical Power & Energy Conf., Montreal, pp. 1–4 (2009)
19. Dochain, D., Vanrolleghem, P.A.V.: Dynamical modeling and estimation in wastewater treatment process. IWA Publ., London (2001)
20. Simeonov, I.: Dynamical modeling and estimation in wastewater treatment process. In: Tzonkov, S. (ed.) ch. 2. Prof. M. Drinov Acad. Publ. House, Sofia (2010)
21. Koncar, V., Vasseur, C.: Control of linear systems using piecewise continuous systems. IEE Control Theory & Appl. 150, 565–576 (2003)
22. Wang, H.P., Vasseur, C., Koncar, V.: Piecewise continuous systems used in trajectory tracking of a vision based x-y robot. In: Novel Algorithms and Techniques Telecommunications, Automation and Industrial Electronics, pp. 255–260. Springer, Netherlands (2008)
23. Wang, H.P., Vasseur, C., Koncar, V., Chamroo, A., Christov, N.: Design and implementation of robust hybrid control of vision based underactuated mechanical nonminimum phase system. Studies in Informatics and Control 19, 35–44 (2010)
24. Wang, H.P.: Observation et commande par systèmes à fonctionnement par morceaux. Editions Universitaires Européennes, Sarrebruck, Allemagne (2011)
25. Youcef-Toumi, K., Fuhlbrigge, T.A.: Application of decentralized time delay controller to robot manipulators. In: Proc. IEEE Int. Conf. on Robotics and Automation, Scottsdale, pp. 1786–1791 (1989)
26. Youcef-Toumi, K., Shortlidge, C.C.: Control of robot manipulators using time delay. In: Proc. IEEE Int. Conf. on Robotics and Automation, Sacramento, pp. 2391–2398 (1991)
27. Cho, G.P., Chang, P.H., Park, S.H.: Robust Trajectory Control of Robot Manipulators Using Time Delay Estimation and Internal Model Concept. In: Proc. of 44th IEEE Conf. Decision and Control, and the European Control Conference (2005)
28. Isidori, A.: Nonlinear Control System, 3rd edn. Springer, London (1995)
29. Wang, H.P., Vasseur, C., Koncar, V., Christov, N.: Recursive model free controller used in MIMO nonlinear system's trajectory tracking. In: Proc. 18th Mediterranean Conf. on Control and Automation, Marrakech, pp. 436–441 (2010)
30. Yordanova, S., Noikova, N., Petrova, R., Tzvetkov, P.: Neuro-Fuzzy Modelling on Experimental Data in Anaerobic Digestion of Organic Waste in Waters. In: Proc. IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sofia, pp. 84–88 (2005)
31. Ahring, B.K. (ed.): Biomethanation. Springer, Heidelberg (2003)
32. Angelidaki, L., Ellegaard, L.E., Ahring, B.K.: Biotech. Bioeng 42, 159–166 (1993)
33. Batstone, D.J., Keller, J., Angelidaki, I., Kalyuzhnyi, S.V., Pavlostathis, S.G., Rozzi, A., Sanders, W.T., Siegrist, M.H., Vavilin, V.A.: Anaerobic digestion model no. 1 (ADM1). IWA Task Group for Mathematical Modelling of Anaerobic Digestion Processes. IWA, London (2002)
34. Lyberatos, G., Skiadas, I.V.: Global Nest. Int. J. 1, 63–76 (1999)
35. Simeonov, I., Momchev, V., Grancharov, D.: Water Research 30, 1087–1094 (1996)
36. Simeonov, I., Karakashev, D.: Mathematical modelling of the anaerobic digestion including the syntrophic acetate oxidation. In: Preprints 7th Vienna Internat. Conf. on Math. Modelling (MATHMOD), on CD (2012)

Grid Based Hydrologic Model Calibration and Execution

Danut Mihon¹, Victor Bacu¹, Denisa Rodila¹, Teodor Stefanut¹,
Karim Abbaspour², Elham Rouholahnejad², and Dorian Gorgan¹

¹ Technical University of Cluj-Napoca, Computer Science Department,
G.Baritiu str. 26-28, 400027 Cluj-Napoca, Romania
{vasile.mihon,teodor.stefanut,victor.bacu,denisa.rodila,
dorian.gorgan}@cs.utlcluj.ro

² EAWAG, Swiss Federal Institute for Aquatic Science and Technology, Switzerland
{karim.abbaspour,elham.rouholahnejad}@eawag.ch

Abstract. The continuous expansion of distributed hydrological models applied on different geographical regions in order to solve and predict water resource problems raised multiple issues related to the model calibration and execution processes. The calibration process was performed on SWAT (Soil and Water Assessment Tool) hydrological model that could be used to predict the impact of land management practices on water, sediment and agricultural chemical yields in complex watersheds. This paper presents methods, algorithms, data access issues and human-computer interaction techniques used in developing a Web application for the Grid based SWAT model execution and calibration, called gSWAT. The SWAT model calibration process is time consuming (e.g. in some situations its execution could reach hours or even days in length). The Grid is the platform that integrates the gSWAT application, due to its parallel and distributed characteristics, offering high computation and storage capabilities in response to the calibration process requirements.

Keywords: Grid infrastructure, SWAT model, calibration, performance gain.

1 Introduction

The prediction of natural phenomena is based on complex models that process large volumes of data in order to obtain accurate results. In most cases a single simulation of such a process is not enough, and the prediction has to be repeated several times. Better predictions require a large number of simulations, which increases the computation power needed to process all this data.

This paper highlights the gSWAT application that allows the calibration and execution of the SWAT hydrological model over the Black Sea Catchment. For such a large watershed (over 2 million square kilometers) the processes involved in the SWAT model calibration had to be performed over a powerful, distributed and scalable infrastructure, the Grid infrastructure in this case.

There is a different meaning between the concepts of SWAT calibration and execution. The first one is used to estimate the model parameters values in order to represent as accurate as possible the geographical watershed that this model is referring to. In order to increase this accuracy, multiple simulations of the same model are used, until a specific objective function is satisfied. The SWAT execution applies only on calibrated models and represents the basis in generating prediction scenarios. The calibration phase represents the most intensive computation process, and that is the main reason why this paper strongly focuses on its description and implementation within the gSWAT application.

In order to produce better predictions regarding water resources problems, the SWAT model has to be calibrated first. The calibration process tries to best estimate model parameters values related to measured values for the same set of parameters. In other words starting from a given numerical interval, new random values must be generated, in such way that they are as close as possible to the measured values [2].

The entire hydrological model could be executed on different Grid nodes. The calibration process is time consuming, in some cases it take 500 or even more executions of the same model, called iterations. So, what the Grid infrastructure offers, is the possibility to run these 500 iterations in parallel (e.g.: use 5 Grid nodes for the entire calibration process, where each node handles 100 iterations). At the end, when all nodes finished their execution, a specialized tool collects the results and regroups them as a single resource. The process of monitoring and execution such a large number of tasks requires a high performance scheduling algorithm [1] that allows parallel and distributed executions over the Grid infrastructure.

SWAT is a hydrological model that operates on a daily time step and it is used for predicting the water resources, sediment, and chemical yields in a specific watershed. It could be applied on small, medium or large watersheds. For example it is currently used to perform macro-scale assessments for the Mississippi River basin [3].

The gSWAT application brings in new features related to SWAT calibration and execution processes. First of all, the usage of Grid infrastructure as a platform for distributed and parallel executions significantly reduces the total execution time. Implementing the gSWAT as a Web application [4] is another important step, because it allows the users an easy access to the application functionality regardless of their location. The new human-computer interaction techniques of the gSWAT application, guides the user through the actions that he performs and offers an advanced recover from error support mechanism.

The following sections describe these aspects in a detailed manner, including both technical and theoretical solutions. One important issue regards the graphical user interface. That is why the most important human-computer interaction techniques for the gSWAT application are also described in this paper.

2 Related Works

There are several models that have similar functionality with SWAT. The most important ones are HSPF (Hydrological Simulation Program - FORTRAN) and SHETRAN (Système Hydrologique Européen). The first one could be described

as a software application that can simulate the hydrological and associated water quality processes on pervious and impervious land surfaces [5].

The second one [6] is used for modeling coupled surface and subsurface water flow in river basins. The processing results could be visualized using animated graphical computer displays. SHETRAN could be easily used in the analysis of environmental impacts: land erosion, pollution, climate change or in surface-water and ground-water resources studies.

One of the main goals of the gSWAT application is related to the assessment of the sustainability and vulnerability in the Black Sea Catchment. Calibrating the SWAT model becomes an important problem when applying the model on such large watersheds. In these cases the calibration time could take days to complete, even on the finest standalone machines. There are a few solutions that could be used to overcome this problem.

The first one is to execute the calibration process on a single-core machine. Important progress was already established in this direction through the SWAT-CUP (SWAT Calibration and Uncertainty Procedures) application. SWAT-CUP is a software tool designed to calibrate the SWAT model, based on some input parameters. This tool also provides complex capabilities regarding the analysis of the calibrated outputs (e.g.: organize data in charts). In 2008 a second version was released [7]. This solution is useful when the SWAT model is used for small watersheds.

The second solution extends the SWAT-CUP application to work on multi-core machines. This significantly reduces the total execution time for the calibration process, but still could not be applied on large watersheds (e.g. Black Sea Catchment), only on small and medium ones.

The last two solutions implies the usage of large distributed infrastructures such as Grid and Cloud computing. Both platforms offer high computation and storage capabilities. The main difference between them is that the Grid could be used free of charge as long as the user has valid Grid certificates, emitted by a competent CA (Certificate Authority). Meantime, using the Cloud infrastructure users have to pay for each time they use its services. With proper configuration [8], the Grid infrastructure could provide significant speed up, regarding the calibration and execution of the SWAT hydrological model. The Experimental results section details all these aspects.

Even though the SWAT-CUP application is used for several years to perform the SWAT model calibration process, the gSWAT tool brings in new features, such as: significantly reduces the total calibration time, offers high storage and computing capabilities due to the Grid infrastructure, supports multi-calibration processes, it is built as a Web application, offers interactive support for result management, etc. Calibration and execution results of the SWAT model could be analyzed and visualized over interactive maps, using different image processing algorithms [9] in order to correctly classify the land management practices on water, sediment or agricultural chemical yields. All of these features will be described in the following sections.

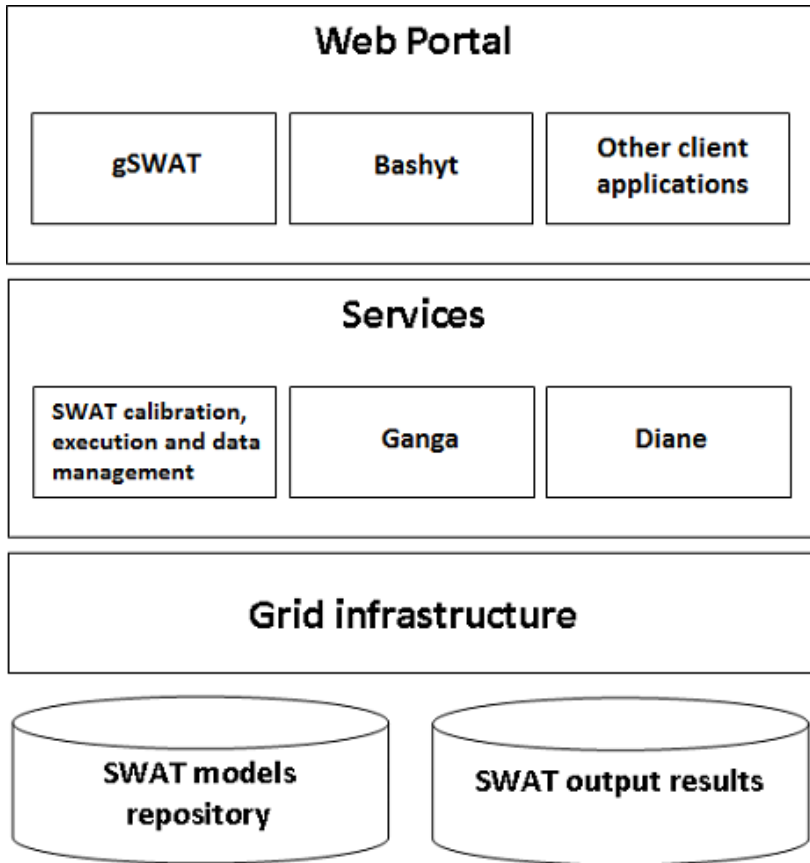


Fig. 1. gSWAT system related architecture

3 gSWAT System Related Architecture

This section describes the general system related architecture (Fig. 1) and the phases involved in the calibration process of the SWAT model over the Grid infrastructure.

The Web Portal represents the top architectural level and consists of several environmental applications: gSWAT, Bashyt [10], GreenLand [11], eGLE [12], etc. All these applications enrolled within the portal share the same users' credentials, by using the single sign on (SSO) mechanism. This means that the user is able to access the functionality of all these applications with a single generic account. Besides this fact, the applications also offer the possibility of independent user authentication, regardless of the previous mentioned one.

The client side (or the graphical interface) is built as a Web application using the Adobe Flex latest technology. The analysis and the visualization of the SWAT model based scenarios are possible through the Bashyt platform. A scenario represents the output result of the SWAT model execution based on an already calibrated model. Different scenarios could be implemented, for simulating or predicting hydrological phenomena (e.g. water-use sustainability, floods, improving irrigation systems, etc).

The Bashyt platform allows the user to visualize the SWAT model execution output (called scenario) as an animated 2D map that displays all the watersheds stations along with some essential characteristics. Scaling and translating the map using mouse events is also possible within the framework of this platform.

The input data, the external dependencies, the execution workflow structure are defined in this stage, based on the user requests, and sent to the Grid infrastructure through GANGA [13] and Diane [14] tools. The Diane application analyzes the calibration process complexity and allocates the number of Grid workers that are going to be used in the execution phase. After that, all data are sent to these nodes, and executed in a parallel manner.

The monitoring component implemented within the GANGA tool offers important feedback about the calibration processes running on the Grid infrastructure. This feedback could then be displayed to the user in the gSWAT graphical interface in terms of: total execution time described as an animated progress bar, Grid process status (Schedule, Submitted, Waiting, Running, Completed), execution details in the form of a system log (contains all internal processing message and error), etc.

The Grid infrastructure offers high storage and parallel computation capabilities [15] in order to successfully calibrate and execute the SWAT hydrological model for different geographical watersheds. Because the output of these processes could reach a few Gb in size, a standalone machine becomes inefficient. The SE (Storage Element) nodes inside the Grid offer storage support for large volumes of data. Accessing these resources requires valid Grid certificates, emitted by a competent CA (Certificate Authority). The data transfer between SE nodes and Grid applications is based on the GSIFTP (Grid Security FTP) protocol. The SRM (Storage Resource Manager) manages all SE nodes inside the Grid infrastructure, and offers support for error and trial recovery mechanisms, minimizes data replication, allocates a specific SE that has enough memory space to support the entire Grid process output, etc. The calibration time could also be reduced by using efficient scheduling algorithms [16].

After the model is calibrated the user could create scenarios based on the outputs of the calibrated SWAT model. The scenarios could be created through the gSWAT execution component. When this process ends, the results are sent to BASHYT platform. This is useful for specialists to create their own scenarios (e.g.: predicting the water-use sustainability) that could analyze and visualize them over an interactive map that displays in an animated manner the prediction results.

4 Data Model and Algorithms

This section points out the problems that arise when storing and accessing data used in the SWAT calibration and execution processes. Also some practical solutions are identified related to these problems, solutions that are included into the gSWAT development steps.

4.1 Data Management

When the SWAT model is applied on large watersheds its size reaches hundreds of mega bytes or even gigabytes of information. Storing these inputs on a single computer machine is impossible. Because gSWAT application is based on the Grid infrastructure, the Grid storage capabilities could be used to solve this problem. There are multiple SE (Storage Element) nodes inside the Grid infrastructure where these data could be placed.

Storing the data into SE nodes [17] is possible only for authorized users that identify themselves by using Grid certificates. These certificates are created by a CA (Certificate Authority) organization on user request. In other words, the users could not access data inside the Grid infrastructure without these valid certificates.

The Web portal supports multiple user categories. Some of them have the possibility to execute and analyze data, and others have only the possibility to visualize data in a user friendly manner. The first type of users must be Grid certificate possessors, because they need to access and manage data according to their needs. The other ones do not have a direct access to data, but they do visualize and download these information. So, another problem arises: how to access Grid data from outside the Grid infrastructure? To solve this kind of problem new Web and Grid services must be implemented in such a way that the user is not aware of the backhand mechanism of data access.

4.2 SWAT Model Structure

The input for the gSWAT application is represented by the TxtInOut folder that contain a large number of files (tens of thousands) that store daily information about vegetation, weather, soil and land management practices occurring in the watershed.

The entire basin where the SWAT model applies could be divided into sub-basins, and each sub-basin is composed of several HRU (Hydrological Response Units). The TxtInOut folder contains information about all the sub-basins, organized in text files.

For each sub-basin the user could specify the operations that are going to be performed (e.g.: add a certain quantity of pesticides, enlarge the percentage of nutrients for the third sub-basin, etc). All these operation are stored into the management file (.mgt extension) that is unique for each HRU.

When calibrating the SWAT model, the user can choose the algorithm that will be used in this process. Currently the gSWAT application is based on a single calibration algorithm, called SUFI2 (Sequential Uncertainty Fitting).

The gSWAT application allows the user to edit the TxtInOut files through the editor component. The files have a default structure that could be modified by the user according to his needs. The output of the gSWAT application provides useful information about the calibration of the SWAT model. This information could be visualized as text (Best_Slim.Sf2, Best_Par.Sf2, etc) or as interactive charts (95ppu.sf2 file).

4.3 Calibration Algorithms

Generally speaking, the calibration process represents the act of adjusting the accuracy of the measured parameter through different measurement techniques [18]. The manual adjustment is a subjective process of comparing the observed values with the ones obtained during a simulation of the calibration process. This scenario is time consuming, because of the large number of hydrological parameters that need to be adjusted. Instead, the development of a semi-automated process is taken into account as one of the gSWAT main features.

Depending on the algorithms used in this process, the calibration results will be different. So, choosing the correct algorithm is a difficult task, and in most cases performed by a domain field specialist. The following paragraphs describe the most important methods used in calibrating the SWAT hydrological model.

SUFI2 (Sequential Uncertainty Fitting) can perform uncertainty analysis and calibrate the model based on a set of parameters. Also, it significantly reduces the process duration related to other algorithms. The p-factor represents the percentage of measured data by the 95% prediction uncertainty; its values are being stored in 95ppu.sf2 output file. The p-factor is obtained through Latin hypercube sampling. The R-factor could also be used to measure the strength of the uncertainty analysis in the calibration process.

Basically, the calibration process repeats until an objective function is satisfied. The SUFI2 algorithm handles multiple objective functions such as: root means square error, Chi square, Nash-Sutcliffe, R^2 and bR^2 . Choosing the right one [19] is closely related to the domain field specialist experience.

For example the bR^2 function allows accounting for the discrepancy in the magnitude of two signals as well as their dynamics. R is computed based on a Residual sum of squares between measured and observed parameters, b is the slope of the regression line between measured and randomly generated values and ϕ is the objective function (1).

$$\phi = \begin{cases} |b| R^2, & |b| \leq 1 \\ |b|^{-1} R^2, & |b| > 1 \end{cases} \quad (1)$$

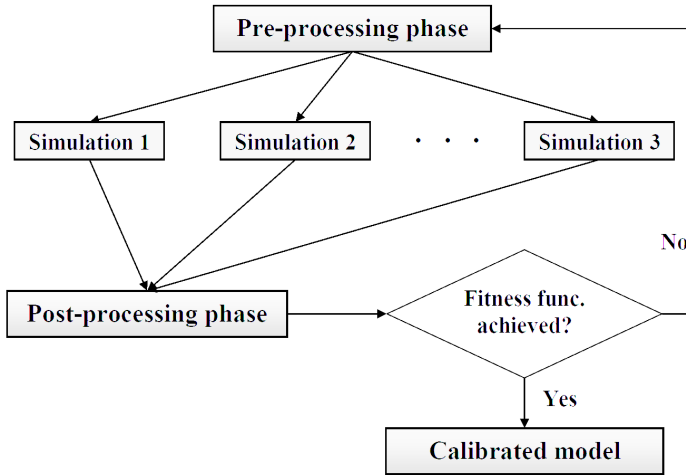


Fig. 2. SWAT calibration process overview

GLUE (Generalized Likelihood Uncertainty Estimation) allows the possibility to have non-uniqueness parameters values during the calibration process. When applying the SWAT model on large watersheds there is the possibility to not be able to generate unique random values in all situations. This is the main assumption used in GLUE algorithm. In this case the objective function is easier to compute by using a technique based on the estimation of the weights or probabilities associated with different parameter sets [20].

ParaSol (Parameter Solution) is another algorithm that allows SWAT model parameters uncertainty analysis. Like all the other methods, this algorithm must satisfy a general objective function in order to complete the calibration process. In this case the objective function is computed based on model outputs and observation time series. Then it aggregates this objective function using a global optimization criterion (GOC). At every simulation rerun, the computed value is stored and then a minimization of this function is performed using the Shuffle Complex (SCE-UA) method [21].

These are the most known calibration methods that could be used for the SWAT model. The idea that a regular user, with no insights about these algorithms, could prepare the SWAT model parameters for calibration has no realistic output. This is the main reason why the gSWAT strongly encourage the calibration process to be performed only by the specialized user category. Meanwhile, the other types of users may visualize and download the results of the prediction process.

4.4 *SWAT Model Calibration Phases*

Each calibration process consists of multiple iterations, executed until a calibration criterion is satisfied (Fig. 2). A single iteration consists in multiple simulations, in some cases this number increases to a few hundreds. At the iteration level there are three main phases of execution: pre-processing, actual execution of data and the post-processing phase. The following paragraphs shortly describe all these internal processes.

4.4.1 The Pre-processing Phase

It is worth to mention that a numerical value interval is attached to each SWAT input parameter. Taking this into account, one iteration takes a discrete random value for each parameter, from the corresponding interval, and performs the calibration algorithm. For better accuracy, the iteration process should include multiple simulations, in order to cover all the values inside these intervals. Generally speaking this number is limited to 500 simulations per each calibration process.

4.4.2 The Execution Phase

This represents the most time consuming step of the entire SWAT model calibration process. That is the may reason why it requires the parallel execution support of the Grid infrastructure. In other words, based on the process complexity a different number of Grid workers are allocated for this process by using the Diane master node.

All the data are initially transferred from the application server to the Grid nodes, using the GANGA scripts. These scripts represent the linkage between the gLite middleware and the gSWAT application. After all the project dependencies are successfully generated at the client side level, they are transferred to the Grid workers in order to be processed. Depending on the number of simulations chosen by the user a different number of worker nodes will be involved in this execution process.

4.4.3 The Post-processing Phase

Just like the pre-processing phase, it takes place on the gSWAT application server after all the Grid workers finish their execution. A collection mechanism generates the final calibration output, based on all the intermediate results given by these Grid workers. An interesting aspect is that initially all these results are stored inside SE nodes. When the user makes a request for this result it is archived and sent to the application server and from here to the graphical user interface.

The gSWAT platform offers a significant speed up regarding the total SWAT model calibration time, by using the Grid infrastructure. Another important aspect of this platform is represented by the analysis and visualization features of the output calibration process (Fig. 3).

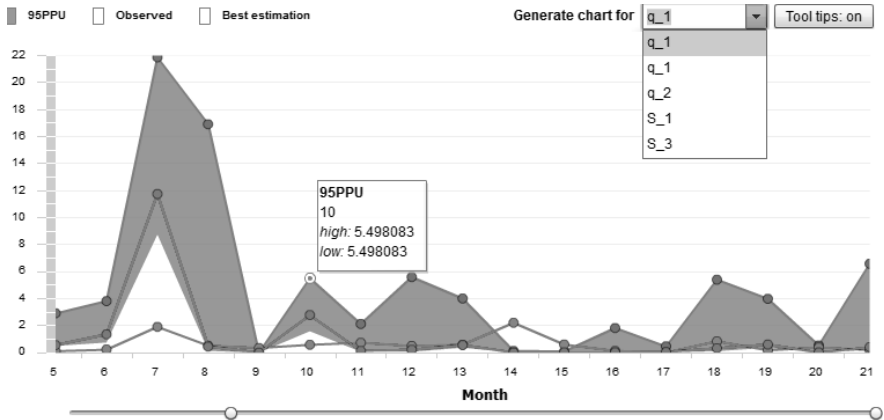


Fig. 3. SWAT calibration output analysis and visualization

5 gSWAT Graphical User Interface

The gSWAT is a Web based application that allows users the possibility to perform the SWAT model calibration through a simple but powerful graphical user interface. It is a client-server application that uses the latest Adobe Flex technology on the client side and java based Web services.

Because it handles multiple user categories, each gSWAT operation must match the user needs, but in the same time they have to be easy to put into practice. When calibrating the SWAT model by using the SUFI2 algorithm, the gSWAT application generates a specific structure for the input and output files. These files could be easily modified by the user through the editor component that allows independent management of each file.

The output results of the SWAT model calibration processes consists of several data files that contain values of the calibrated parameters. One of these files is extremely important because it could be displayed to the user as an interactive chart (Fig. 3). As can be seen parameters values are displayed to the user in all the chart's inflexion points.

The chart from Fig. 3 highlights the SWAT calibration result (Best estimation item) and the observed values for the same set of model parameters (Observed item). Also an uncertainty analysis could be performed based on this kind of data representation.

The x axis of the chart represents the monthly time period used in the calibration process, whereas the y axis contains the value representation for the same parameters set. For better analysis the time period could be compressed or expanded by using the horizontal slider attached to this chart.

6 Experimental Results

Two kinds of experiments were conducted in order to prove the Grid infrastructure efficiency over single and multi-core standalone machines. It is worth to mention that the second experiment highlights the Grid scalable nature regarding the number of users that perform simultaneous SWAT calibration processes. The RO-09-UTCN site, within the envirogrids.vo.eu-egee.org VO (Virtual Organization) was used for both experiments. It contains 1024 CPUs and 13TB of storage space.

The first experiment was performed based on a medium scale SWAT hydrological model (around 40000 input files) that represents the Danube watershed. The objective of these measurements is to highlight the Grid processing power over standalone machines. The hardware resources for this experiment were represented by a desktop computer with 2xQuad core processor at 2.13Ghz and 4 GB of RAM, along with the Grid infrastructure. The number of simulations started from 1 up to 100. It is worth saying that each calibration process is executed on multiple Grid worker nodes. The Grid process submission using GANGA and Diane tools do not allow the possibility to specify what kind of hardware resources to use on execution phase. This way a random group of available Grid workers are selected, with different hardware capabilities.

In the case of Grid execution, the simulations were divided into multiple groups (each such group consists of 10 simulations). The Grid execution was performed in parallel at group level, and sequentially for all the iterations inside each group.

After all the inputs were specified, the experiment began. An average of seven minutes execution time per simulation was obtained on the standalone machine. The Grid execution performs faster on large number of simulations, and slower in other cases (Fig. 4).

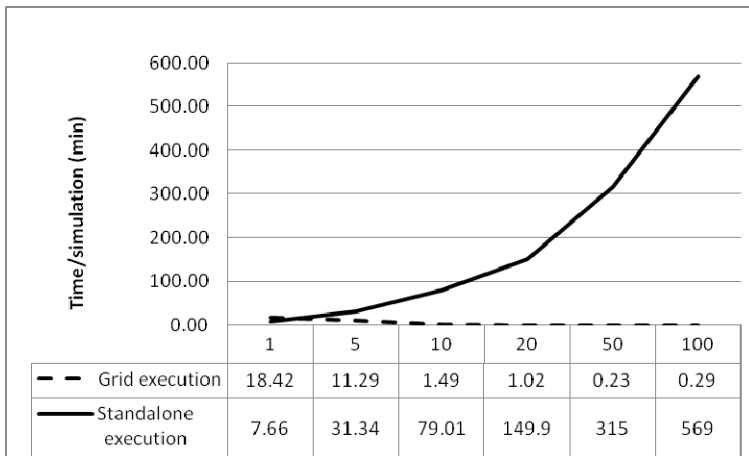


Fig. 4. Grid efficiency over standalone execution

In the conducted experiment the critical inflexion point for the Grid execution is around 2.5 simulations. The Grid calibration process consists of several phases, highlighted in Table 1, and the total execution time is dependent on all of these intermediate steps. Using less the 2.5 simulation decreases the Grid performance; because the time needed to perform the steps 1, 2, 3, and 5 significantly interfere in the total Grid calibration time.

It is well known that the Grid becomes efficient for large volume of data processing. Taking this into account, the execution gain between Grid nodes and standalone machines is representative for larger number of simulations.

The second experiment highlights the scalable nature of the Grid infrastructure. Several measurements were conducted, with different number of users, starting from 1 up to 7. In order to increase the experiment's complexity, the simulation number was varying for each user. Constants values of 50, 250, and 500 simulations were used in the SWAT calibration process.

Table 1. Grid execution phases involved in a SWAT calibration process

No.	Phase	Description
1	Process decomposition	The entire SWAT calibration process is decomposed into multiple atomic tasks that are going to be executed as different processes over Grid worker nodes.
2	Workers allocation	Depending on the calibration process complexity a specific number of Grid nodes will be allocated for this execution
3	Task submission	The action of sending all the inputs and the external dependencies to that worker. Because the SWAT hydrological model is already stored inside Grid, on the SE nodes, this time is less relevant than in the case of transferring data from the application server
4	Execution	The actual execution process, as described in the previous sections
5	Output generation	Generates the final calibration result, based on all intermediate outputs generated by the Grid parallel executions

Fig. 5 introduces some small fluctuations regarding the total calibration time for each simulation. Mainly, this difference occurs due to the fact that it is almost impossible to maintain the same execution context for each experiment: network traffic, access to the same memory space, Grid workers load, etc.

The main objective of this experiment is to highlight the fact that the average Grid execution time for a simulation remains constant, regardless of the Grid usage intensity degree.

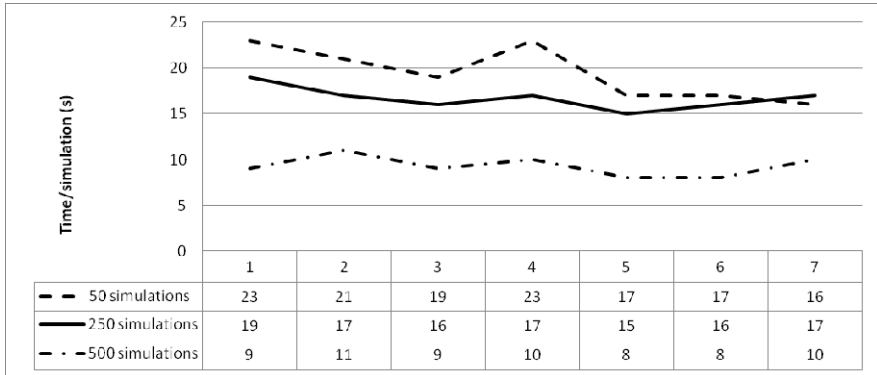


Fig. 5. Scalability of the Grid infrastructure

7 Conclusions

This paper describes the calibration and execution processes of the SWAT hydrological model, based on the gSWAT platform. The Grid infrastructure is one of the distributed platforms that provide a significant performance gain when calibrating the SWAT model. The communication mechanisms between the architectural modules of the gSWAT platform were also highlighted throughout this paper. The experimental results validate the methods and algorithms used in the calibration process, by pointing out the Grid infrastructure efficiency, flexibility and scalability.

Acknowledgment. This research is supported by the enviroGRIDS Project funded by the European Commission, through the Contract 226740.

References

1. Aysan, H., Dobrin, R., Punnekkat, S.: Probabilistic Scheduling Guarantees under Error Bursts in Controller Area Network (CAN). In: 18th International Conference on Control Systems and Computer Science, pp. 853–859. Politehnica Press, Bucharest (2011)
2. Zhang, X., Srinivasan, R., van Liew, M.: Multi-Site Calibration of the SWAT Model for Hydrological Modeling. Transactions of the American Society of Agricultural and Biological Engineers 51, 2039–2049 (2008)

3. Gassman, P.W., Reyes, M.R., Green, C.H., Arnold, J.G.: The Soil and Water Assessment Tool: Historical Development, Applications, and Future Research Directions. *Transactions of the American Society of Agricultural and Biological Engineers* 50, 1211–1240 (2007)
4. Badea, D., Duca, A.: T100-A Content Management System for PHP Web Applications Development. In: 18th International Conference on Control Systems and Computer Science, pp. 767–771. Politehnica Press, Bucharest (2011)
5. Bicknell, B.R., Imhoff, J.C., Johanson, A.S.: Hydrological Simulation Program – FORTRAN (HSPF), User’s Manual For Release, EPA, U.S. Environmental Protection Agency (2001)
6. Ewen, J., Parkin, G., O’Connell, E.: SHETRAN: Distributed Basin Flow and Transport Modeling System. *Journal of Hydrologic Engineering* 5, 250–258 (2000)
7. Abbaspour, K.C.: SWAT-CUP2: SWAT Calibration and Uncertainty Programs – A User Manual, Department of Systems Analysis, Integrated Assessment and Modeling, Eawag. Swiss Federal Institute of Aquatic Science and Technology, Duebendorf (2008)
8. Ditu, B., Tapus, N.: Improvements of Instruction Scheduling. In: 18th International Conference on Control Systems and Computer Science, pp. 545–552. Politehnica Press, Bucharest (2011)
9. Popa, M., Lupea, D., Lazea, G.: Complex Image Pre-processing Algorithms based on Integral Image Applied to Object Recognition Problems. In: 18th International Conference on Control Systems and Computer Science, pp. 598–603. Politehnica Press, Bucharest (2011)
10. Cau, P., Manca, S.: The Bashyt DSS: a Web Based Decision Support System for Water Recourses Management. In: 4th International SWAT Conference (2007)
11. Mihon, D., Băcu, V., Ștefănuț, T., Gorgan, D.: Considerations on the Grid Oriented Environmental Application Development Framework. In: IEEE 6th International Conference on Intelligent Computer Communication and Processing, pp. 419–426 (2010)
12. Ștefănuț, T., Băcu, V., Gorgan, D.: e-Learning Lesson Development and Execution Based on gProcess Workflow Description Platform and eGLE E-learning Platform. In: HiPerGRID - 3rd International Workshop on High Performance Grid Middleware, pp. 431–436 (2009)
13. Moscicki, J.T., Brochu, F., Ebke, J., Egede, U., Elmsheuser, J., Harrison, K., Jones, R.W.L., Lee, H.C., Liko, D., Maier, A., Muraru, A., Patrick, G.N., Pajchel, K., Reece, W., Samset, B.H., Slater, M.W., Soroko, A., Tan, C.L., van der Ster, D.C., Williams, M.: Ganga: A tool for computational-task management and easy access to Grid resources. *Computer Physics Communications* 180, 2303–2310 (2009)
14. Moscicki, J.T.: DIANE – Distributed Analysis Environment for GRID-enabled Simulation and Analysis of Physics Data. In: Nuclear Science Symposium, vol. 3, pp. 1617–1625 (2004)
15. Stanciu, A., Enciu, B., Neagu, G.: Auto-administrating and self-repairing systems by enforcing a configuration policy. In: 18th International Conference on Control Systems and Computer Science, pp. 827–832. Politehnica Press, Bucharest (2011)
16. Pop, F., Baron, O.A., Cristea, V.: Rescheduling Service for Reliable Distributes Systems. In: 18th International Conference on Control Systems and Computer Science, pp. 553–560. Politehnica Press, Bucharest (2011)

17. Ciobanu, V., Petrescu, A.: Safe Storage of ECG Signal Waveform in Cloud. In: 18th International Conference on Control Systems and Computer Science, pp. 576–581. Politehnica Press, Bucharest (2011)
18. Liu, Y.B., Batelaan, O., De Smedt, F.: Automated calibration applied to a GIS-based flood simulation model using PEST, Department of Hydrology and Earth System Science, pp. 2137–2145 (2009)
19. Poke, M., Slusanschi, E., Podareanu, D., Herisanu, A.: Parallel Numerical Modeling in Airfoil Flow Simulations. In: 18th International Conference on Control Systems and Computer Science, pp. 561–568. Politehnica Press, Bucharest (2011)
20. Franks, S., Beven, K., Chappell, N., Gineste, P.: The Utility of Multi-Objective Conditioning of a Distributed Hydrological Model Using Uncertain Estimates of Saturated Areas. *Hydrology and Earth System Sciences* 3, 477–489 (1999)
21. Zakayo, S.: Multi-Objective Calibration of SWAT Model for Pesticide Simulation, UNESCO-IHE, Institute for Water Education, Master of Science Thesis (2009)

Composite Application for Water Resource Management

Mariana Mocanu¹, Marian Muste², Vasile Lungu¹, and Radu Drobot³

¹ University POLITEHNICA from Bucharest, Romania, Splaiul Independenței. 313, Bucharest, Romania

{mariana.mocanu,vasile.lungu}@cs.pub.ro

² University of Iowa, Iowa, U.S.A

marian-muste@uiowa.edu

³ Technical University of Civil Engineering, Bucharest, Romania

drobot@utcb.ro

Abstract. The paper presents the necessity of using modern information technology in water resource management and the current situation of watershed science and management. The advantages of using composite applications, related to the importance of the water resource management systems are outlined. The architecture of Hydro NETWORK portal, developed by the team, and the database and reporting services are further presented. The paper presents the integration in the portal of two other applications for analysis of mutual influence of surface water and groundwater in river basins.

Keywords: Hydroinformatics, water management systems, composite applications.

1 Introduction

It is increasingly recognized that water as an absolutely vital resource, and at times as a major hazard, poses critical challenges for people in the 21st century. Uses of water include agricultural, industrial, household, recreational and environmental activities, and require fresh water. Awareness of the global importance of preserving water for ecosystem services has only recently emerged as, during the 20th century, more than half the world's wetlands have been lost along with their valuable environmental services. There are well-known, substantial concerns that peoples' uses of water may significantly alter water-quantity and -quality processes in water-cycle systems at the local and regional scales. Addressing these concerns requires a substantially better understanding of the linkages and feedback between the various systems than presently is available. The traditional pillars of the natural systems scientific studies are observation (plus experiment), theory, and analysis (plus computation). Modern information and communication technology capabilities now allow us to address a new class of problems around

the organization of data and information leading to knowledge extraction. The digital revolution is changing radically the way we conduct our science, as well as affecting all facets of society, and it can be argued that informatics became the fourth pillar of the scientific method. Informatics is the science and engineering that occupies the gap between information and communication technology systems and cyberinfrastructure, and use of digital data, information, and related services for research and knowledge generation [5]. The concept is, however, not new and it has predecessors in the water sciences.

At the beginning of the 90's, the rapid process of electronic encapsulation of information and knowledge in hydroscience led to the European concept of hydroinformatics (<http://ncl.ac.uk/hydroinformatics>). Independent from these developments, the U.S. National Science Foundation began in 2002 to reorganize the manner in which it supports computational infrastructure in science and engineering and introduced for this purpose the general concept of cyberinfrastructure (CI). CI is defined as "... grids of computational centres, some with computing power second to none; comprehensive libraries of digital objects including programs and literature; multidisciplinary, well-curated federated collections of scientific data; thousands of online instruments and vast sensor arrays; convenient software toolkits for resource discovery, modelling, and interactive visualization; and the ability to collaborate with physically distributed teams of people using all of these capabilities" (cise.nsf.gov/sci/reports/atkins.pdf).

Recent advances in high-performance computing, communication technologies, GIS-based tools along with innovative statistical, data-driven models, and knowledge discovery models, can characterize the physical-biochemical habitat with increased spatial resolution over large-scale areas [3]. Collectively, these advancements lead to an "information-centric" approach for watershed investigation and management that capitalizes on observations and their interpretation [2].

2 Current Developments

2.1 *Hydroinformatics*

Integrated water resources management (IWRM) is a systematic process for the sustainable development, allocation and monitoring of water resource use in the context of social, economic and environmental objectives (EU Directive 60/2000). It contrasts with the sectorized approach that applies in many countries. When responsibility for drinking water rests with one agency, for example for irrigation water or for the environment, lack of cross-sector linkages leads to uncoordinated water resource development and management, resulting in conflict, waste and unsustainable systems. Integrated management means that all the different uses of water resources are considered together. Water allocations and management decisions consider the effects of each use on the others. They are able to take account of overall social and economic goals, including the achievement of

sustainable development. This also means ensuring coherent policy making related to all sectors. The basic IWRM concept has been extended to incorporate participatory decision-making so that different user groups (communities, environmentalists) can influence strategies for water resource development and management. Management is used in its broadest sense. It emphasizes that the focus is not only on development of water resources but also to the consciously managing of water development in a way that ensures long term sustainable use for future generations.

Common features of the watershed science and management information systems include [5]:

- implemented at the natural landscape unit scale (e.g., basin or watershed)
- consideration of all water cycle fluxes: vertical (precipitation, surface water, evapotranspiration, groundwater) and horizontal (runoff-stream) and their interactions with ecological and socio-economical aspects
- extensive datasets obtained through monitoring, modelling, and post-processing
- multilevel (scientist, managers, education, training, outreach, general public), multi-task (quantitative, qualitative) analysis and visualization
- near-real time operation of the investigative/management platforms (monitoring, interfacing with simulators, process prediction)
- tools for knowledge discovery (mining, data-driven modelling), dissemination, collaboration/participation
- feed-back loops for observatory/monitoring operations, quality control, and application of adaptive strategies (decision-making process, systems control)

Water communities make great strides on clarifying frameworks, terminology, and developing conceptual models for cyberinfrastructure-based environmental observatories. It is agreed that their primary role is to enable hypothesis testing at full scale and verify pilot-study findings through observations at large scales in comparable settings. Observatories enable cross-disciplinary process discovery and understanding through observation, simulation, analysis, and knowledge synthesis. They contain experimental facilities (smaller-scale controllable instrumented catchments) that are strategically implemented within the observatory using nested or gradient design [8]. The former design connects progressively larger order watersheds within the observatory to facilitate scaling studies. The second design connects a set of sites that isolate, at the greatest extent possible, gradients of individual environmental variables or simply leverage existing infrastructure. Simulation models are used to understand processes, to support the observatory design, and predict process aggregation at larger scales using various scaling theories. Critical cyberinfrastructure is needed to enable communication, storage and analysis of the knowledge that is acquired from the deployed sensing systems and other available data sources, as well as modelling, collaboration, and knowledge networking support.

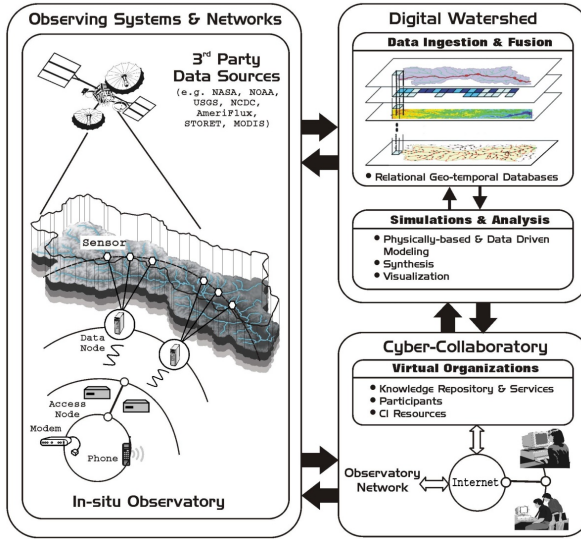


Fig. 1. Components of a water-centric observations system

The Cyber-observatories aim at integrating tools and methods in one place such that the conversion of the data in information and subsequently in knowledge takes place seamlessly using customized workflows (see Figure 2) [5].

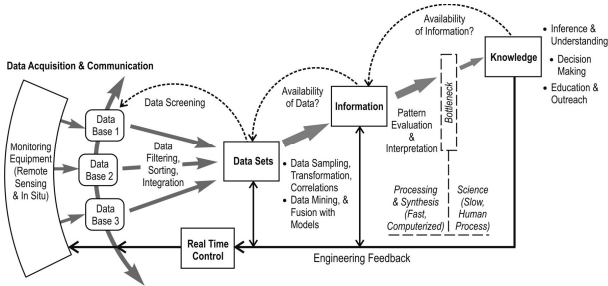


Fig. 2. The data-to-knowledge transformation process taking place in the information system (adapted from Fletcher, 2006)

While providing a detailed technical architecture for a cyberinfrastructure-instrumented environmental observatory is challenging, attempts are made in Figure 1 to provide an overall conceptual architecture [4]. The conceptual framework, still evolving, usefully indicates key cyberinfrastructure elements for the system. Efficient observatory designs are based on open services-oriented architecture that loosely couples self-contained services communicating with each other, and that can be called upon from multiple clients in a standard fashion. This

architecture facilitates automation of time-consuming activities, allowing scientists and engineers to spend more time interpreting and developing insights from data. Additional benefits associated with the services-oriented architecture include: scalability, security, easier monitoring, standards-reliance, interoperability across a range of resources, and plug-and-play interfaces.

Central to the observatory engineered system is the digital watershed (DW) concept. DW is a comprehensive characterization of ecohydrologic systems that use integrated data and simulations models to facilitate study of multi-scale, multi-process dynamics of watersheds (Maidment, 2006). The DW digital description covers both the natural environment and the man-made constructed infrastructure (e.g., dams, water abstraction and discharge systems). It includes time series objects, channel objects, spatially-distributed data layers, etc. A key component of the DW is the data model (DM), which is a permanent information infrastructure that directly links water-flow and environmental processes over large domains and multiple scales. DM applies to any type of waterbody, i.e., watershed, lake, estuary, coastal area. It digitally describes watershed features including GIS data (terrain, stream network, soils, land cover, geology), hydrologic observation data (streamflow, groundwater levels), weather and climate data (precipitation, air temperature, relative humidity) collected locally, by remote sensing or produced by weather and climate models, upland and in-stream water quality data (pollutant types and levels, habitat characteristics), and socio-economical data within the boundaries of the waterbody. Access to 3rd party and locally acquired multi-disciplinary data is made available through web services. Historical data and recent observational information are stored in a relational database structure so as to be communicated via web-portals to geographically remote users.

Modelling and analysis included in the DW entail process conceptualization and methodologies for mapping process input to output. They create a unifying framework for the synthesis of field information, improvement of sampling strategies, testing of hypotheses, and identification of optimal management strategies for complex systems that minimize cost (including environmental degradation) and maximize the performance of a water body through feedback loops. Conventional model-based simulation approaches make use of a sequential process, based on difficult data collection and preparation, periodic and disconnected simulations, followed by further disconnected post-processed visualization and analyses. This approach increases cost through poor use of human and computer resources, loss of information, and excessive offline operations that increase the time lag between questions and answers. This fragmented approach forces individuals to think inside the box by using highly simplified models over small geographic areas, and results in a significant gap between what is possible and practical, and between basic research and the practice of modelling/visualization. Customized metaworkflows allow to link heterogeneous workflows (set of operations that can be executed in order to automatically pass the data from one operation in the sequence to the next) and software tools (often created by different users using multiple software technologies) into user-friendly interactive systems where the data-simulation model fusion takes place in real time.

The cyber-collaboratory (CYC) is essentially a web portal that allows sharing of resources, models, data and ideas by scientists, managers and any other interested stakeholders in virtual organizations. The observed and computed informations are stored in a digital library (DL), which is a basic database that contains historical and new data preserved for further analysis, fusion, visualization, added-value processing, and dissemination. The library indexes disparate sources of data, models and information using standardized metadata description of each source. Successful implementation of processed/synthesized data requires rigorous validation and documentation of the quality/uncertainty of the stored data and information. Virtual environments enable to employ dynamic visualization of the stored data in which interactive intervention is possible and encouraged, with heavy dependence on geographic information systems and translation through multiple space/time scales.

2.2 Composite Applications

Water-related processes are very diverse, still they interfere and influence each other. Systems were developed for many of these processes, however, due to the complexity of these individual systems on the one hand, and the technological limitations and heterogeneous processing of the information systems, on the other hand, these systems work mostly independent.

The need to use and combine heterogeneous applications under a common framework, led to the definition of composite applications concept. Composite application is a relatively new term in computer sciences, it expresses a perspective of software engineering that defines an application built by combining multiple existing functions into a new application. Composite applications can be built using any technology or architecture [3]. A composite application consists of functionality drawn from several different sources. The components may be individual selected functions from within other applications, or entire systems whose outputs have been packaged as business functions, modules, or web services [10].

The concept reflects the influence of the globalization tendency applied to the information technology sector. Large scale business applications are now developed by a large number of individuals, over a long period of time and on a variety of platforms, using different technologies. Even more complex business applications that provide some flexibility are only good at automating series of processes but these often are bridged by individuals using third party data transfer functionalities or even basic copy-paste procedures. This slows down the entire mechanism severely and grinds productivity to a complete halt in the absence of the human factor. Moreover, some applications become necessary after the initial components have been built and since most of existing software is monolithic and offers poor integration or interoperability features, stitching the components together to make a single entity becomes almost impossible. As such, the usual approach to this problem was to rewrite the components from scratch and then create a new monolithic, integration challenged piece of software [2].

This is why we often see basically the same features ported across different platforms under various brand names. Another problem is that software developers have been relentless to even take this into consideration because they usually create functionalities that offer an acceptable level of security, integrity and are closed source (for evident economic purposes). Moreover, since no official specifications have ever been compiled that offer an acceptable, cross platform technology standard, working with this in mind only complicates application structure needlessly.

Under these circumstances, Composite Applications represent the right approach for the development of medium and large IT - systems. Composite applications help companies, to deal with heterogeneous IT structures and to ensure stable system operation. Appropriate practice and procedure models are used, besides the actual techniques and tools for the development of the systems.

3 System Architecture

The development of a Water Resource Management System, a web portal based on composite application, requires several phases: the identification of business processes and data flows, the system modeling, the system and software architecture design, implementation and testing.

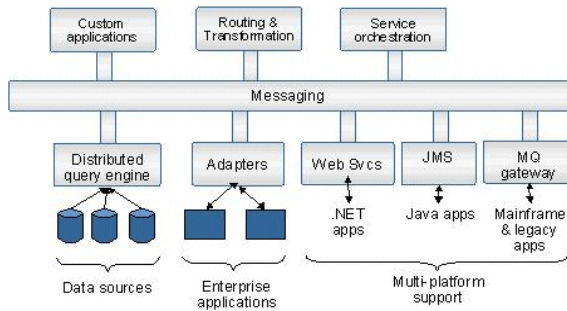


Fig. 3. Composite Application Architecture

During the modeling phase and system and software architecture design phase, an important target is the definition of services and interfaces, which allows a step by step development of the system. The architecture of the portal developed by the team, named *Hydro NETWORK*, is presented in fig. 3.

Such a framework can be used to wrap existing systems, using SOA adapters for exposing their functionality as web services, in this way, making them remote-accessible and specific water resources management systems, like Water Management Information System (WISKI), WaterWare or Hydstra, to the SOA messaging transport layer or other locally developed applications.

In the development of the composite application for water resources management, after establishing the system architecture and choosing the implementation technologies, the next step was the design of the graphic user interface of the Hydro Network portal. The first service that was developed, accessible through the portal, is the service for monitoring river basins. The river basin or catchment is the geographical area where from a fluvial system, consisting of a river and its tributaries, gathers its waters. Through its characteristics, the river basin strongly influences the hydrologic processes and phenomena. That's why it represents a reference unit of great importance for hydrological studies.

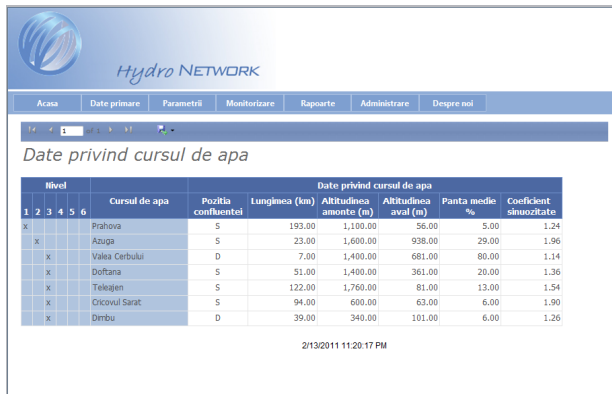


Fig. 4. Tree representation of a river, its tributaries, and their characteristics

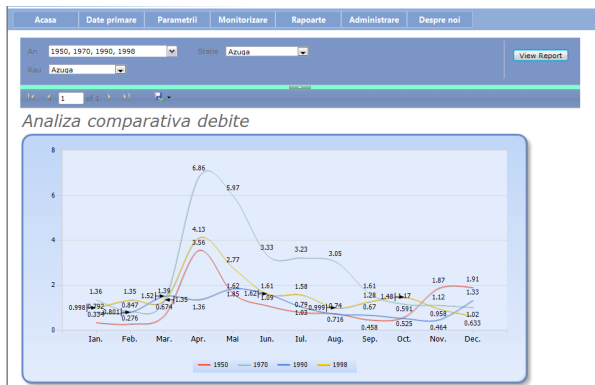


Fig. 5. Comparison of river flows

The hydrological sciences define with great accuracy the characteristics of almost all types of waters, still the information available about the Romanian rivers is not easy to obtain from public sources, and even less is available on the internet. The developed application is based on data from the Cadastral Atlas of Romanian Waters, which contains a detailed description of river parameters and historical records of the river data.

The description of the river basin is represented in a tree type, which allows the identification of tributary level (figure 4). Based on the existing records, the application allows six levels of tributaries. For each of them, hydrographical data are stored.

The application allows visualization of primary data and of a large variety of reports. The reporting tools enables representation both in table form and as charts or diagrams (figure 5).

4 Technologies

The application was developed using Microsoft Visual Studio 2010 Professional, an integrated environment that simplifies the creation, debugging and implementation of software, in particular of services. The .NET Framework 4 was used for the development and operation of next-generation applications and XML Web services, while Windows Communication Foundation (WCF) was used for building service oriented applications. By using WCF, data can be transmitted from an „endpoint” to another, through asynchronous messages. An endpoint service can be part of a continuous service hosted by IIS, or it may be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. Messages can be simple characters or words, sent as XML, or streams of binary data.

.NET Language-Integrated Query (LINQ) was chosen as it allows a holistic approach to the data sources that adds query facilities for general use in .NET Framework [6]. They apply to all sources of information, not just to relational databases or XML data. .NET Language-Integrated Query defines a set of standard query operators that allow cross transaction, filtering, and search operations to be expressed in a declarative manner, in any programming language based on .Net. For the data base implementation, Microsoft SQL Server 2008 R2 was used [6]. Microsoft SQL Server 2008 R2 Reporting Services and Microsoft SQL Server Reporting Services Report Builder 3.0 offered a complete platform, server based, designed to support a wide variety of reporting requirements that enable organizations to provide relevant information where needed across the enterprise. [10]

For the data layer, data sources are diverse. Historical data have to be converted from spreadsheet files, database files, or require even manual input, while more recent and real-time data require 3S technologies. 3S Technology is an organic whole consisting of GPS, RS and GIS. It is an important support technology to obtain, store and manage, renew, analyze and apply spatial data, as most of data referred by the water resources information are spatial data related to the geographical position [4].



Fig. 6. Database structure for reporting services

The data model for the reporting application is presented in figure 6. The tables of AppWS database contain characteristics of rivers and of hydrometric stations.

The reporting services create two additional databases, ReportServer\$CIPROS and ReportServer\$CIPROSTempDB.

ReportServer\$CIPROS is a SQL Server database that stores simple and linked reports, shared data sources, templates, logfiles for reports and security settings for the reports. ReportServer\$CIPROSTempDB is a temporary database for the report server database that stores data about sessions and runtime, reports saved in the cache memory and intermediate tables.

5 The References Section

Surface waters are classified into stagnant water (seas and oceans, lakes, etc.) and running waters (streams - rivers). Groundwater arise from precipitation that infiltrates into the soil, or from water infiltration from river or lake beds.

It is known that between surface water and groundwater is a very close connection. In general, the rivers feeding the aquifers in mountain areas receive water from underground in lowland areas (fig. 7) [1]. Due to this strong interdependence, any change in flow regime and surface levels may lead to changes in groundwater levels and vice versa.

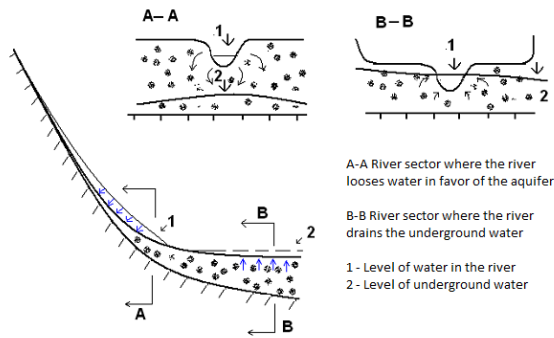


Fig. 7. Mutual influence of groundwater and river flow

For the study of the mutual influence of groundwater and river flow, the proposed composite application integrates two services that model the two water types - surface waters and stagnant waters. There already exist applications that model the two processes, but the composite application will allow us to research their mutual interference.

GMS software (Groundwater Modelling System) is used for modelling the groundwater and its characteristics. The results will be used both independent and in another application that computes surface water characteristics. The combined use of both applications will generate the behavioural model of the aquifer, influenced by the surface water. For the aquifer computation, a finite element method is used.

The river basin is divided in sectors. The computation is based on a balance equation, some parameters being considered constant for a sector and a time interval [7]. The two applications communicate on database level. Based on the results, even though the model is still a simplified one, some conclusions can be derived. Using input variables that express the environmental changes for the two applications described above, predictions about the water level can be made. Based on these results, appropriate decisions can be taken.

6 Conclusions

The development of integrated water management systems must cover a large variety of natural phenomena that interfere with human action. Hydroinformatics is an area that just targets responsible use of the Earth's most precious resources - water, by implementing the knowledge acquired over time about water use, in high quality software applications. This can be realised using the advantages of composite applications framework. The presented Hydro NETWORK is far from being a complete solution. It is a core that will be extended through additional services. The progress requires development of more accurate theoretical models, as well as input of geo-hydrological data.

References

1. Cheveresan, M., Drobot, R.: A Conceptual Data Model for Flood Management at a River Basin Scale. *Scientific Journal - Series: Mathematical Modelling in Civil Engineering* 3, 5–11 (2010) ISSN 2066-6926
2. Ciobanu, R., Dobre, C.: Data Dissemination in Opportunistic Networks. In: *Proceedings CSCS-18, 18th International Conference on Control Systems and Computer Sciences*, Bucharest, May 24-27 (2011)
3. Culita, J., Dumitrascu, A., Stefănoiu, D.: Real-time Monitoring and Forecasting of the Ecological Processes. In: *Proceedings CSCS-18, 18th International Conference on Control Systems and Computer Sciences*, Bucharest, May 24-27 (2011)
4. Constantin, A.G., Radulescu, F.I.: OLAP Cube data analysis package. In: *Proceedings CSCS-18, 18th International Conference on Control Systems and Computer Sciences*, Bucharest, May 24-27 (2011)
5. Maidment, D.: Hydrologic Information System. In: *CUAHSI Workshop*, Austin, TX (2006)
6. Mathas, C.: *Composite Applications erfolgreich entwickeln*. Software & Support Media GmbH. Entwickler press (2010) ISBN 978-3-86802-046-5
7. Muste, M.: Toward the Integration of Watershed Science and Management. In: *Proceedings 32nd IAHR Congress*, Venice, Italy (2007)
8. Muste, M., Adler, M.-J., Drobot, R., Mocanu, M.: Cyberobservatory for Supporting Science, Management, and Public Warning: International Case Studies. In: *BALWOIS 2010 - Ohrid, Republic of Macedonia - 25 (May 29, 2010)*
9. Pavlov, S., Belevsky, P.: *Windows Embedded CE 6.0 Fundamentals*. Microsoft Press (2008)
10. Persa, D., Persa, E.C.: A Proposal for Ontology Driven Inter-agents Communication in an Geospatial Data Mining System. In: *Proceedings CSCS-18, 18th International Conference on Control Systems and Computer Sciences*, Bucharest, May 24-27 (2011)
11. Pietraru, V., Drobot, R., Selerescu, M.: Modelling the Influence of Mountain River Withdrawal on Groundwater Levels (in Ro), *Hidrotehnica*, vol. 23(8) (1978)
12. WATERS (2008): Science, Education and Design Strategy, WATer and Environmental Research Systems Network (Draft February 2008), WATERS Network Headquarters, <http://watersnet.org/> (last accessed March 29, 2008)
13. <http://www.fgmorph.com/>
14. <http://msdn.microsoft.com/library/bb308959.aspx>
15. <http://www.microsoft.com/sqlserver/en/us/product-info/overview-capabilities.aspx>

Energy Efficiency Dependency Analysis for a Data Center

Iulia Dumitru^{1,2}, Stéphane Ploix², Ioana Făgărășan¹, and Sergiu Stelian Iliescu¹

¹ Department of Automatic Control and Industrial Informatics,
University POLITEHNICA of Bucharest, 313 Avenue Splaiul Independentei
060032 Bucharest, Romania

² G-SCOP Laboratory, INP Grenoble, 46 Avenue Felix Viallet
38031 Grenoble, France
iulia.dumitru@aii.pub.ro, {ioana, iliescu}@shiva.pub.ro,
stephane.ploix@inpg.fr

Abstract. Data centers energy efficiency has become an issue given the rising of energy prices and the increase of Web applications, which are hosted, mainly in data centers. In order to understand the opportunities for improving data center energy efficiency, it is necessary to determinate the real energy requirements for one computer application. This article will examine the power chain starting from the facility level and ending at the processor level. In this paper a data center energy model is proffered along with two important indicators that address energy efficiency: the first one is addressing Facility Efficiency and the second one is addressing IT Efficiency. The energy model is taking into account a global overview of the data center (power distribution units, cooling system and the IT system). The aim of the research is to find effective solutions to make data center reduce power consumption while keeping the desired quality of service or service level agreement.

Keywords: data center, energy efficiency, energy management, efficiency indicators, simulation and modeling.

1 Introduction

Data centers are mission-critical components of large companies and frequently cost hundreds of millions of dollars to build. Yet few persons understand the true cost of managing and operating such facilities. It is estimated that the total data center energy consumption as a percentage of total US energy consumption has doubled between 2000 and 2007 and is set to double yet again by 2012 [1]. The high utility costs of such an increase is reason enough to address power consumption.

About five years ago, the need to define and measure energy efficiency was recognized and it became obvious that the increasing amount of energy necessary to power and to cool the IT facilities is also a potential threat to business profitability.

Information technology operations are a crucial aspect of most organizational operations. One of the main concerns is business continuity. Companies rely on their information systems to run their operations. It is necessary to provide a reliable infrastructure for IT (information technology) operations, in order to minimize any chance of disruption. A data center contains primarily electronic equipment used for data processing (servers), data storage, communications (network equipment), specialized power conversion and backup equipment.

2 Data Center Power Chain

Data centers use a significant amount of energy to supply three key components: IT equipment, cooling and power delivery. The energy required can be better understood by examining the power used for each level.

The main power supply entering data centers is used for lights, cooling system and security fire protection. Then electricity is converted from AC to low-voltage DC power in the Power Distribution Unit (PDU). The low-voltage DC power is used by the server’s internal components, such as: the central processing unit (CPU), memory, disk drives, chipset and fans.

The first step in prioritizing energy saving opportunities is to gain a solid understanding of data center energy consumption. In a data center there are a lot of components that consume power. The visible components that consume power are: the racks, the servers, the network cables and the power cables. But what it is not visible are the CPUs running programs and the constant flow of information in and out. From [2], Figure 1 that outlines a typical data center energy consumption ratio can be depicted. This is a power analysis of a typical highly available dual-power path data center with N+1 CRAC units, operating at a typical load of 30% of design capacity.

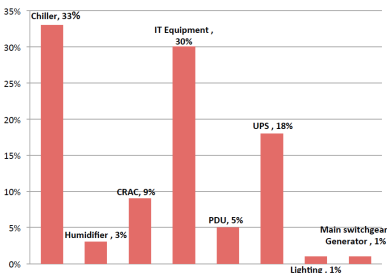


Fig. 1. Power consumption in a typical data center

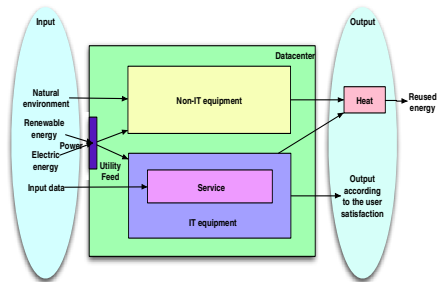


Fig. 2. Data Center Model

In Figure 2, an energy repartition in a typical data center it is proposed. A data center is divided into IT equipment such as servers and non-IT equipment such as air-conditioning. Energy is required as input data into the data center and an output, that has to cope with the customer satisfaction, is obtained. The final energy is

composed of the grid energy and renewable energy. Output is also accompanied by heat, which can be reused. Energy losses can also be found as output in the data center. Since the natural environment (for example, a site location in a cold area) is a large factor of input, it was also incorporated into this model [3] and [2].

The power consumption is not constant with time but varies according to different parameters. The main are the workload of the data center and the outside environment. Modeling the energy efficiency and the losses of the data centers equipment is a complex tasks and crucial assumptions yield great errors. First of all, the assumption that the losses associated to the power and cooling equipment are constant with time is wrong. It has been observed that the energy efficiency of this equipment is a function of the IT load and the non-IT equipment (devices responsible for cooling and power delivery). In addition, these equipment are usually operating at lower than the maximum capacity loads and this increases the system's losses. Finally, the heat generated by the non-IT equipment is not insignificant see [2].

3 Data Center Indicators

An indicator/a metric is a scale for measuring some important characteristics of a system. This includes a procedure or methodology for making the measurement. Implementing a metric allows the manager of a system to know how well the system is performing at some point in time. This makes it possible to adjust one or more parameters of the system and assess system's impact when measured again utilizing the same metric. In this way it becomes possible to optimize whatever aspect of the system that the metric quantifies. To obtain a specific desired goal for the system, however, the particular metric one utilizes must be chosen carefully. Using the wrong metric in this process will lead to either erratic or invalid results.

Energy efficiency metrics can be used to shape energy efficiency strategies, and to determine the effectiveness of the measures to reduce energy consumption. Over time, the data center managers and industry people tried to develop a universally accepted metric for data center efficiency [4].

Today, there are many metrics that communicate different aspects of data center efficiencies, and there are more being created. One should not look at only one metric in particular. The same is true for assessing the efficiency of a data center. A data center is best characterized by a series of metrics that should interact with each other.

In this chapter two important indicators that address energy efficiency will be presented. One is addressing Facility Efficiency and one that is addressing IT Efficiency.

3.1 Power Usage Efficiency – PUE

The PUE (Power Usage Effectiveness) - see [5] - is defined, as a ratio between the Total Facility Power and the IT Equipment Power. The PUE indicates how much power is used by the facility infrastructure to power and cool the IT equipment, and to power the redundant distribution systems required for maintaining the expected availability and reliability.

If the data center were 100% efficient, all the power supplied to the data center would reach the IT equipment. In the real world however there are numbers of ways the electrical energy is consumed by devices other than IT loads, because of the practical requirements of keeping IT equipment properly housed, powered, cooled and protected so that can provide its useful computing. Non-IT devices that consume data center power include such things as transformers, uninterruptible power supplies (UPS), power wiring, fans, air conditioners, pumps, humidifiers and lighting.

PUE is a very complex metric. To correctly use this metric it is very important to understand first of all the load components and second the categories of measurement. Below the components and the sub-components involved in the computing of this metric as well as the different levels of implementation and the categories for measurement of the PUE are summarized see [6] and [7].

- **IT Equipment Power** This includes the load associated with all the IT devices, i.e. compute, storage and network along with supplemental equipment i.e. KVM switches, monitors, and workstations/laptops used to monitor or otherwise control the data center.
- **Total Facility Power** This includes all IT Equipment power as described above plus everything that supports the IT equipment load.

There are two fundamentally different methods for obtaining data necessary to make the PUE calculation. One can either estimate the power by using available information on the equipment factoring in appropriate operational and ambient properties, or one can measure the actual power consumption of the required components.

For a dedicated data center, the total energy in the PUE equation will include all energy sources at the point of utility handoff to the data center owner or operator. For a data center in a mixed-use building, the total energy will be composed of all the energy required to operate the data center, similar to a dedicated data center, and should include cooling, lighting, and support infrastructure for the data center operations.

Table 1 is presenting different levels of implementation for the PUE with the position of each power meter ([8]) as well the categories for each one (the four categories for the measurement of the PUE were proposed in U.S. Department of Energy (2010)).

Table 1. The different levels of implementation and the categories for PUE measurement

	PUE ₀ , PUE ₁ Level 1 (Basic)	PUE ₂ Level 2 (Intermediate)	PUE ₃ Level 3 (Advanced)
IT Equipment Power Measurement	UPS	PDU	Server
Total Facility Equipment Power Measurement	Data center Input power	Data center power input power less shared HVAC	Data center input power less shared HVAC plus building lighting
Minimum Measurement	Monthly	Daily	Continuous

The PUE of a data center is not a static value. Varying server and storage utilization, the fractions of design IT power actually in use, environmental conditions, and other variables strongly influence PUE.

The PUE is a metric that can deliver useful information about the losses in the data center. It is known that different types of cables and different components can provide different losses.

Because the PUE is a metric that becomes a concern for many data center managers, a great number of enterprises developed PUE calculators (42u.com and Baudry K J Site) or proposed algorithms [10].

While PUE it is a good metric to drive overhead power use down, it doesn't address the efficiency with which the computing equipment is applied. For example, you could have a meager low-level of compute utilization, but still achieve a small PUE value, if the denominator from the formula increases. This metric provides an overall measurement of the infrastructure efficiency i.e. higher values relative to the peer group suggest higher potential to improve the efficiency of the infrastructure systems (HVAC, power distribution, lights) and vice versa.

The PUE metric can be also depicted as divided into the following components ([6]):

$$PUE = \frac{\text{Cooling Load Factor (CLF)} + \text{Power Load Factor (PLF)} + \text{IT Load Factor (ILF)}}{\text{IT Load}} \quad (1)$$

Where:

- Cooling Load Factor (CLF) - Represents the total power consumed by chillers, cooling towers, computer room air conditioners (CRACs), pumps, etc.
- Power Load Factor (PLF) - Represents the total power dissipated by switch-gear, uninterruptible power supplies (UPSs), power distribution units (PDUs).

In this chapter it will be showed that even if PUE was very quickly implemented and adopted by the industry, it doesn't show how effective the energy is used by the IT equipment in a facility. This metric mainly indicates how efficient is the cooling equipment.

The power consumed by the cooling components and the power consumed by the delivery components are all related to the power consumed by the IT equipment. Based on this, we can establish a relation between the power consumed by the cooling components, the power delivery components and the power consumed by the IT equipment. It can be stated that the power consumed by the cooling equipment is equal to the power consumed by the IT equipment plus the power used by the delivery components.

$$P_{cooling}^{heat} = P_{IT}^{heat} + P_{delivery\ components}^{heat} \quad (2)$$

Also:

$$P_{cooling}^{heat} = e \bullet P_{cooling}^{electric} \quad (3)$$

$$P_{IT}^{heat} = e \bullet P_{IT}^{electric} \quad (4)$$

$$P_{delivery\ components}^{heat} = e \bullet P_{delivery\ components}^{electric} \quad (5)$$

Where e is an amplification factor.

On the other hand, the Total Facility Power represents the sum of the power consumed by the IT Equipment, the Power Delivery Components (UPS, switch gear, generators, PDUs, batteries and distribution losses external to the IT equipment), the Cooling system components (chilies, computer room air conditioning units (CRACs), direct expansion air handler (DX) units, pumps, cooling towers) and other miscellaneous component loads such as data center lighting.

$$\text{Total Facility Power} = P_{delivery\ components} + P_{cooling} + P_{IT} \quad (6)$$

From this statement it can be inferred that:

$$PUE = \frac{\text{Total Facility Power}}{IT\ Power} \quad (7)$$

$$PUE = \frac{P_{cooling}^{electric} + P_{IT}^{electric} + P_{delivery\ components}^{electric}}{P_{IT}^{electric}} \quad (8)$$

$$PUE = \frac{P_{cooling}^{heat} + P_{IT}^{heat} + P_{delivery\ components}^{heat}}{P_{heat}^{IT}} \quad (9)$$

$$PUE = 2 \bullet \frac{P_{cooling}^{heat}}{P_{heat}^{IT}} \quad (10)$$

This equation reinforces what it was said: this metric does not account for the energy efficiency of the IT itself [11].

Since this metric does not account for the efficiency of the IT itself, it is important to note that if a data center has a low PUE, there may still be major opportunities to reduce overall energy use through IT efficiency measures such as virtualization, etc. The ability to decrease PUE is also affected by climate (e.g. free cooling offers much greater potential in cooler climates).

The PUE metric it will be more relevant and will deliver more relevant information if it is used along with a Cooling Metric. The cooling metric which is proposed is the Data Center Cooling System Efficiency. This metric is defined as a ratio between the Average Cooling System Power Usage and the Average Cooling Load in the Data Center. The cooling load represents the amount of heat that must be removed and it is equal to the rate the heat is generated plus the net flow of heat into the system not associated with cooling.

3.2 Data Center Energy Productivity - DCeP

The focus on the efficient delivery of power to IT equipment has been useful, but given that infrastructure efficiencies are flattening out and processing power continues to improve exponentially. Any effort to manage the energy efficiency of the data center must start seriously focusing on IT efficiency as well.

The ideal metric should work on whatever workload the data center is currently processing and not substitute or inject any sort of synthetic workload. The metric should take into account that not all tasks that a data center performs have equal value to the end user or the business interests of the owner of the data center. It should account for the fact that the value of a task is a function of time. Based on the needs of the customer (which may be formalized in a Service Level Agreement [SLA] contract), some tasks have more or less value depending on the elapsed time required to run to completion. Other tasks have a constant value up to some absolute time deadline. When the deadline passes, the value drops, perhaps to zero. This metric should also be able to account for these concepts. The metric should recognize the fact that each data center operator will have a different take on the value of each of the various tasks that it runs during a given period of time. Finally, the metric should utilize LEAN principles, in other words, it should only give credit for work that is useful to the end customer see [6].

An SLA [12] sets the expectations between the consumer and provider. It helps define the relationship between the two parties. It is the cornerstone of how the service provider sets and maintains commitments to the service consumer.

The SLA encapsulates many "behind-the-scenes" factors that contribute to energy consumption.

While the PUE metric mentioned above is extremely useful to provide insight on, and help manage, facility infrastructure, they provide no guidance as to how to manage the IT equipment within the data center. Understanding the limitations of existing metrics determines us to look for means to measure the output of these facilities metrics that establish the Useful Work produced by the data center.

The DCeP ([13]) is a metric that quantifies the useful work that a data center produces based on the amount of energy it consumes. This is the first metric that measures the actual productivity of the data center, because it takes into consideration the level of satisfaction related to Service Level Agreement.

Useful work is defined to be the sum over i of all tasks $\mathbf{1}$ through \mathbf{M} initiated within the assessment window multiplied by a time-based utility function according to the SLA. The factor V_i assigns a normalized value to each task so that they may be algebraically summed. T_i eliminates all tasks that are either initiated prior to the assessment window or are initiated within the window but do not complete.

Based on the need of the customer this metric would be fine if the so-called useful work wouldn't be so hard to compute. In conclusion a current weakness of this formulation is that the metric is quite difficult to compute and that the definition of the metric requires some manual intervention to define tasks types and assign values and utility functions or each task type. Because of this, other productivity proxies were developed such as: Weighted CPU Utilization.

4 A Data Energy Center Model

Energy proportionality is the degree to which the electrical energy consumed by a data center depends on the traffic load placed on it. An energy model is needed to guide the optimization process.

The model must account both for the energy consumed by servers and the energy consumed by the data center infrastructure supporting those servers. Rather than developing a single unified model, the problem will be approached from a number of different angles, cover several design choices, and synthesize a collection of data center models. Since data centers are highly complex and evolving systems, some architectural details will be abstracted away.

Understanding and modeling data center power consumption is a complicated undertaking. Individual data centers are highly complex systems, with a number of interacting mechanical, electrical and computational sub-systems. The power consumed by the cooling system, for example, can depend both on the nature of the airflow in the server room and on the server load balancing algorithms being used.

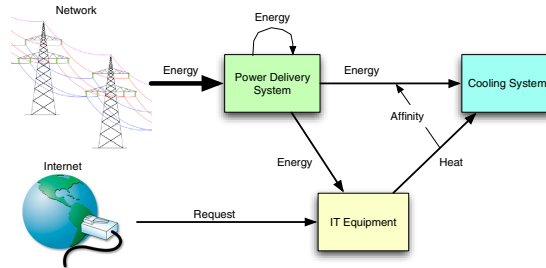


Fig. 3. Energy path in a typical data center

Figure 3 explains the existing relations between the different components of a data center. The energy consumed by servers is influencing the consumption of all the other components. Thus, minimizing the server's consumption will have significant repercussions on the overall consumption of the data center.

The approach is to focus on the three data center sub-systems that typically account for almost all of the power consumption. It outlines how they consume electricity, and describes the mathematical formula that models the relationship between each sub-system's electricity consumption and the data center's utilization level.

The three sub-systems on which the approach will focus are [14]: IT equipment (servers), Cooling (CRACs, chillers) and Power distribution (UPS's, PDUs). It was assumed that other sub-systems, such as lighting draw a small (1%) and fixed amount of power, independent of utilization levels.

4.1 IT Equipment Energy Consumption

Even if the IT equipment power includes the load associated with all the IT devices, i.e. compute, storage and network along with additional appliances i.e. KVM

switches, monitors, and workstations/laptops used to monitor or otherwise control the data center, in the approach will be used only the energy model for the servers.

The studies ([15]) concluded that server’s power consumption could be approximated with reasonable accuracy using a simple linear model. Idle data centers consume large amounts of electricity (more than half of their peak power in the studied data centers) and electricity consumption rises, roughly linearly, in function of CPU utilization u . Thus:

$$P_{IT}(u) = P_{idle} + (P_{peak} - P_{idle}) \cdot u \tag{11}$$

Where: P_{idle} is the total power consumed by the servers when they are all idle; P_{peak} is the total power when all servers are at peak load; P_{idle} and P_{peak} depend on the specific server hardware used and the number of servers in the data center, and so can vary from data center to data center.

If the server hosts virtual machine, then, the P_{idle} can be expressed as a function of number of cores that the server has and the number of virtual machine hosted:

$$P_{idle}(N_{core}, N_{VM}) = P_{idle\ default} + (N_{maxcore} - N_{core}) \cdot a + N_{VM} \cdot b \tag{12}$$

Where: $P_{idle\ default}$ is the total power consumed by the servers when they are all idle and there is no virtual machine running on the server; $N_{maxcore}$ are the server’s number maximum of a cores; N_{core} are the server’s number of core used; N_{VM} are the total number of virtual machines running on the server; a, b constant that depend on the server’s type

Because the server’s numbers of cores that are used can be changed only from Bios, and it cannot be modified this number dynamically, the equation can be reduced at:

$$P_{idle}(N_{core}, N_{VM}) = P_{idle\ default} + N_{VM} \cdot b \tag{13}$$

In Table 2 several servers as well as their characteristics are presented. The power curves are constructed using data from published SPECpower benchmarks [16]. The SPECpower benchmark is designed to simulate how efficiently a machine can run a typical business application on an enterprise Java platform.

Table 2. Different types of servers

Machine	CPU	Cores	Disk	Rate	Peak Power	Idle Power	a	b
Fujitsu TX150	Xeon	4	HDD	0.08	112 W	24.3W	-	-
IBM X3250	Xeon	4	HDD	0.06	113 W	42.3W	-	-
SGI XE250	Xeon	8	HDD	0.10	322 W	187 W	-	-
Dell R610	Xeon	12	SSD	0.27	242 W	61.9W	-	-
HP DL170h	Xeon	48	SSD	1.00	952 W	219 W	-	-
G-SCOP	Intel	4	HDD	-	78.39W	48.3W	2.5	1

4.2 Cooling System Energy Consumption

The electricity entering a data center is dissipated as a large amount of heat that must be evacuated from the building. The following abbreviations will be used in the algorithm:

- **PRF** - power on the floor - power used by the facility
- **PIT** - power needed by computers and additional IT equipment
- **PPDU** - power used by the power delivery units
- **PCRAC** - power used by the computer room air conditioning unit fans
- **PChiller** - power used for refrigeration by the chiller unit

The total power on the floor can be express as [17]:

$$P_{RF} = P_{IT} + P_{PDU} + P_{CRAC} + P_{Chiller} \quad (14)$$

Computer Room Air Conditioners (CRAC's) and fans are used to remove hot air from servers on the data center floor and bring in fresh cooler air. Conventional CRAC's transfer heat from the air to a fluid coolant (e.g., water) that is then pumped to large chillers or cooling towers in another part of the facility. The heat is expelled into the external atmosphere and the cooled fluid is circulated back to the CRAC's.

The cooling system's energy it is modeled as a function of the thermal load placed on it.

The sections below develop CRAC and chiller energy models. In the modeling it will be ignored the energy consumed by other components like pumps and humidifiers.

4.2.1 CRAC Unit

The cooling cost is defined as [18] and [19]:

$$P_{CRAC} = \frac{P_{IT}}{CoP(T_{sup})} \quad (15)$$

Where:

- CoP is the coefficient of performance of the CRAC set to supply cold air at T_{sup} temperature
- T_{sup} is the temperature of the supplied cold air and it characterizes the efficiency of a CRAC, i.e. it is defined as the ratio of the amount of heat removed by the cooling device to the energy consumed by the cooling device performing the removal.

The CoP model used for water-chilled CRAC units in a HP utility data center it is [18]:

$$CoP(T) = (0.0068 \cdot T_{sup}^2 + 0.0008 \cdot T_{sup} + 0.458) \quad (16)$$

4.2.2 Chillers

Chillers extract heat from a coolant fluid and return that fluid to the CRAC units. Among other factors, chiller power consumption depends on the amount of heat extracted and on the selected return temperature for the fluid.

Note that the thermal load can be expressed as a function of the IT power $P_{IT}(\mathbf{u})$ and so is a function of CPU utilization \mathbf{u} .

All raised floor power needs to be cooled by the chilling system, which required power for refrigeration (under steady state condition, the total raised floor power equal to the total cooling power):

$$P_{chiller} = \frac{P_{RF}}{CoP} \tag{17}$$

The coefficient of performance is equal at:

$$CoP = -1 + \frac{T_{cwRT}}{T_{chwST}} + \frac{1}{Q_{evap}} \cdot \left(\frac{q_{evap} \cdot T_{cwRT}}{T_{chwST}} - q_{cond} \right) \cdot f_{HX} \tag{18}$$

where:

- T_{cwRT} = entering (return) condenser water temperature (Kelvin)
- T_{chwST} = leaving (supply) evaporator water temperature (Kelvin)
- Q_{evap} = evaporator load
- q_{evap} = rate of internal losses in evaporator
- q_{cond} = rate of internal losses in condenser
- f_{HX} = dimensionless term

Most papers assume an average CoP= 4.5 [17].

The HVAC and data center communities have developed chiller power models Table 3 that will be used directly. A set of chilled water plant models specified by the California utility Pacific Gas and Electric (PG&E) [20] as energy efficiency guidelines for data centers are deployed. The PG&E models are for entire chilled water plants were measured following the requirements:

- 24H/ 7 days
- The baseline chilled water supply temperature set-point is $T_{chwST} = 6$ °C, constant
- The baseline cold condenser water temperature set-point is $T_{cwRT} = 26$ °C, constant

Table 3. Chiller function

Chiller	Capacity	Full load	Power model
HP (variable)	600 ton	0.28 kW/ton	$P_{chillers}(u) = 0.022+0.055 \cdot u+0.026 \cdot u^2$
HP (constant)	650 ton	0.36 kW/ton	$P_{chillers}(u) = 0.009+0.017 \cdot u+0.054 \cdot u^2$
PG&E (A)	>300 ton	0.68 kW/ton	$P_{chillers}(u) = 0.030+0.069 \cdot u+0.094 \cdot u^2$
PG & E (B)	<300 ton	0.85 kW/ton	$P_{chillers}(u) = 0.013+0.104 \cdot u+0.122 \cdot u^2$

4.3 Power Distribution System Energy Consumption

Data centers have intricate power management infrastructures that accept high voltage AC power from an electrical utility, process that power, and deliver an uninterrupted low-voltage supply to the IT equipment. This infrastructure can impose a significant overhead: even in recently built facilities, distribution losses can account for more than 10% of the total electricity [21].

The efficiency of the power distribution system is related to the electrical load placed on it. Since the loads (IT and cooling) are functions of utilization, the distribution system's losses can be also modeled as a function of data center CPU utilization u .

In a conventional data center, power from the utility is first converted to medium voltage power (using a large and efficient transformer) and then fed into one or more central uninterruptible power supplies (UPS's). The role of the UPS is to provide temporary power if utility power fails, until local generators can be brought online.

Table 4. PDU function

PDU Type	Efficiency 40 % load	Efficiency 10% load	Loss Model
Standard	96%	93%	$P_{PDU}(u) = 0.0065 + 0.0080 \cdot u + 0.0553 \cdot u^2$
Energy Star	97%	96%	$P_{PDU}(u) = 0.0037 + 0.0055 \cdot u + 0.0512 \cdot u^2$
Premium	99%	97%	$P_{PDU}(u) = 0.0033 + 0.0166 \cdot u^2$

Table 5. UPS function

PDU Type	Efficiency 40 % load	Efficiency 10% load	Loss Model
APC [23] (delta-c)	96%	88%	$P_{UPS}(u) = 0.013 + 0.006 \cdot u + 0.011 \cdot u^2$
APC [23] (double-c)	91%	75%	$P_{UPS}(u) = 0.013 + 0.006 \cdot u + 0.011 \cdot u^2$
Eaton [24]	94%	88%	$P_{UPS}(u) = 0.059 \cdot u + 0.013 \cdot 0.007 \cdot u + 0.074 \cdot u^{1.843}$
Average [24]	91%	79%	$P_{UPS}(u) = 0.024 + 0.027 \cdot u + 0.038 \cdot u^{2.315}$

Sometimes redundant parallel UPS's are used, so that a UPS failure can be masked. Power from the UPS is then stepped down to a lower voltage (commonly using less efficient transformers) and then delivered to power distribution units (PDUs) that are located near servers. Each PDU performs another voltage conversion, along with other power conditioning tasks, and supplies multiple IT equipment racks (a PDU may supply 20-60 racks, with 10-80 servers per rack, depending on the types of servers and PDU capacities [22]). Some energy is also lost in the wiring and switchgear.

For the PDU the following function expressed in Table 4 are deployed. The used models for the UPS can be found in Table 5.

5 Validation of Algorithm and Results

The model described above was implemented in Matlab/Simulink. Figure 4 shows the Simulink energy model and Figure 5 depicts the IT model.

Table 6. The parameters used in simulation

The parameters used in simulation	Simulation (Figure 6)	Simulation (Figure 7)
Enter the server model 1 to 6 (1: Fujitsu; 2:IBM; 3:SGI; 4:Dell; 5:HP; 6:G-SCOP)	6	5
Number of virtual machines	3	1
Numbers of servers	1	2
Ambient temperature from 20 to 38	30	26
Enter the chiller model 1 to 4	4	4
Enter the PDU model 1 to 3	2	1
Enter the UPS model 1 to 4	1	1
Resulting PUE	1.53	1.04

Running a simulation test the following results illustrated in Figure 6 and Figure 7 were obtained:

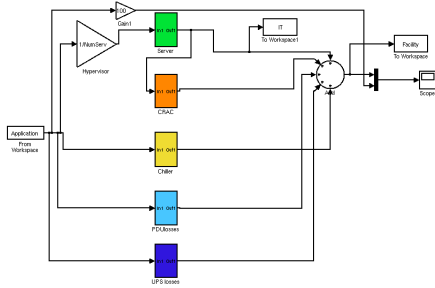


Fig. 4. Data Center Matlab Model

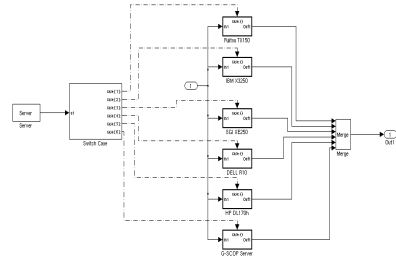


Fig. 5. IT Matlab Model

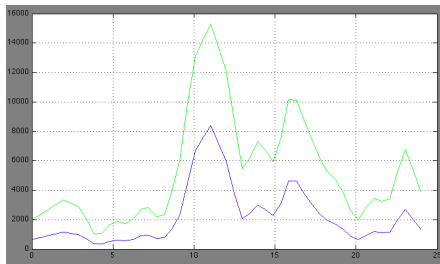


Fig. 6. Result 1

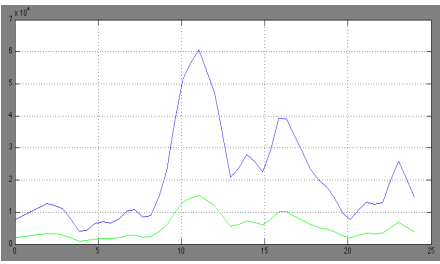


Fig. 7. Result 2

6 Conclusions

Data centers are highly complex and evolving systems and in the process for increasing their energy efficiency a global approach has to be considered. This article has examined the power delivery chain starting from the facility level and ending at the processor level. A data center energy model is proffered along with two important indicators that address energy efficiency: the first one is addressing Facility Efficiency and the second one is addressing IT Efficiency. The proposed energy model takes into account the different types of servers with different levels of availability and power consumption, as well as the global overview of the data center (power distribution units, cooling system and the IT system). The energy consumed by servers is influencing the consumption of all the other components. Thus, minimizing the server's consumption will have significant repercussions on the overall consumption of the data center. The problem is to minimize the energy consumed by servers taking into account the global vision of the data center and the different degrees of freedom [25]. An energy model it is very important to test the different optimization scenarios and to examine the gain obtained by reducing the data center power consumption while keeping the desired quality of service or service level agreement

Concluding, our future work will deal with the problem of selecting a power efficient configuration and a corresponding mapping algorithm for the multiple running applications. The major problem that dynamic migration has to deal with is to decide what is the best location for executing a new job, depending on the resources it requires in order to fulfill its QoS (quality of service) and according with the provider targets for power efficiency, reliability, etc ([26]). The scheduling policies are usually used to balance the systems load effectively or achieve a target quality of service (QoS). The need for a scheduling algorithm rises from the requirement of most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

References

1. Kant, K.: Data center evolution. A tutorial on state of the art, issues, and challenges. *Computer Networks* 53(17), 2939–2965 (2009)
2. Rasmussen, N.: Electrical efficiency modeling for data centers. American Power Conversion by Schneider Electric, White Paper 113 (2009)
3. Iliescu, S.S., Fagarasan, I.: Modern approaches in power system control. In: 16th IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR 2008), vol. 1(1), pp. 41–44 (2008)
4. Dumitru, I., Ploix, S.: Data center control guidelines for obtaining energy efficiency. In: Proceedings of the 18th International Conference on Control Systems and Computer Science, CSCS 18 (2011)
5. Needle, B.J.: Minimizing energy costs in data centers: Energy management by design. *The Magazine of 7x24 Exchange International NewsLink* (2008)
6. Belady, C., Patterson, M.: The green grid productivity indicator. *The Green Grid White Paper* 15 (2008)

7. Power, E.N.: Energy logic: Calculating and prioritizing your data center it efficiency actions. A White Paper from the Experts in Business-Critical Continuity™ (2009)
8. Avelar, V.: Méthode de calcul de l'efficacité énergétique (PUE) dans les datacenters. American Power Conversion by Schneider Electric, White Paper 158 (2010)
9. Verdun, G., Azevedo, D., et al.: The green grid metrics: Data center infrastructure efficiency (DCiE) detailed analysis. The Green Grid White Paper 14 (2008)
10. Ursianu, V., Moldoveanu, F., Ursianu, R., Ursianu, E.: Software quality assurance for monitoring and control systems in the energy field. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS 18), vol. 2(1), pp. 811–818 (2011)
11. Dumitru, I., Fagarasan, I., Iliescu, S., Said, Y., Ploix, S.: Increasing energy efficiency in data centers using energy management, pp. 159–165 (August 2011)
12. Wustenhoff, E.: Service level agreement in data center. Sun BluePrints OnLine (2002)
13. Haas, J., Monroe, M., Pflueger, J., Pouchet, J., Snelling, P., Rawson, A., Rawson, F.: Proxies proposals for measuring data center efficiency. The Green Grid White Paper 17 (2008)
14. Qureshi, A.: Power-demand routing in massive geo-distributed systems. Thesis - Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science (2010)
15. Warkozek, G., Drayer, E., Debusschere, V., Bacha, S.: A new approach to model energy consumption of servers in data centers. In: IEEE International Conference on Industrial Technology ICIT (2012)
16. Benchmark, S.: ssj2008, <http://www.spec.org/powersj200>
17. Rajarshi, D., Kephart, J.O., Jonathan, L., Hendrik, H.: Utility-function-driven energy efficient cooling in data centers. In: Proceeding of the 7th International Conference on Autonomic Computing, ICAC 2010, pp. 61–70. ACM, New York (2010)
18. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling "cool": Temperature-aware resource assignment in data centers. In: Proceedings of the 2005 US NIX Annual Technical Conference (2005)
19. Tang, Q., Gupta, E.K.S., Varsamopoulos, G.: Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. IEEE Transactions on Parallel and Distributed Systems 19(11) (2008)
20. PG&E, High tech energy efficiency baselines: Data centers, PG&E's Customized New Construction and Customized Retrofit Incentive Programs (2010)
21. Hamilton, J.: Cloud computing economies of scale. Amazon Presentation (2010)
22. Xiaobo, F., Wolf-Dietrich, W., Andre, B.L.: Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th Annual International Symposium on Computer Architecture, ISCA 2007, pp. 13–23. ACM, New York (2007)
23. Avelar, V.: Making large ups systems more efficient. American Power Conversion by Schneider Electric, White Paper 108 (2006)
24. Eaton, Eaton: Ups efficiency calculator, <http://powerquality.eaton.com/calculator/>
25. Said, Y., Bergeon, S., Ploix, S., Giap, Q., Dumitru, I.: Towards a global energy management within a data center, pp. 1–8 (October 2011)
26. Dumitru, I., Stamatescu, G.: Dynamic management techniques for increasing energy efficiency within a data center. In: The 1st Unite Doctoral Symposium (2011)

Intelligent Building Management: A Service-Oriented Approach

Catalin Chera, Laurentiu Bucur, and Serban Petrescu

Computer Science Department, "Politehnica" University of Bucharest, 060042 Romania
catalin.chera@cs.pub.ro, {laur.bucur,bspetrescu}@gmail.com

Abstract. This paper introduces the current design of an intelligent building management framework. This framework consists of a service-oriented device network, a simple controller that can interoperate with a variety of devices, platforms, and networks, and a set of applications that allows users to design and compose control policies and services. This framework allows users to contribute information about policies, devices, and control services, and allows end users to compose control and policy services to be executed.

Keywords: service oriented, intelligent building management, policy based computing.

1 Introduction

This paper presents the overall architecture for the FCINT project [1] (Ontology-based Service Composition Framework for Syndicating Building Intelligence). This project aims to provide a service-oriented approach to control and manage building facilities via intelligent controllers.

Recently, numerous smart home projects have been initiated around the world, and they addressed various aspects of home applications and control. Specifically, Tiresias.org lists hundreds of smart home projects ranging from security, solar and wind energy efficiency, home appliances, smart workplace, voice-activated control, RFID, to elderly care and comfortable living for the physically impaired, hearing impaired, and visually impaired people. In the U.S., Arizona State University [2],[3], Duke University [4], Iowa State University [5] and Washington State University [6], among many other universities, have their smart home projects. In Europe, the SOFIA project [7] created its own version of smart home that outlines several generations of smart home. The DEHEMS project (www.dehems.org) and the FCINT project have also advanced in that direction. The Hydra project [8] just ended involved many countries and proposed a service-oriented network to support smart home applications. The SENSEI project connects devices with ontology for cyber-physical applications.

Organizations such as Cisco, Home Depot, ETRI, IBM, Intel (Home Energy Management) [9], Microsoft (Microsoft Future Home) [10], Philips, Siemens [11], and Toyota [12] have their own visions and projects. For example, the Microsoft Future Home uses voice recognition, cellular phones, cloud computing, visual and interactive wallpaper, and intelligent human-computer interaction to enhance daily living. The Siemens Smart Home Solution [11] is based on cellular phones and addresses comfort, security, energy (HVAC and lighting), healthcare, communication, and entertainment.

These projects have different focus and objectives, and involve different technologies. For example, Washington State University focuses on data mining and learning from data to support various activities and consumption; the HYDRA project proposed a tiered architecture and initiated a service-oriented approach for facility management; the SOFIA project aimed at providing great life experience supported by knowledge engineering; the Intel project provides computing processors to support smart home applications; the Siemens project leverages its large pool of devices and know-how to support a wide range of home activities. One may classify these projects as related to hardware (such as Intel), to devices, to data mining and optimization, to energy (solar panels and efficient buildings), to knowledge engineering (ontology, reasoning, and learning), and to user interface and experience (such as interactive wallpapers).

This project aimed to develop an infrastructure for people to share their building control services using a service-oriented platform. It will be an open platform to allow different people to participate and contribute: end users may contribute their profiling and preference data, device companies can supply device information including functional descriptions and input/output, software developers can contribute their control and policy services. In this way, both users and developers can compose new services by mashup for their application.

This project will provide an infrastructure for controlling building facilities with the following features:

- A. The infrastructure should minimize the change to the existing infrastructure or installation;
- B. The infrastructure should allow different people to contribute different parts of the system including information related to control devices, control policies, buildings, weather, and energy consumptions. It should allow changes to be made while keeping track of these changes;
- C. End users can be able to understand, discovery, compose, or select appropriate control or policy services from the server for their buildings based on their own preferences.

This paper is structured as follows: Section 2 reviews the related work on device control. Section 3 presented the roadmap for the FCINT project. Section 4 illustrates the software architecture of the FCINT project. Section 5 concludes this paper.

2 Related Work on Device Interoperability

Recently, significant progress has been made in control devices with respect to device interoperability, e.g., how different devices from different companies can communicate and cooperate to perform missions. Numerous research organizations have proposed service-oriented infrastructure to support device interoperability. Most notable approaches include SODA [13], DPWS, and earlier UPnP and OSGi (www.osgi.org). A common thread of these approaches is to wrap the functions provided by devices as services that can be published, discovered, composed, executed, and monitored, just like software services in enterprise computing.

For example, SODA, supported by SODA Alliance with 27 partners from 6 countries, is an OSGi-based service-oriented architecture for connecting sensors and actuators. The SODA Stack consists of four layers from presentation, ESB, adapter, and device from top to bottom. The Stack provides a logical model for interfacing devices into the enterprise. In this way, the control devices will act like software services in enterprise computing without any software installation. Any new devices wrapped according to the SODA specification can be added to the network, and cooperate with numerous devices already in the network without any modification to the device. The adapter layer, through the SOA binding framework, bridges the gap between the device drivers and the communication to an Enterprise Service Bus (ESB) [14]. Its responsibilities are handling the session-level communication and device registration on the ESB. The Eclipse Foundation [15] has developed a Device Kit for SODA-compliant medical devices. The ESB layer can be any ESB suite installed in the enterprise. The Situation Layer includes all applications that respond and interact with the lower levels.

DPWS, a new international standard based on SODA, is supported by industrial giants such as Microsoft, Nortel, Red Hat, and Schneider Electric. It is an open-source specification on top of SODA. The DPWS uses WS: Addressing [16], WS: Discovery [17], WS: Eventing [18], SOAP [19], and XML [19]. In the EU SIRENA project, Schneider Electric produced early DPWS-compliant embedded devices [20]. Furthermore, SOA4D (Service-Oriented Architecture for Devices) [21] has established a website to host open-source solutions to support SODA, DPWS, and related technologies. As of February 2011, it has listed only 8 projects in two application areas.

WS4D is another organization that provides information related to DPWS. It uses WS-Discovery, WS-Eventing, WS-MetaDataExchange [22], WS-Transfer [23], WS-Security [24], WS-Policy [25], WS-Addressing, SOAP-over-UDP [26], SOAP, WSDL, XML, and XML Schema [19] as its stack of technology. The main applications of WS4D include cellular phones, wireless sensor nodes, and automation devices. UPnP shares many objectives with DPWS, and DPWS may eventually replace UPnP. UPnP has many standard SOA features such as media and device independence, user interface control, operating system and programming language independence, programmatic control, extensibility, and it

supports device addressing (based on IP), discovery, description, control, event notification, and presentation. The standard provides a set of specifications for secure messaging, device discovery, description and eventing from a device exposed as a web service. In DPWS a device is identified by a Hosting Service. Essentially, DPWS brings most SOA features from enterprise computing to the device level.

Many existing standards are available to connect devices in a network for building facility management such as LonWorks [27] (ISO/IEC 14908.1 standard), KNX [28] (ISO/IEC 14543-3 standard), and BACnet, and BACnet is a standard developed by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) supported by approximately 190 device vendors.

3 Roadmap of Control Devices

Smart home controllers can be divided into the following generations:

1. *Networked*: Devices are connected either via wired or wireless networks, and they can be remotely controlled via standard-based protocols such as BACnet standards. This is the current state-of-practice as many modern buildings are in this state, including the Palace of Parliament in Romania. Most devices in the Palace are connected, and many of them have sensors to detect movement. For example, the lights will automatically turn on once the sensors detect human movement in the room in the Palace today.

2. *Service-oriented networks for devices*: Devices are not only connected, but their services can also be published, discovered, composed, executed, and monitored in a service-oriented manner using standard-based protocols such as SOAP, WS-eventing, and WS-addressing. This is the current state-of-the-art, and DPWS is a representative standard. However, the design, implementation, and experience of using this approach are just beginning to appear in the literature. Applying existing standards in SOA leverage significant technologies that have been developed, however, most these protocols and standards have been developed for enterprise computing, the applicability of these to building facility control needs to be validated by further research and experimentation. In this generation, user applications need to be map to device services that have been published.

3. *Policy-based computing with simulation and ontology*: Devices are not only connected on a service-oriented network, but also user control services including policies can be published, discovered, composed, executed (or enforced). This is further supported by simulation so that users can simulate the behavior before deploying policies to be enforced. Currently, there is no standardized solution for device networks at this time. While policy enforcement for enterprise computing has been studied [29],[30],[31],[32], and policy enforcement has standards such as WS-Policy, specific policy enforcement for device networks has not been addressed. Instead, WS-policy has been proposed in the DPWS stack. In WS-policy, policies are enforced at the beginning of a service call in enterprise computing, but policies in a device network may need to be enforced at all times

(e.g., a temperature detector needs to enforce a policy of low temperature for a room at all times). Thus, policy standards for enterprise computing can be used at a higher layer, but at a lower layer, a new kind of policy mechanisms including policy specification, discovery, and enforcement for a device network will be needed. The low-level policy may be device-specific or application-specific, but will often focus on specific functions of the concerned device.

4. *Intelligent control*: In this phase, massive data concerning intelligent buildings such as devices, weather, BIM, and personal preferences will be available for data analysis, and computational intelligence can be applied to data. While numerous data mining and computational intelligence algorithms are available such as those done by the CASAS project, it is necessary to experiment these algorithms on buildings with devices connected in a service-oriented device network.

4 Current Project Components

In figure 1 the overall project architecture is illustrated. Central to the architecture is the client side Smart Building Controller. The client will host a service-oriented infrastructure that allows various services and policies to be installed and configured for execution and device control. Once a new service is loaded, the client will have a different behavior. The client will have an event-driven architecture (EDA) [33] and runs in real time to handle various events.

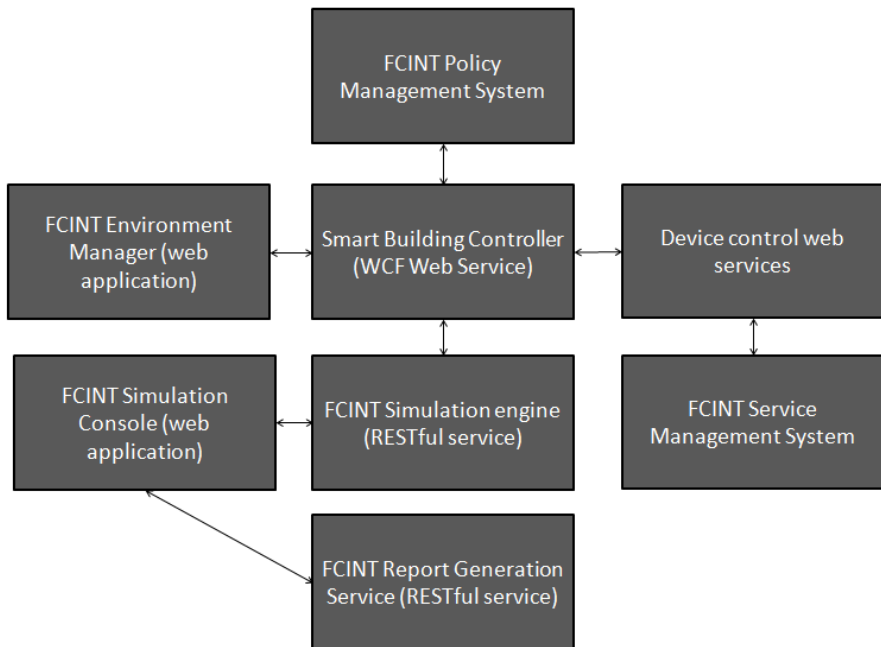
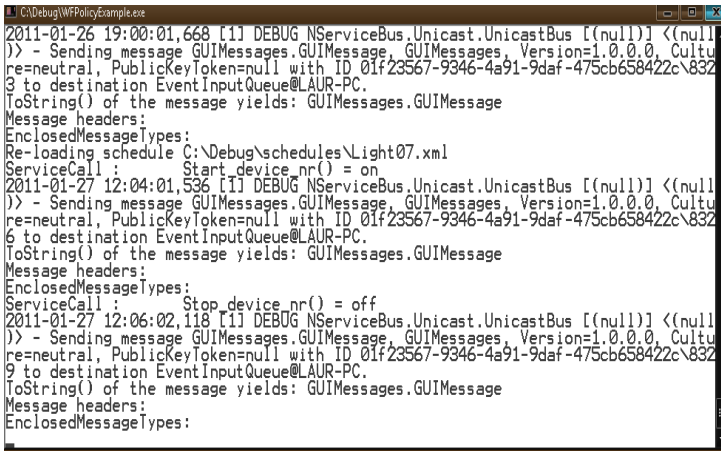


Fig. 1. The FCINT software architecture

4.1 The FCINT Smart Building Controller

The FCINT Smart Building Controller [34] is a service-oriented application that runs in the premises of a building environment to control. It's to control the devices and installations in the building by means of policy-based web service policy orchestration.



```

C:\Debug\WFPolicyExample.exe
2011-01-26 19:00:01,668 [1] DEBUG NServiceBus.Unicast.UnicastBus [(null)] <(null)
>> - Sending message GUIMessages.GUIMessage, GUIMessages, Version=1.0.0.0, Cultu
re=neutral, PublicKeyToken=null with ID 01f23567-9346-4a91-9daf-475cb658422c\832
3 to destination EventInputQueue@LAUR-PC.
ToString() of the message yields: GUIMessages.GUIMessage
Message headers:
EnclosedMessageTypes:
Re-loading schedule C:\Debug\schedules\Light07.xml
ServiceCall : Start_device_nr() = on
2011-01-27 12:04:01,536 [1] DEBUG NServiceBus.Unicast.UnicastBus [(null)] <(null)
>> - Sending message GUIMessages.GUIMessage, GUIMessages, Version=1.0.0.0, Cultu
re=neutral, PublicKeyToken=null with ID 01f23567-9346-4a91-9daf-475cb658422c\832
6 to destination EventInputQueue@LAUR-PC.
ToString() of the message yields: GUIMessages.GUIMessage
Message headers:
EnclosedMessageTypes:
ServiceCall : Stop_device_nr() = off
2011-01-27 12:06:02,118 [1] DEBUG NServiceBus.Unicast.UnicastBus [(null)] <(null)
>> - Sending message GUIMessages.GUIMessage, GUIMessages, Version=1.0.0.0, Cultu
re=neutral, PublicKeyToken=null with ID 01f23567-9346-4a91-9daf-475cb658422c\832
9 to destination EventInputQueue@LAUR-PC.
ToString() of the message yields: GUIMessages.GUIMessage
Message headers:
EnclosedMessageTypes:

```

Fig. 2. The FCINT Smart Building Controller console

A simple scenario for demonstrating the policy-based approach of the SBC operation is a policy of the form:

If the windows are open, stop the air conditioner.

In this scenario there are two web services orchestrated by the SBC:

- Air_Condition_1*
- WindowService*

Air_Condition_1 receives commands from the SBC as web service calls and controls the air conditioning units. The *WindowService* receives status requested from the SBC as web service calls and returns the state of the windows. This policy can be expressed as a rule of the form:

on (WindowService.window_opened_event) *if*
(Air_Condition_1.IsOpened)=true) *then* Air_Condition_1.Close() *do* log_error

The *WindowService* registers the *window_opened_event* with the Smart Building Controller. The event triggers the policy that checks the status of the air conditioner by invoking the *Air_Condition_1* service instance. If the air conditioner is running then the policy enforcement invokes the *Close()* method. If the call to the service results in an error, the compensation action *log_error* is performed. The result of the policy enforcement is translated into the appropriate call to the air conditioning control web service that shuts down the air conditioning devices.

4.2 The FCINT Simulation Framework

The FCINT Simulation Console (figure 3) and Simulation Engine (figure 5) constitute a service oriented web application that allows:

a) The estimation of operating costs (consumption) of devices and facilities in a building, without their mandatory physical installation on site. Optionally, however, the simulator can estimate the energy consumption of real devices and installations controlled by a Smart Building Controller. This is achieved by the service composition mechanism between the FCINT Simulation Engine and the SBC Web service.

b) Testing the response of a Smart Building Controller to external events. This is achieved by creating various simulation scenarios (figure 4) in the simulation console, which contain events that are sent for execution to the Smart Building Controller during the execution of the scenario. The types of events are: Start/Stop commands for real devices and custom events that can originate from a real device. Upon receiving a custom event, the Smart Building Controller executes one or more event-triggered policies that specify the SBC behavior in the event of their occurrence, using a rule-based policy approach.

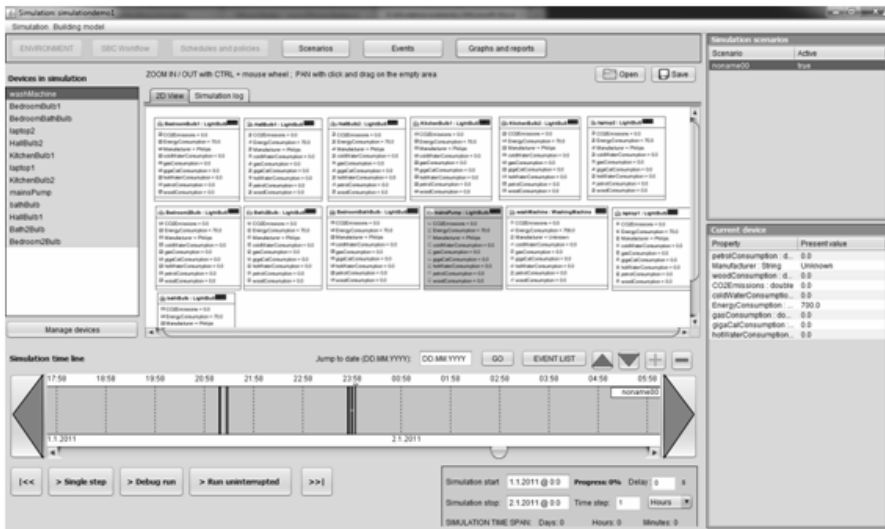


Fig. 3. The FCINT Simulation Console

The FCINT Report Generation service (figure 6) allows the Simulation Console to create a PDF report of a simulation. The report (figure 7) summarizes the energy, water, gigacallory and fossil fuel consumption and estimated costs for the duration of a simulation. The generated reports are PDF files that are an estimate of the total utility bill for the operation of a building environment.

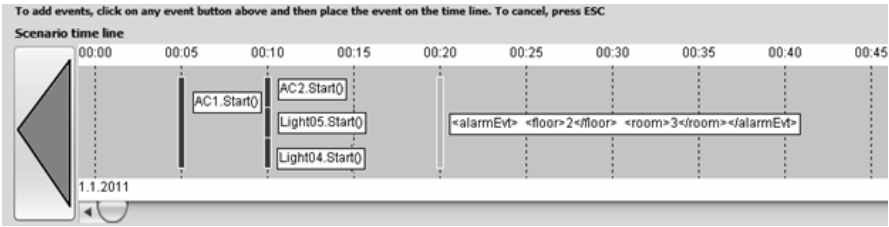


Fig. 4. A simulation scenario containing command and custom events

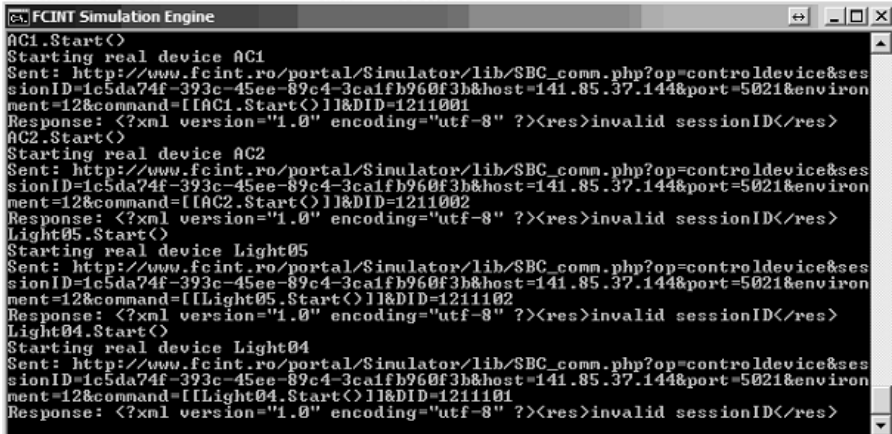


Fig. 5. The FCINT Simulation Engine

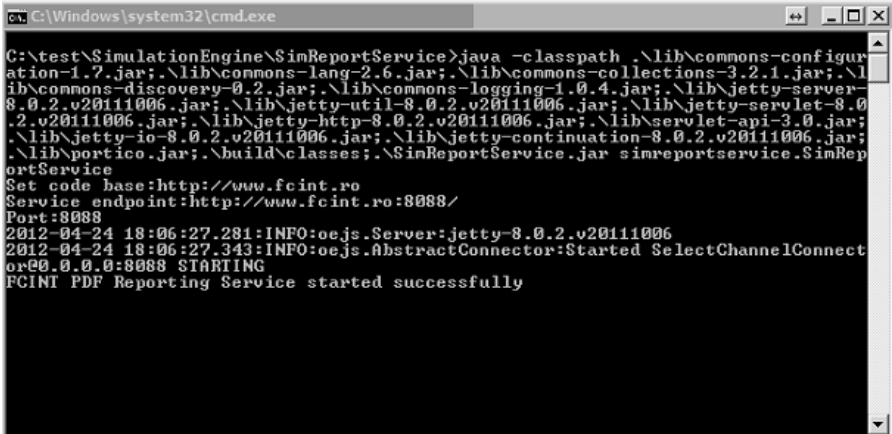


Fig. 6. The FCINT Report Generation Service

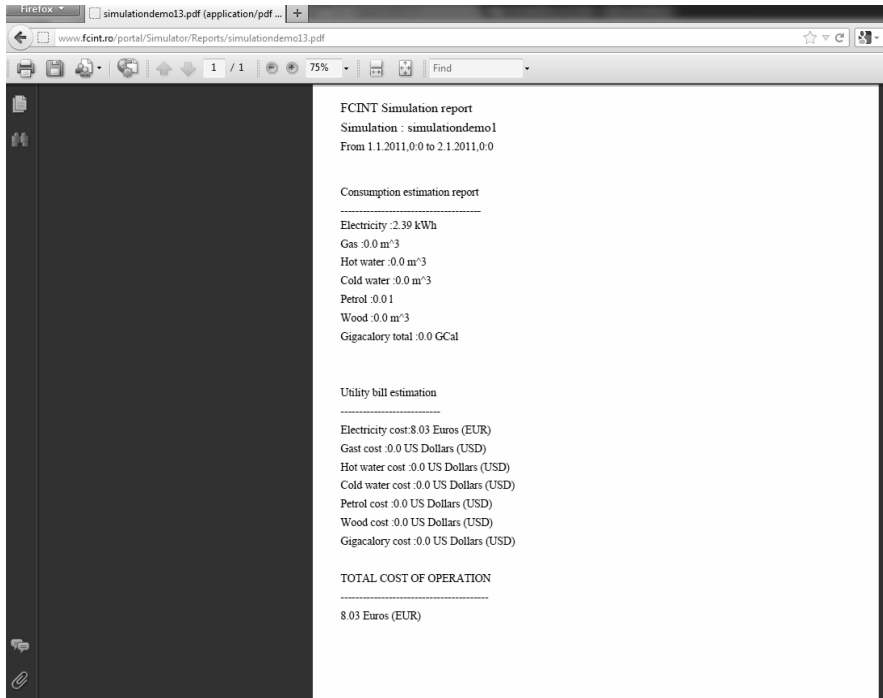


Fig. 7. Sample FCINT PDF simulation report

4.3 The FCINT Policy Management System

In the FCINT Smart Building Controller, users can specify control policies for their buildings according to their preferences. A policy is a set of rules that specifies constraints on the execution of the workflow. In this case, policies apply to the Smart Building Controller's orchestration workflow. Policies are loaded and enforced at runtime by the SBC using dynamic code generation. The FCINT Smart Building Controller uses PSML-P policy specification language [35] for policy enforcement.

A PSML-P Policy is a set of rules of the form:

$$\text{on (E) if (C) then P do A.} \tag{1}$$

where:

- E is an event;
- C is a condition which triggers the execution of a policy;
- P is a property the system must hold;
- A is a compensation action which is executed if P is not satisfied;

The types of events supported by the proposed architecture are:

- a) The beginning and ending of a service call, denoted as S+/-S-. These rules can be used to allow or deny a certain service call.
- b) Data access, denoted by a small capital letter followed by a + for the writing operation and the d- for the reading operation. Example: d+/d-.
- c) An arbitrary event type registered on the ESB by any of the following:
 - Services on the Service Layer;
 - Portal COM module;
 - User Interface Module.

Based on the policy chain, the Smart Building Controller generates a sequential workflow that is executed when the event occurs. A special type of policy rule is the continuous rule that is enforced at each step of the orchestration workflow.

The FCINT Policy Management System is an application that allows the user to create, edit and manage the policies installed in a Smart Building Controller. The user can define both continuous operation policies and event-triggered policies by editing the rules using the user interface shown in figure 8. The user can also enable, disable and delete policies (figure 9).

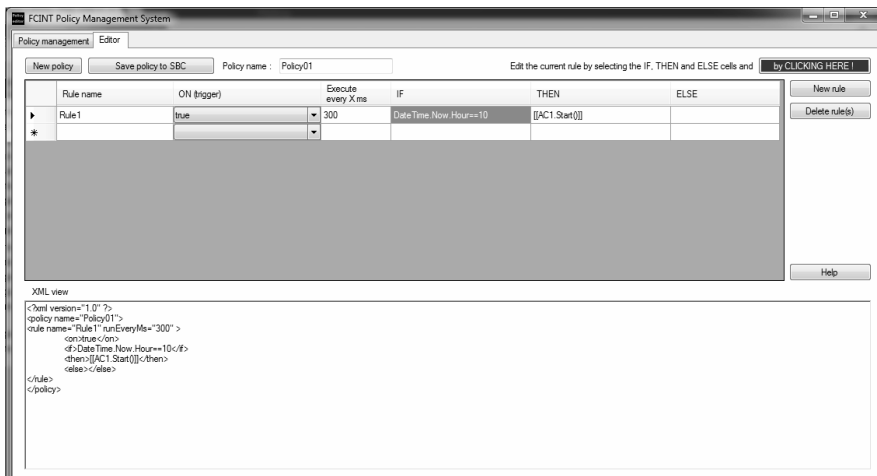


Fig. 8. The FCINT Policy Management System – Policy editing features

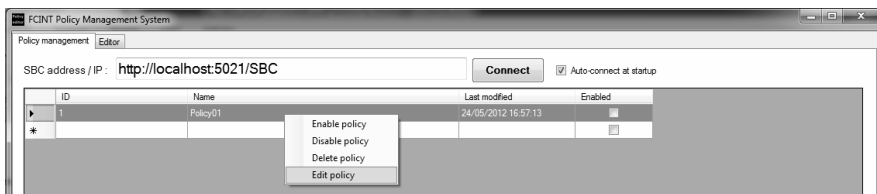


Fig. 9. The FCINT Policy Management System –Management features

4.4 The Device Control Web Services

The Device Control Web Services are web services running in the building that control physical devices and expose their functionality via the WSDL API. The Smart Building Controller orchestrates the execution of the device control Web Services based on user commands and on the set of policies active at any given time. The device control web services are installed, configured and finally tested from the FCINT Environment Manager.

4.5 The FCINT Service Management System

The FCINT Service Management System is a web-based repository where service developers can publish device control web services (figure 10) and where Smart Building Controller administrators can search for services (figure 11) that can be downloaded, installed and configured as device control web services.

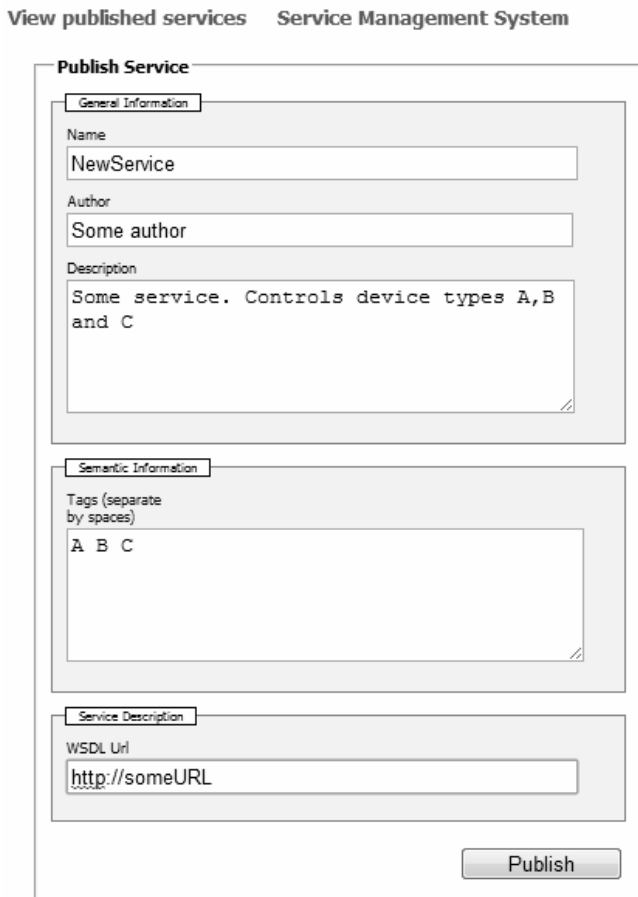


Fig. 10. The FCINT Service Management System service publishing feature

View published services Service Management System

Find Services

Keywords

Search names
 Search authors
 Search descriptions
 Search tags
 Use semantic inference

Published Services (2)

Name:	<input type="text" value="Air Condition 1"/>
Author:	<input type="text" value="Chera Catalin"/>
WSDL:	<input type="text" value="http://141.85.37.144/Air_condition_1/AC.asmx?WSDL"/> Click Here
Description:	<input type="text" value="Air condition"/>
Tags:	HAIER air cool heat dry condition
Name:	<input type="text" value="Air condition 2"/>
Author:	<input type="text" value="Chera Catalin"/>
WSDL:	<input type="text" value="http://141.85.37.144/Air_condition_2/AC.asmx?WSDL"/> Click Here
Description:	<input type="text" value="Air condition"/>
Tags:	HAIER air cool heat dry condition

Fig. 11. The FCINT Service Management System service discovery features

4.6 The FCINT Environment Manager

The FCINT SBC can be accessed externally by a web application – the FCINT Environment Manager (figure 12). The building web control application connects to a Smart Building Controller instance to list the devices and device groups installed. It allows the user to create and configure the calendars and operating schedules of the devices and to visualize the scheduled actions and SBC activity logs for a given time interval. It also allows the visualization of SBC device status and their direct control.

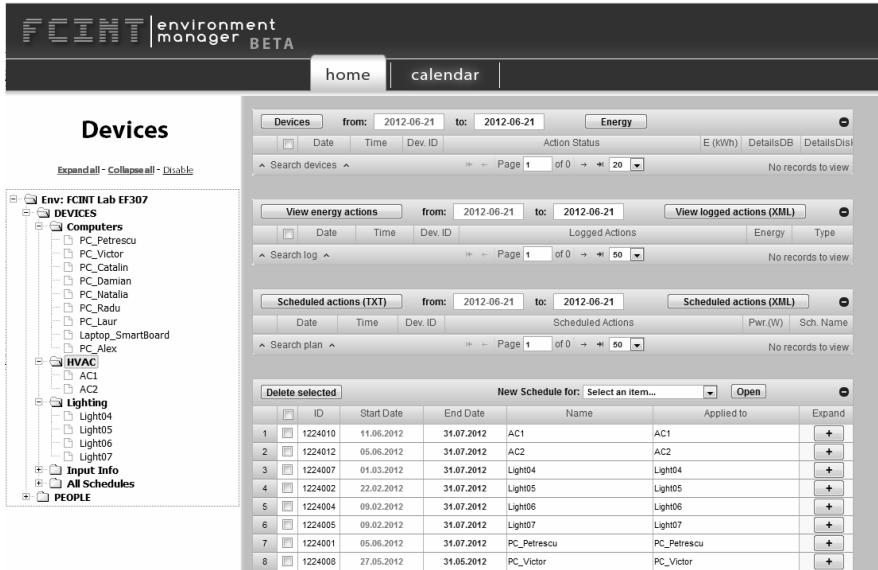


Fig. 12. The FCINT Environment Manager

5 Conclusions

This paper presents the overall architecture of the FCINT project, and it includes a small but smart controller, a set of web applications that allow users to publish, discover, compose, simulate control and policy services and a set of tools that allow the creation, management and simulation-based testing of smart building control policies.

Acknowledgments. The research presented in this paper was supported by the EU POS-CCE project FCINT No. 181/18.06.2010.

References

1. FCINT project, <http://www.fcint.ro>
2. Xu, J., Lee, Y.H., Tsai, W.T., Li, W., Son, Y.-S., Park, J.-H., Moon, K.-D.: Ontology-Based Smart Home Solution and Service Composition. In: ICCESS 2009 (2009)
3. Lee, Y.H., Li, W., Tsai, W.T., et al.: A Code Generation and Execution Environment for Service-Oriented Smart Home solutions. In: Proc. of IEEE SOCA (October 2009)
4. Duke University, Smart Home project, <http://smarthome.duke.edu/projects/list>, http://www.tiresias.org/research/guidelines/smart_home.htm
5. Iowa State University, Smart Home project, <http://smarthome.cs.iastate.edu/index.html>
6. Washington State University Smart Home project, <http://ailab.wsu.edu/casas/>
7. Katasonov, A., Palviainen, M.: Towards Ontology-driven Development of Applications for Smart Environments. In: Proc. Intl. Workshop on the Web of Things at PerCom 2010, Mannheim, Germany, March 29-April 2 (2010)
8. The EU HYDRA project, <http://www.hydramiddleware.eu/news.php>
9. Intel Home Energy Management, <http://edc.intel.com/Applications/Energy-Solutions/Home-Energy-Management/>
10. Microsoft Future Home, <http://www.youtube.com/watch?v=ODpReoKQVXM>
11. Siemens, http://www.siemens.com/innovation/en/publikationen/publications_pof/pof_fall_2008/gebaeude/vernetzung.htm
12. Toyota Dream House, <http://tronweb.super-nova.co.jp/toyotadreamhousepapi.html>
13. The SODA Alliance, <http://www.sensorplatform.org/soda/>
14. Enterprise Service Bus (ESB), http://en.wikipedia.org/wiki/Enterprise_service_bus
15. Eclipse, <http://www.eclipse.org>
16. WS-Addressing, <http://www.w3.org/Submission/ws-addressing/>
17. OASIS Standard. Web Services Dynamic Discovery (WS-Discovery) (July 1, 2009), <http://docs.oasis-140open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.docx>
18. Box, D., et al.: Web Services Eventing (WS-Eventing) (March 15, 2006), <http://www.w3.org/Submission/2006/SUBM-149WS-Eventing-20060315/>
19. Chen, Y., Tsai, W.T.: Service-Oriented Computing and Web Data Management, From Principles to Development, 2nd edn. Kendall Hunt Publishing Company (2010) ISBN 978-0-7575-7747-5
20. Jammes, F., Mensch, A., Smit, H.: Service-Oriented Device Communications using the Device Profile for Web Services. In: MPAC 2005, pp. 1–8 (2005)
21. SODA Alliance, <http://www.sensorplatform.org/soda>
22. Web Services Metadata Exchange (WS-MetadataExchange), <http://schemas.xmlsoap.org/ws/2004/09/mex/>
23. WS-Transfer, <http://www.w3.org/Submission/WS-Transfer/>

24. WS-Security, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
25. WS-Policy, <http://www.w3.org/Submission/WS-Policy/>
26. SOAPoverUDP, <http://schemas.xmlsoap.org/ws/2004/09/soap-over-udp/>
27. The LonWorks 2.0 Platform, http://www.echelon.com/products/lonworks_platform.htm
28. The KNX Standard specifications, <http://www.knx.org/knx-standard/knx-specifications>
29. Tsai, W.T., Chen, Y., Paul, R., Zhou, X., Fan, C.: Simulation Verification and Validation by Dynamic Policy Specification and Enforcement. *Simulation, Transactions of the Society for Modeling and Simulation International* 82(5), 295–310 (2006)
30. Tsai, W.T., Chen, Y., Paul, R., Chung, J.-Y.: Data Provenance in SOA: Security, Reliability, and Integrity. *Service Oriented Computing and Applications Journal* (2007)
31. Tsai, W.T., Huang, Q., Xiao, B., Chen, Y., Zhou, Z.: *Collaboration Policy Generation in Dynamic Collaborative SOA*. IEEE Computer Society Press (2007)
32. Tsai, W.-T., Zhou, X., Wei, X.: A Policy Enforcement Framework for Verification and Control of Service Collaboration. *Information Systems and E-Business Management* 6, 83–107 (2008)
33. Event-Driven Architecture, http://en.wikipedia.org/wiki/Event_driven_architecture
34. Bucur, L., Tsai, W.T., Petrescu, S., Chera, C., Moldoveanu, F.: A Service-oriented Controller for Intelligent Building Management. In: *Proceedings of the 18th International Conference on Control Systems and Computer Science, Bucharest, Romania (2011)*
35. Xu, J.: PSML-O: Ontology Specification Language and its Applications, <http://gradworks.umi.com/33/61/3361858.html>

Intelligent Forecasting of Indoors Ecological Processes

Janetta Culiță, Dan Ștefănoiu, and Alexandru Dumitrașcu

“Politehnica” University of Bucharest,
Faculty of Automatic Control and Computer Science
313 Splaiul Independentei, Sector 6, 060042-Bucharest, Romania
{jculita,dumalex_77}@yahoo.com, danny@indinf.pub.ro

Abstract. The paper addresses the problem of real-time monitoring and intelligent forecasting for indoors ecological phenomena. The process yields a collection of ecological parameters viewed as distributed time series. Data acquired by means of a wireless network of sensors are modeled by using complex algorithms in view of prediction. An evolutionary searching strategy (Particle Swarm Optimization-PSO) is proposed in order to intelligently find the most accurate prediction model. This strategy is adapted to predictors like PARMA, PARMAX, KARMA and FORWAVER, which are implemented within the monitoring and forecasting system, in order to estimate the future evolution of some ecological parameters. The monitoring system was effectively integrated in an industrial application dealing with automatic irrigation of a small greenhouse. The forecasting simulation results with real data and a comparative assessment of best predictor performances obtained with PSO are presented in the end.

Keywords: particle swarm optimization, intelligent forecasting, real-time remote monitoring, multi-variable signals, distributed time series.

1 Introduction

Rapid climate changes and the negative impact of industry upon the environment require designing and employing of an automatic monitoring system of geographical areas. The general purpose of monitoring is to forecast the behavior of the ecological system, in view of disaster anticipation or avoidance.

The ecological phenomena could be evidenced either in an open space or in an enclosed space. In some situations, especially in a microclimate, usually the ecological parameters gathered from different channels (ambient temperature and humidity, dew point, solar radiation) are quite correlated, which could improve their prediction accuracy. In this respect, the parameter variations are interpreted and modeled as a collection of distributed time series (ts).

The paper mainly presents an intelligent technique for forecasting of multi-variable signals, which is integrated in an ecological monitoring and forecasting

system *EcoMonFor*. According to our approach, the ecological signal prediction relies on numerical models (such as ARMA, ARMAX or state space representation) that have previously been implemented as PARMA, PARMAX, KARMA, FORWAVER predictors described in [1], [2]. Due to a variable number of model parameters, an exhaustive searching procedure for finding the optimal (the most accurate) predictor is rather inappropriate and quite inefficient. Consequently, an evolutionary strategy, namely the *Particle Swarm Optimization (PSO)* [3], has been improved and adapted to the ecological monitoring framework. The strategy is employed here in order to select the optimal structural indices of the prediction models that lead to the best predicted values.

EcoMonFor was successfully integrated in a remote monitoring and control architecture designed to match a small greenhouse [4]. The application aim is the automatic watering of plants, in order to yield suitable growth of plants and to increase their life level.

The paper is structured as follows: Section 2 introduces the improved PSO intelligent searching strategy. The distributed architecture for greenhouse monitoring and control is introduced in section 3. The application of PSO procedure on the aforementioned prediction models is achieved in section 4 and the prediction performances are shown within section 5. A conclusion and the references list complete the article.

2 Improved Particle Swarm Optimization Strategy

Recent technological advances and accurate observations on ecological phenomena based on populations of life forms allowed developing a new class of optimization techniques, based on evolutionary strategies. The main idea founding such strategies is to emulate the manner in which life is searching for optimal development, according to Natural Selection Law (firstly formulated by Darwin). Thus, populations of non human entities exhibit an amazing level of intelligence when searching for best food or optimal environmental conditions. The optimum searching mechanism, although of heuristic nature, has been modeled and integrated into several numerical procedures like genetic algorithms, ants or bees behavioral algorithm, and (increasingly) many others.

The PSO procedure introduced next belongs to the same category and seemingly is the best strategy for the monitoring application. The principle of PSO is illustrated in [Figure 1](#). A population of entities, referred to as *particles*, is initiated to run on some trajectories, starting from initial positions. Trajectories are adaptively adjusted on the run, according to some fitness function (that has to be optimized).

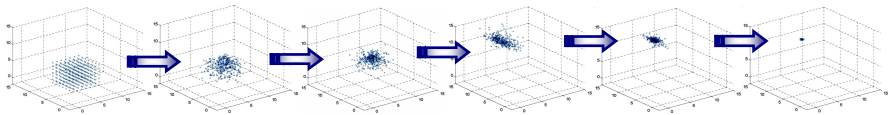


Fig. 1. Principle of Particle Swarm Optimization technique

The goal is to concentrate the population around some optimum of fitness function and to increase the chance that the optimum be global over the searching space. The population is denoted by $\mathcal{P} = \{\mathbf{x}_p\}_{p \in \overline{1, P}}$. Positions of particles are actually accounted in \mathcal{P} (as vectors of some Euclidean space). Usually, the number of particles (P) varies from 50 to 500. The population evolves towards optimum through *generations*. At every generation, particles take new positions, which can be updated by means of additive corrections, like below:

$$\mathbf{x}_p^{m+1} = \mathbf{x}_p^m + \Delta \mathbf{x}_p^{m+1}, \quad \forall p \in \overline{1, P}, \quad \forall m \geq 0, \quad (1)$$

where m is the *generation index*. Corrections are computed as displacements of particles with some speeds with ΔT delay, that is: $\Delta \mathbf{x}_p^{m+1} = \mathbf{v}_p^{m+1} \Delta T$. Speeds can also be updated according to the following numerical recipe:

$$\mathbf{v}_p^{m+1} = \mu_p^m \cdot \mathbf{v}_p^m + \lambda_{p,c}^m \cdot \frac{\mathbf{x}_{p,0}^m - \mathbf{x}_p^m}{\tau_{p,c}^{m+1}} + \lambda_s^m \cdot \frac{\mathbf{x}_{\mathcal{P},0}^m - \mathbf{x}_p^m}{\tau_s^{m+1}}, \quad \forall p \in \overline{1, P}, \quad \forall m \geq 0, \quad (2)$$

where:

- $\mu_p^m \in [0, 1]$ is the particle *mobility factor*; its opposite $(1 - \mu_p^m)$ stands for the *particle inertia*;
- $\mathbf{x}_{p,0}^m$ is the best position of particle, on its way from the beginning to the current generation;
- $\mathbf{x}_{\mathcal{P},0}^m$ is the best position reached by a particle of the whole population (from the beginning to the current generation);
- $\lambda_{p,c}^m \in [0, 2]$ is the normalized *cognitive variance* of particle positions with respect to its best position;
- $\lambda_s^m \in [0, 2]$ is the normalized *social variance* of all particles positions with respect to the best population position;
- $\tau_{p,c}^{m+1} \geq \Delta T$ is the delay of particle transition from current position to its best position;
- $\tau_s^{m+1} \geq \Delta T$ is the delay of population transition from current positions of its particles to the best population position.

The cognitive and social variances are actually facets of group intelligence that can drive the population to a better state or evolution instant. The variables in (2) will be defined in the sequel by arbitrarily setting $m \geq 0$ and $p \in \overline{1, P}$. Note that the definitions hereafter are improved comparing to those introduced in [3]. Denoting by $F[x]$ the fitness function, the mobility factor is:

$$\mu_p^m = \frac{\|F[\mathbf{x}_{p,0}^m]\| - \|F[\mathbf{x}_p^m]\|}{\|F[\mathbf{x}_{p,0}^m]\| - \|F[\mathbf{x}_{p,1}^m]\|}, \quad (3)$$

where $\mathbf{x}_{p,1}^m$ is the worst position of particle, on its way from the beginning to the current generation. The relative cognitive variance is computed in two steps. First, the absolute variance is evaluated:

$$\sigma_{p,c}^m = \sqrt{\frac{1}{m+1} \left| \sum_{i=0}^m (\|\mathbf{x}_{p,0}^m\|^2 - \|\mathbf{x}_p^i\|^2) \right|}. \quad (4)$$

Then the variance is normalized with respect to minimum and maximum values, after updating them with (4):

$$\lambda_{p,c}^m \stackrel{def}{=} 2 \frac{\sigma_{p,c}^m - \sigma_{p,c,\min}^m}{\sigma_{p,c,\max}^m - \sigma_{p,c,\min}^m}. \quad (5)$$

In a similar manner, the relative social variance can be evaluated. Thus, the absolute value leads to the relative value in the same way:

$$\sigma_s^m \stackrel{def}{=} \sqrt{\frac{1}{P} \sum_{p=1}^P \|\mathbf{x}_p^m - \mathbf{x}_{p,0}^m\|^2} \Rightarrow \lambda_s^m \stackrel{def}{=} 2 \frac{\sigma_s^m - \sigma_{s,\min}^m}{\sigma_{s,\max}^m - \sigma_{s,\min}^m}. \quad (6)$$

It is practically impossible to detect the transition delays. Therefore, they are selected at random. More specifically, the *transition frequencies* $v_{p,c}^{m+1} = 1/\tau_{p,c}^{m+1}$ (*cognitive*) and $v_s^{m+1} = 1/\tau_s^{m+1}$ (*social*) are randomly generated in $(0, 1/\Delta T]$.

Equation (2) is the searching engine of PSO strategy. Three terms are contributing to particle speed updating. The first one is based on the current speed. The weight applied by the mobility factor shows that the former speed is strongly attenuated when the particle is close to its best position. This means the particle is rather attracted by such a position. The second term quantifies the cognitive motivation of particle to move, in terms of driven speed. As the population evolves, every particle acquires some experience that can determine its future behavior. As result of particle experience, the *cognitive speed* becomes important when the trajectory is drifted away with respect to its best position. Finally, the third term expresses the influence of population on each particle behavior, naturally referred to as *social speed*. Depending on particle position versus population best position and variance, the social speed increases either when the population is dispersed or the particle is far away from the best position.

The evolutionary character of PSO algorithm is involved by the randomly selected transition frequencies. Population can thus escape from the capture of some local optimum, by jumping onto another zone. Like most of the evolutionary procedures based on populations, the trade-off between diversity and convergence is important. Diversity of population allows one to check for optimum in many

zones of searching space and, thus, to increase the chance of global optimum finding. On the contrary, convergence (or *particles agglomeration*) is necessary to avoid searching oscillations. The key parameter of this trade-off monitoring is the absolute social variance. Three populations can evolve in parallel: the current one, the elite (that keeps all the best positions) and the anti-elite (that keeps all the worst positions). Every time the product between social variances of current generation and elite overflows or underflows some bounds, the diversity-convergence trade-off is violated. Over-floating is pointing to abnormal increase of population diversity, i.e. to oscillatory behavior. To reduce it, crossover between current generation and elite can be performed. Under-floating is caused by concentration of both populations around some optimum, which may decrease the chance to reach for the global optimum. In this case, diversity is increased by performing crossover between current generation and anti-elite. By crossing over two particles \mathbf{x}_1 and \mathbf{x}_2 with speeds \mathbf{v}_1 and \mathbf{v}_2 , respectively, two offspring are obtained:

$$\tilde{\mathbf{x}}_1 = \gamma\mathbf{x}_1 + (1-\gamma)\mathbf{x}_2 \quad \& \quad \tilde{\mathbf{x}}_2 = \gamma\mathbf{x}_2 + (1-\gamma)\mathbf{x}_1, \quad (7)$$

with corresponding speed values:

$$\tilde{\mathbf{v}}_1 = \gamma\mathbf{v}_1 + (1-\gamma)\mathbf{v}_2 \quad \& \quad \tilde{\mathbf{v}}_2 = \gamma\mathbf{v}_2 + (1-\gamma)\mathbf{v}_1, \quad (8)$$

where $\gamma \in [0,1]$ is selected at random (uniformly). When crossover is applied on current population and elite or anti-elite, all of them of size P , the offspring population consists of $2P$ particles. Only the most fitted $(P-1)$ offspring are selected to complete the current population, after removing all its particles, excepting for the most fitted one. (Perhaps this one is the global optimum.)

The PSO algorithm is initiated to run from some initial population (usually, uniformly distributed inside the searching space). To stop the procedure, several tests can be performed. Two very effective tests are the following: the maximum number of iterations was touched (say 100) or the most fitted particle succeeded to survive within the elite population for several generations (say at least 5).

3 Monitoring and Control Architecture of the Greenhouse

The greenhouse consists of six plants that are located in two separated rooms, to create different microclimates. The improper care of plants led to construction of an automatic irrigation system. Figure 2 depicts the distributed monitoring and control architecture of the greenhouse, which integrates: the automatic control system of irrigation (on the left side down), the irrigation system (on the left side up) and, mostly concerned, the ecological monitoring system EcoMonFor (on the right side). Constructively, EcoMonFor was separated in two components: a mobile part, referred to as *EcoMonFor-M*, figured inside the (red) ellipsis and a fixed part, namely *EcoMonFor-F*, represented in the right down corner.

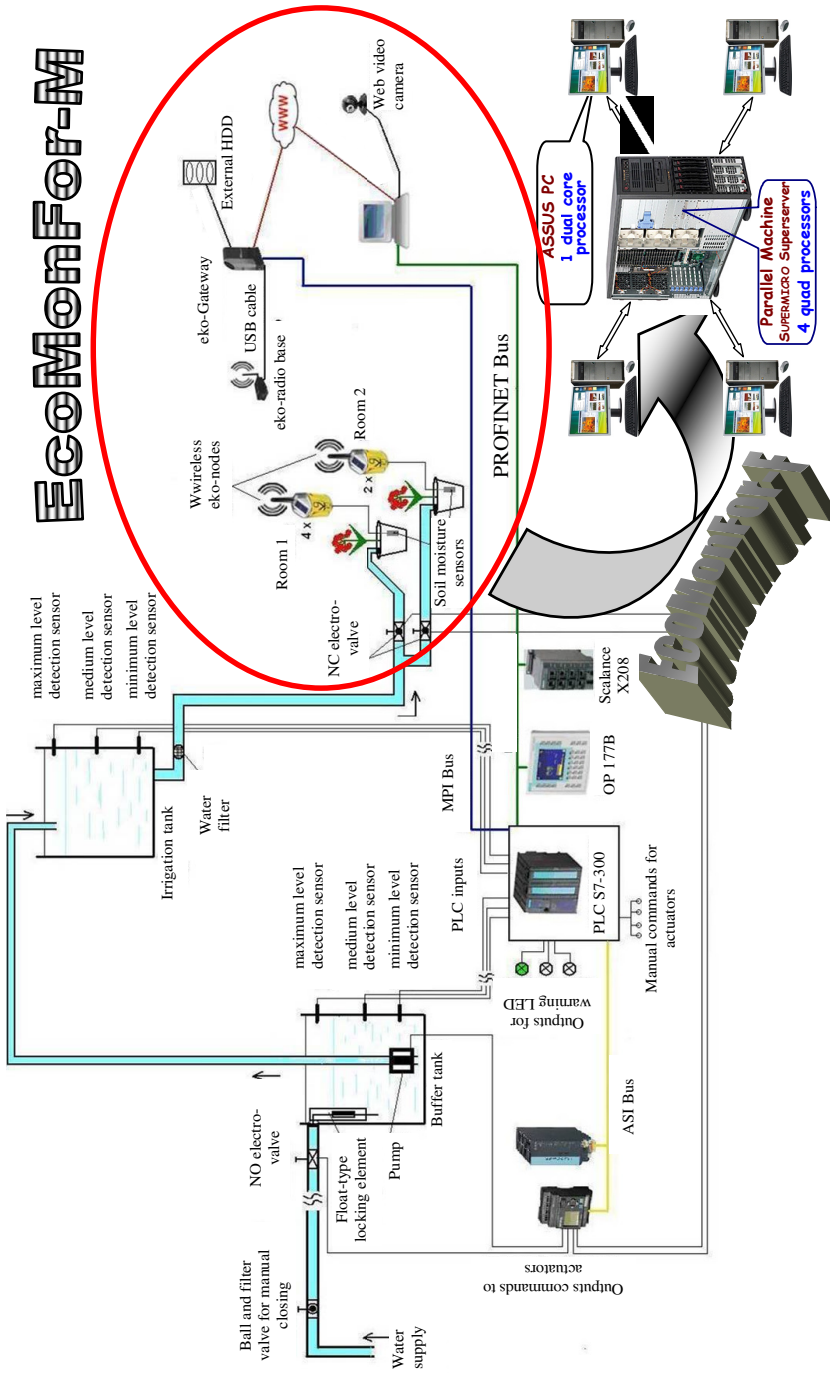


Fig. 2. Monitoring and control architecture of the greenhouse

The mobile monitoring system is structured on three hierarchical levels as shown in the right upper side: the set of wireless eko-sensors; eko-Gateway – a centralizing (kernel) equipment of the sensor network; a computer (or laptop).

The last two components are wirelessly connected to Internet, in order to enable running remote applications. EcoMonFor-M mainly is responsible for remote data acquisition and monitoring, which means it could cover an extended geographical area. The data collection supplied by eko-Gateway module is sent to EcoMonFor-F with the aim of high quality prediction of the ecological phenomena. This strategy is suggested by the curved arrow in the image bottom. The core of the fixed component consists in a parallel computer with 16 processors. This is connected via internet to an extensible computer network. The central unit is hosting complex algorithms for modeling, identification and forecasting of distributed ecological signals, such as: PARMA, PARMAX, KARMA and FORWAVER.

Both components of EcoMonFor are working following a natural strategy. First, the acquisition and the preliminary processing of data are performed. The visual monitoring of the greenhouse stands for the second step, which is executed in parallel with the acquisition and is remotely performed by eko-Gateway. Two web graphical user interfaces are implemented on the computer connected to eko-Gateway. The first one, eko-View, gives the user the ability to set and display the configuration of the sensor network and then to start monitoring and acquisition, from anywhere in the world, as illustrated in Figure 3.

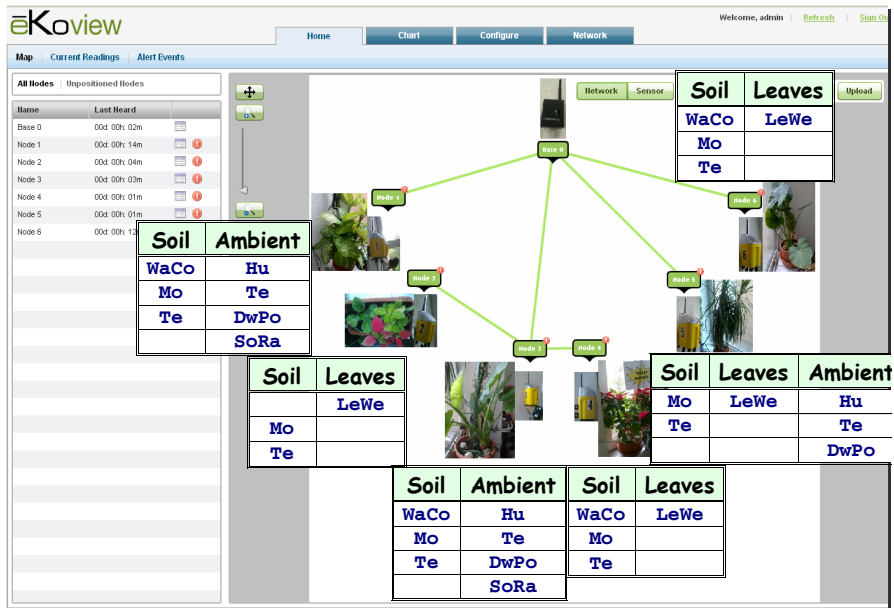


Fig. 3. Synoptic map of the monitored ecological parameters inside the greenhouse

In Figure 3, the plants are represented by their photos. Every node is capable of transmitting data from at most 4 sensors, while a sensor can measure 1-3 ecological parameters simultaneously (for example, there is a singleton sensor for soil moisture and humidity or for ambient humidity, temperature and dew point). Though, the number of the ecological parameters differs for each sensor. Figure 3 also shows the synoptic map of the monitored ecological parameters associated to each acquisition node. A number of 21 sensors have been used, which are transmitting data for 31 ecological parameters, as it can be noticed from the figure. In order to forecast the ecological parameters of the greenhouse and validating the correlations between them data have to be grouped in blocks, according to their possible correlations (e.g., humidity is correlated to temperature which, in its turn, is correlated to solar radiation). Each block corresponds to a node and contains data from 3-4 acquisition channels. In table 1, one can see the varying range and measurement units of the greenhouse parameters.

Table 1. Ecological parameters of sensors network

Soil	Leaves	Ambient
Moisture (Mo)	Leaf Wetness (LeWe)	Humidity (Hu)
0 ... 240 [cbar]	0 ... 1024 [CntS]	0 ... 100 [%]
Temperature (Te)		Temperature (Te)
-40 ... +65 [°C]		-40 ... +65 [°C]
Water Content (WaCo)		Dew Point (DwPo)
0 ... 100 [%wfv]		-10 ... 50 [°C]
		Solar Radiation (SoRa)
		0 ... 1800 [W/m ²]

The second interface, eKo-Greenhouse, is more oriented towards the irrigation application. It aims to help the user to remotely interact with the greenhouse, via internet, by accessing the process parameters and controlling the automatic irrigation system. The final step of EcoMonFor operating strategy corresponds to data modeling on prediction purpose. A convivial graphical interface *eKo-Forecast* was implemented in MATLAB programming language, in order to complete a forecasting experiment [5]. It facilitates running PARMA, PARMAX, KARMA and FORWAVER predictors within *FORTIS (FOREcasting of Time Series)* simulator. The interface offers a graphical illustration of the forecasting results. Although all predictors can proceed on the same fixed or mobile component either, the faster predictors of FORTIS (PARMA and FORWAVER) are commonly hosted by the mobile entity and the slower algorithms, PARMAX and KARMA, are usually executed on the fixed component.

4 Applying PSO to Derive Optimal Prediction Models

The processed data gathered from different channels of the greenhouse are considered as a collection of ny distributed *time series* (ts) $\mathcal{Y} = \{y_j\}_{j \in \overline{1,ny}}$. The classical model \hat{y} of a *ts* is expressed by:

$$\hat{y} \equiv y_T + y_S + y_{ARMA} \tag{9}$$

where the trend y_T and seasonal (periodic) signal y_S reflect a deterministic behavior, whereas the auto-regressive and moving average (ARMA) process y_{ARMA} signifies a stochastic one. Generally, y_T is represented as a low degree polynomial variation:

$$y_T(t) = a_0 + a_1t + \dots + a_pt^p, \quad \forall t \in \mathbb{R}, \tag{10}$$

where the degree $p \in \mathbb{N}$ and the coefficients $\{a_i\}_{i \in \overline{0,p}}$ are unknown. They could be estimated by using the *Least Squares Method* (LSM). The periodic behavior is decoded by using two approaches that employ LSM, as well: a time based one (*Wittacker-Robinson Method*) and a frequency based one (*Fourier-Schuster Periodogram*) [6]. The *Minimum Prediction Error Method* (MPPEM) [7] returns the stochastic component y_{ARMA} .

Modeling and prediction of distributed ts are implemented within four predictors: PARMA, PARMAX, KARMA and FORWAVER. Within PARMA and PARMAX predictors, the vector $\mathbf{y} = [y_1 \dots y_{ny}]^T$ represents the output of a MIMO system. Detailed descriptions of PARMAX, KARMA and FORWAVER predictors are found in [1] and [2]. Hereafter, the PSO-PARMA is only described.

When implementing PARMA, first a (SISO)-ARMA filter is being built:

$$A_j(q^{-1})v_j \equiv C_j(q^{-1})e_j, \quad \forall j \in \overline{1,ny}, \tag{11}$$

where each *ts* v_j ($j \in \overline{1,ny}$) of length $N_y \in \mathbb{N}^*$ is a colored noise produced by the corresponding white noise e_j ($j \in \overline{1,ny}$); A_j and C_j are polynomials of degrees na_j and nc_j , respectively. The white noises corrupting the data are supposed to be Gaussian and uncorrelated. Then, in case of distributed ts, model (11) can be extended to the rough MIMO-ARMA model:

$$\mathbf{A}(q^{-1})\mathbf{v} \equiv \mathbf{C}(q^{-1})\mathbf{e}, \tag{12}$$

by simply considering that the output channels are not correlated to each other. In equation (12), $\mathbf{A} \in \mathbb{R}^{ny \times ny}(q^{-1})$ and $\mathbf{C} \in \mathbb{R}^{ny \times ny}(q^{-1})$ are diagonal polynomial matrices that can be identified via MPPEM. An approximating *auto-regressive* (AR) model with degree $n\alpha$ returns the estimated values of white noise:

$$\hat{e}_j \equiv \hat{A}_j^{n\alpha} (q^{-1}) v_j, \quad \forall j \in \overline{1, n_y}. \tag{13}$$

The model in (13) is identified by *Levinson- Durbin Algorithm (LDA)* [7]. Usually, $n\alpha$ is sensibly larger than the degrees of polynomials in (12).

After estimation of parameters, the optimal predictor can be evaluated. The output predicted data and the corresponding white noise values are recursively computed on a prediction horizon of length $K \geq 1$ (for any $k \in \overline{N_y + 1, N_y + K}$), as follows:

$$\left\{ \begin{array}{l} \hat{v}_j [k | N_y] = -\hat{a}_{j,1} \hat{v}_j [k - 1 | N_y] - \dots - \hat{a}_{j,n\alpha_j} \hat{v}_j [k - n\alpha_j | N_y] + \\ \quad + \hat{c}_{j,1} \hat{e}_j [k - 1] + \dots + \hat{c}_{j,n\alpha_j} \hat{e}_j [k - n\alpha_j]; \\ \hat{e}_j [k] = \hat{v}_j [k | N_y] + \hat{\alpha}_{j,1} \hat{v}_j [k - 1 | N_y] + \dots + \hat{\alpha}_{j,k-N_y-1} \hat{v}_j [N_y + 1 | N_y] + \\ \quad + \hat{\alpha}_{j,k-N_y} \hat{v}_j [N_y] + \dots + \hat{\alpha}_{j,n\alpha} \hat{v}_j [k - n\alpha]. \end{array} \right. \tag{13}$$

By definition, on the measuring horizon $\overline{1, N_y}$, predicted outputs are identical to measured outputs. The performance of all predictors (including PARMA) can be assessed by means of *prediction quality (PQ)* criterion, defined in [8].

The PSO technique was employed for all predictors mentioned above [1], [2]. Obviously, PQ criterion stands for the cost function F , which has to be maximized. Since it depends on the structural indices of the models, the particle has to be defined according to each predictor. We exemplify next, how PSO was applied on ARMA predictor and then, we illustrate the *PSO-ARMA* algorithm.

Thus, the particle is described by $\overset{def}{\mathbf{x}} = [p \quad na \quad nc \quad n\alpha]^T$, referring to the degrees of polynomials in trend component, ARMA model and AR approximating model, respectively. Consequently, the current population, the elite population and the anti-elite one, each of dimension P , are represented as 4 or 3 rows and P columns matrices, like in figure 4. Each column is a combination of structural indices necessary to identify the prediction model ARMA.

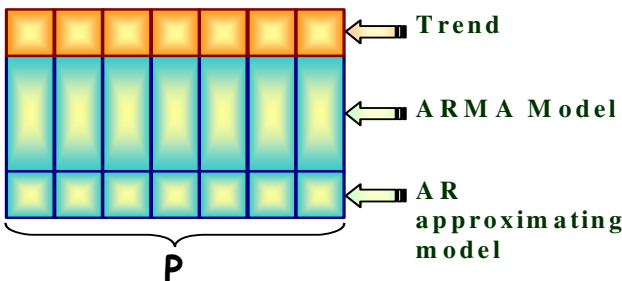


Fig. 4. The matrix representation of particle population for ARMA model

The steps of PSO-PARMA algorithm are summarized next.

⇒ **Input data:**

- the data block containing y with finite length, N_y ;
- the prediction horizon length K (with $K \ll N_y$); usually, $K \leq 7$;
- the configuration parameters:
 - maximum value of polynomial trend degree, $P_{\max} \in \mathbb{N}$ (by default, $P_{\max} = 10$);
 - the maximum values of ARMA model structural indices, $NA, NC \in \mathbb{N}$ (by default, $NA = NC = \lfloor N_y / 3 \rfloor$);
 - the particles number of populations in PSO, $P \in \mathbb{N}$ (by default, $P = 50$).

1. Center data to average and define the generic particle, according to figure 4, i.e.

$$\mathbf{x} = [p \quad na \quad nc \quad n\alpha]^{T, \text{def}}$$

2. Preserve the last $\lfloor K/2 \rfloor$ samples (but at least 3), for model training (i.e. PQ evaluation, for each particle in current population).
3. Create the initial population with P uniformly distributed particles on the searching space set by the maximum values of the structural indices and pass to the next generation by using equations (1)–(2) (where the mobility factor, the cognitive and social variances are set at random, and the initial speed is null).
4. Create the elite and anti-elite populations (by using PQ).
5. While the stop test of PSO is not achieved :
 - 5.1. Compute the mobility of each particle in the current generation.
 - 5.2. Compute the relative cognitive and social variances for each particle in the current generation.
 - 5.3. Generate the transition frequencies at random (uniformly).
 - 5.4. Update the speeds and positions of particles in current generation.
 - 5.5. Update the elite and anti-elite populations.
 - 5.6. Asses the variances of the current and elite populations.
 - 5.7. If the product of the two variances significantly increases, perform crossover between current and elite populations.
 - 5.8. If the product of the two variances significantly decreases, perform crossover between current and anti-elite populations.
 - 5.9. If crossover occurred, keep the particles producing the highest values of PQ (including the most valuable particle of elite). Then update the elite and anti-elite populations.
6. Select the optimal particle of elite (which produces the highest value of PQ). This is uniquely associated to a polynomial trend y_{T, p_0} with degree p_0 , to a seasonal component y_{S, p_0} (eventually null) and to a triplet of structural indices $\{na_0, nc_0, n\alpha_0\}$.

7. Update the deterministic component on the measure horizon (including the preserved samples) and compute the colored noise: $v \equiv y - y_{T,p_0} - y_{S,p_0}$. Then, estimate the ARMA model of the colored noise v , for the triplet of structural indices $\{na_0, nc_0, n\alpha_0\}$ (by using the whole data set).
8. Perform extrapolation and prediction of ts, by using the model (9) and the recursive equations of ARMA predictor (13):

$$\hat{y}_{T,S,ARMA} [N_y + k | N_y] = \hat{y}_T [N_y + k] + \hat{y}_S [N_y + k] + \hat{y}_{ARMA} [N_y + k | N_y],$$

$$k \in \overline{1, K}. \quad (14)$$

9. Compute the prediction error variances $\{\sigma_k^2\}_{k=1, \overline{K}}$ (see [6]).

☞ *Output Data:*

- the set of predicted values: $\{\hat{y}_{T,S,(AR)(MA)} [N_y + k | N_y]\}_{k=1, \overline{K}}$;
- the prediction error variances $\{\sigma_k^2\}_{k=1, \overline{K}}$.

5 Simulation Results

The parameter of interest for automatic control is the soil moisture (Mo). Nevertheless, Mo is correlated with the other ecological parameters. Therefore, our simulations are concerned with all parameters of figure 3. However, the variations in figure 5, are depicted for Mo only, together with their best detected trends. Simulations have proven that this parameter is one of the most difficult to predict.

A collection of 13 data blocks has been employed to predict the 31 parameters by means of the four predictors: PARMA, PARMAX, KARMA and FORWAVER. The data blocks resulted from combinations of soil or ambient parameters, as shown by the synoptic map of figure 3. However, the SoRa parameter has been assigned either to soil or ambient data blocks. The PARMAX predictor was the most employed, since it has to be run several times for each channel. In order to reduce the simulation time, the EcoMonFor-F computer network was extended up to 16 PCs, including the laptop of EcoMonFor-M. The ecological phenomena usually act slowly. Therefore it is suitable to predict values every 3-4 hours. The simulation time for predictors varied between several minutes and several tens of hours, depending on their complexity, the number of analyzed ecological data and the modeling of stochastic component. However, the intelligent search for optimum, performed by PSO technique, led to important gains in terms of computational time, for all predictors.

When using PARMA or FORWAVER predictors, no correlations between Mo and other parameters are considered. They are therefore the fastest predictors. On the contrary, with PARMAX and KARMA, Mo was predicted when considering correlations with soil temperature (Te) and soil water content (WaCo) or leaf wetness (LeWe), as the chart of figure 3 already pointed. Usually, for them, the

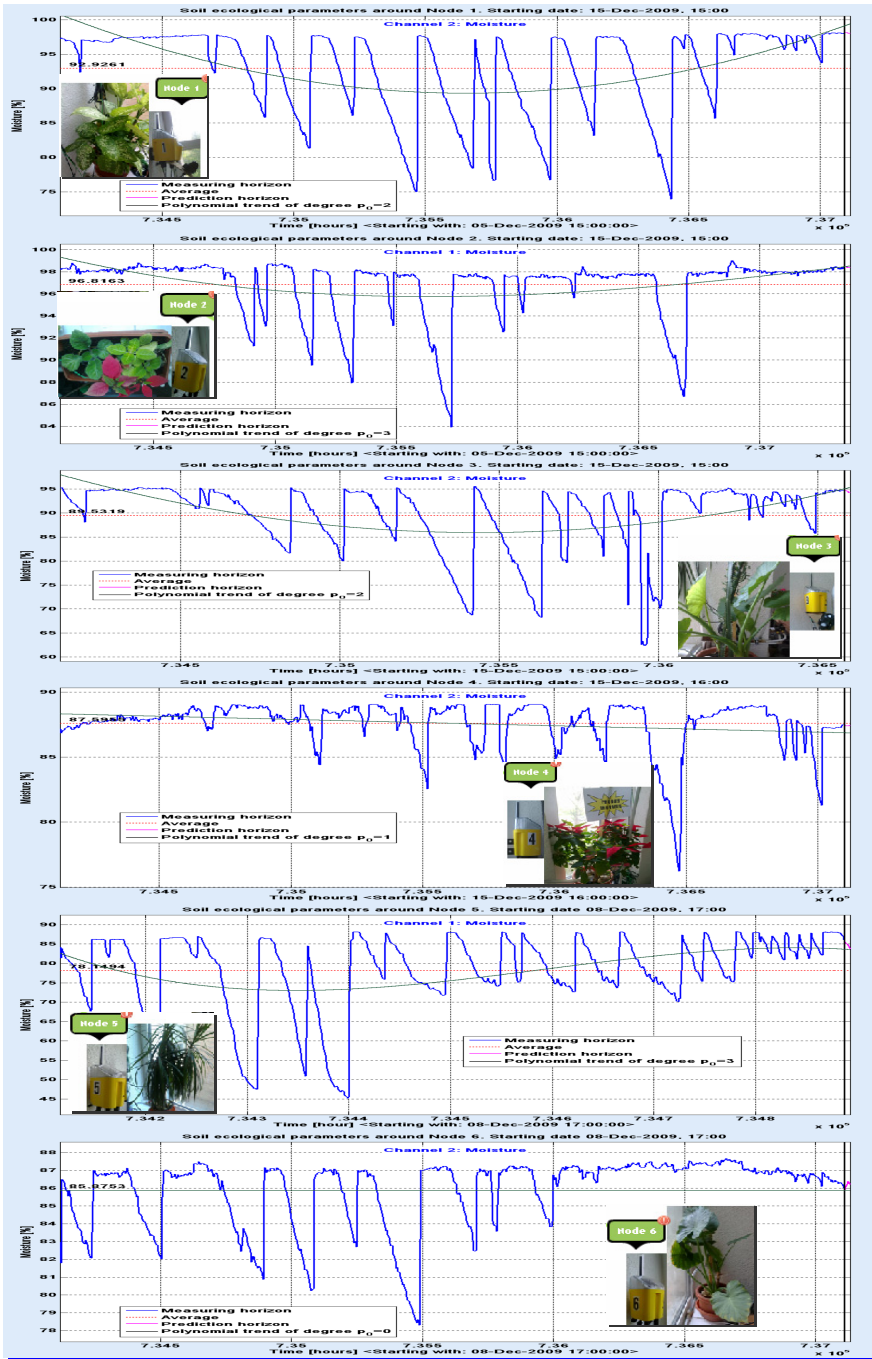


Fig. 5. Soil moisture variations within the greenhouse

Table 2. Prediction Quality (PQ[%]) for all monitored ecological parameters

Node	Parameter	PARMA	PARMAX	KARMA	FORWAVER
1	WaCo	71.14	77.72	72.47	78.17
	Mo	63.39	56.48	35.74	69.12
	Te (soil)	65.65	77.41	47.3	81.23
	Hu	78.72	81.33	79.59	82.6
	Te (ambient)	76.6	74.85	57.52	78.11
	DwPo	75.59	76.84	64.42	76.66
	SoRa	74.83	74.01	73.33	78.88
2	Mo	64.77	71.58	38.7	76.07
	Te_s	78.18	77.49	57.46	79.34
	LeWe	77.95	78.94	6650	80.43
3	WaCo	69.73	77.79	78.36	71.58
	Mo	62.78	70.68	78.47	65.71
	Te_s	77.74	77.41	64.45	78.38
	Hu	77.17	82.47	55.59	81.64
	Te_a	77.55	78.54	66.83	78.24
	DwPo	76.33	79.22	76.81	76.58
	SoRa	74.83	73.48	12.97	74.2
4	WaCo	70.26	74.64	60.84	82.63
	Mo	60.69	69.2	50.71	63.39
	Te_s	77.8	78.55	60.18	79.99
	LeWe	77.62	78.58	38.8	78.91
5	Mo	74.95	83.58	47.04	76.22
	Te_s	74.1	81.26	55.21	79.69
	LeWe	64.26	64.53	27.48	67.61
	Hu	72.09	80.24	41.67	72.86
	Te_a	79.46	82.71	61.3	79.7
	DwPo	66.89	72.08	50.88	67.28
6	WaCo	77.34	78.44	60.23	77.42
	Mo	73.34	76.89	52.05	73.88
	Te_s	81.81	84.83	62.62	82.44
	LeWe	24.77	21.87	22.37	25.2

running time is bigger. Table 2 shows the PQ values (in percents) obtained for all monitored parameters, after running the predictors. The higher PQ, the better. The best PQ values for Mo are emphasized in the table. As a general result, PARMA is never the best, but the fastest. However, its performance is fair, with a good trade-off between speed and accuracy, which allows assigning this predictor the bronze medal. For the silver medal, FORWAVER is the righteous selection. (This predictor is based on orthogonal wavelets.)

The gold medalist is PARMAX, with 3 best predicted Mo values out of 6. Correlations between Mo and the other parameters helped the predictor to provide the best results. It must be noted however that, in spite of PSO integration, PARMAX is by far the slowest predictor. The big unexpected deception is KARMA, which is based on state space representation. Its reduced performance is probably due to the limitations of N4SID identification algorithms. The N4SID procedures are over-sensitive to the variation of internal states number. Just removing or adding one single state can dramatically modify the predicted values outside as well as inside the measure horizon.) The bronze-silver-gold classification is confirmed by prediction performance of the other greenhouse parameters as well (although, sometimes FORWAVER is better than PARMAX), revealed by table 2.

6 Conclusion

This article approached the problem of monitoring and intelligent forecasting of small greenhouse parameters. The monitoring system (namely, EcoMonFor) integrates three user friendly interfaces eko-View, eko-Greenhouse and eko-Forecast, which are implemented on a mobile or fixed web computer. An intelligent method is proposed for ecological signal prediction, in order to achieve the best prediction performances. It is based on the adapted and improved evolutionary algorithm PSO. A comparative study between four predictors performance (PARMA, PARMAX, KARMA, FORWAVER) revealed that the best accuracy of prediction is achieved by PARMAX; next come FORWAVER and PARMA, depending on the correlation degree between the monitored ecological parameters. Nevertheless, PARMA is often preferred for its prediction superior speed, combined with only a slight decrease in terms of prediction quality.

References

- [1] Stefanoiu, D., Culita, J.: Multivariable Prediction of Physical Data. PUB Scientific Bulletin, A Series 72(1), 95–102 (2010)
- [2] Stefanoiu, D., Culita, J., Ionescu, F.: FORWAVER – A Wavelet Based Predictor for Non Stationary Signals. In: Industrial Simulation Conference ISC, Lyon, France, pp. 377–381 (2008)
- [3] Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Piscataway – N.J., U.S.A, pp. 1942–1948 (1995)

- [4] Dumitrascu, A.: Contributions to industrial computer networks in process control, PhD thesis, “Politehnica” University of Bucharest, 218 pages (2010)
- [5] Culita, J., Ștefănoiu, D.: FORTIS – An integrated Simulator for Distributed Time Series Forecasting. In: Industrial Simulation Conference ISC, pp. 27–33, Budapest, Hungary, June 7-9 (2010)
- [6] Ștefănoiu, D., Culita, J., Tudor, F.: Experimental Approaches in Process and Phenomena Identification. AGIR Press, Bucharest (to appear, 2012)
- [7] Söderström, T., Stoica, P.: System Identification. Prentice Hall, London (1989)
- [8] Culita, J., Ștefănoiu, D., Dumitrascu, A.: EcoMonFor – A System for Greenhouses Monitoring and Forecasting. In: Industrial Simulation Conference, Venice, Italy, June 6-8, pp. 262–269 (2011)

Urban Traffic Monitoring and Control as a Cyber-Physical System Approach

Simona Iuliana Caramihai and Ioan Dumitrache

Automatic Control and Computer Science Faculty, University “Politehnica” of Bucharest,
313 Splaiul Independentei, Bucharest, 060042, Romania
dumitrache@ics.pub.ro, simona.caramihai@aii.pub.ro

Abstract. The paper presents an integrated system for Urban Traffic Monitoring and control conceived in the perspective of a Cyber Physical Approach. Traffic “actors” are defined as classes of agents, each of them with a set of possible behaviors and behavioral drivers. Some of the behaviors are dependent on the actual physical structure/ implementation of agents. Different levels of control can be defined with regard to the physical functioning context and the overall system will have a complex emergent behavior. At this moment of conception, traffic actors include mainly components of traffic infrastructure and respective control and monitoring units, but the design is open-system, so as to define also “vehicle” class.

1 Introduction

In a world where the number of cars is increasing at far more important rate than that of the infrastructure development, traffic issues are to be treated with great responsibility.

There are many problems to be addressed, from the traffic participants’ security, to the traffic management and monitoring, minimization of fuel consumption, pollution reduction, routing optimization, vehicle speed adaptation to traffic context, a.s.o. There are also local solutions to many of these particular problems, but a Global Traffic Monitoring and Control System, able to solve all of them for a large urban network is still not yet in use. Moreover, the socio-economical perspectives of development in large urban areas will raise new problems, which will need solutions implying vehicle-to-vehicle communication for enhanced safety and convenience (“zero-fatality” areas), drive-by wire, next generation autonomous vehicles and others.

These challenges could only be solved by taking a new approach in the design of large, open, complex, dynamical and flexible control and monitoring structures.

Literature provides a series of works on modeling [1] and [2], simulation, traffic management, optimization and control [3], as well as traffic forecasting.

Given the complexity and the particularities of urban traffic control and optimization problems, research in the area has turned towards intelligent

technique approaches, such as rule-based systems for predefined plan selection, fuzzy rules control systems, neural networks, or neuro-fuzzy systems, cellular automata modeling and evolutionary algorithms optimization [4]

As the complex problem definition for urban traffic control should take into account not only the mathematical and modeling aspects, but also implementation issues as well as feedback and validation capabilities, which are linked to physical devices incorporated both in the traffic control and measurement infrastructure, but prospectively also in traffic actors, as vehicles, the most promising solving approach seems to be a modular, agent-oriented one. But the interaction of agents in such a system is at least as important as their functionality, leading to complex emergent behaviors whose a priori analysis is not possible. Inherent hardware and software heterogeneity add new dimensions to the problem.

A new and emerging approach that takes explicitly into consideration aspects as these mentioned above is that of Cyber-Physical Systems (CPS). Under this acronym are denominated “smart systems that have cyber-technologies, both hardware and software, deeply embedded in and interacting with physical components, sensing and changing the state of the real world” [5]. Their behavior is highly dynamical and emerges from the interaction of the behaviors of all the components, being subject to abrupt commutations as well as smooth evolution between different functioning regimes, in answer to environmental changes.

Traffic control is one of the main fields declared as special challenges for CPS-oriented solving [6]. Some of the reasons are including:

- The difference between the dynamics of infrastructure development and the increasing number of vehicles
- The socio-economical importance of transport system for the mobility of workforce as well as for the urban well-being
- The necessity to integrate in traffic control systems both modern information, communication and control approaches and already-existent local control devices
- The high variance and dynamics of the controlled traffic system
- The necessity of rapid functional reconfiguration of the system

This paper will present the work performed in the framework of a Romanian National Funded Research Project (“Intelligent techniques for modeling, analysis and optimization of urban traffic”), concerning the urban traffic control, with a special emphasis on the city of Bucharest.

The project team included the company in charge with the implementation of a modern traffic control system in Bucharest and some of the specifications of the control platform have taken into account the work already performed by them, namely the capabilities of the junction control units, sensors type and position and communication capabilities of the on-site network (already installed when the project started).

The research work was centered on the objective of eliminating traffic congestion by adapting the duration of green/ red lights to the traffic regime.

For this purpose a software platform was conceived allowing both:

- *Off-line functionalities* as designing an urban traffic network based on infrastructure elements (junction patterns, street capacities, street topology a.s.o.), testing the effectiveness of a given activation profile for specified traffic flows, testing different control approaches at local level, generating new activation profiles as functioning scenarios, based on traffic flows, identifying functioning patterns based on measured car flows a.s.o.)
- *On-line functionalities* as monitoring and control

Basic specification for the structure of such a platform included modularity, open-system architecture, event-driven asynchronous and concurrent functioning, and local control autonomy.

The main formal support was chosen in order to take into account the hybrid structure of the traffic: the continuous part represented by the car flow and the discrete part given by the light generation.

It is intended to allow the preliminary test of the control concept in order to prepare the Cyber-Physical oriented design of the complete control structure.

A detailed description of the agent based design follows in section 2, while section 3 presents a control architecture currently under test. Section 4 describes the testing software platform, in both structure and functionality. A module that performs a genetic algorithms (GA) solution search has been implemented and there are two other modules under construction – one implementing fuzzy control and another implementing a neural networks approach. Section 5 will present details of the GA approach and results obtained on a case study. The final section will present conclusions and future work.

2 Agent-Based Design for Urban Traffic Light Control

Urban traffic is characterized as a large-scale, complex, non-linear, time-varying system for which modeling, simulation and control are difficult tasks.

In this context and also due to its geographically distributed nature, the domain of traffic and transportation management is well suited to distributed architectures and multi-agent based approaches. The agent-based approach is justified both by the difficulty of globally modeling a traffic network and by the necessity of a more decentralized control approach with smaller control centers that have to manage only parts of the global structure.

In designing the agent implementation we have adopted a natural decomposition of a traffic network into *active* elements, such as crossroads, that have a definite role in regulating the car flow, and *passive* elements, such as road segments which are only physical parameters of the system. Modular models of crossroads [7] have been encapsulated into collaborative software agents that act as managing active elements within the control architecture. This software entity has been designed to implement behaviors that can simulate, monitor and control a crossroad and will henceforth be called *junction-agent*.

The general architecture of a junction agent is presented in Fig. 1. The World Model includes the model of the crossroad being monitored as well as other traffic state specific information and control parameters. The Value Judgment and Behavior Generation enable the agent to assess the state of traffic in its crossroad, derive appropriate actions and test their outcome on the crossroad model stored in the World Model.

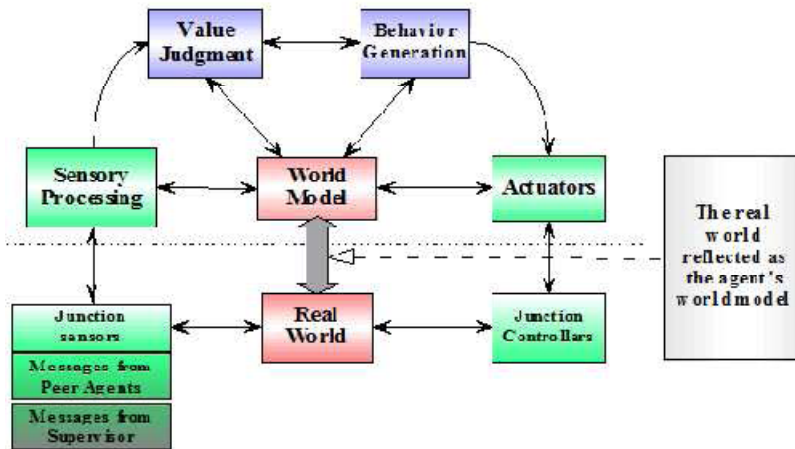


Fig. 1. The general architecture of a junction agent

The functioning of a junction agent is asynchronous and event based. It consists of a composite behavior running two parallel sub-behaviors (see Fig. 2): the basic behavior for managing the crossroad and a listener behavior for perceiving the environment and communication. A formal description of the agent model and a detailed review of the event list that the agent can process can be found in [8].

Adjusting the timing of the traffic light phases is the primary means of action of the junction agent. Light phase information is stored in the world model in the form of an activation profile which captures the succession of light phases as well as their durations.

If desired, the user can create new agent structures by designing behaviors with extended capabilities which can be easily integrated with the platform by merely instantiating the appropriate classes of agents and agent behaviors.

3 Control Architecture

The decentralized modeling approach presented in Section 2 transforms the congestion prevention into a green light timings optimization problem, decomposable in multiple sub-problems which can be solved locally.

Despite the fact that an optimal control solution cannot be applied in the case of a complex system such as urban traffic, the bottom-up approach of local congestion prevention mechanisms shows surprisingly good results [3].

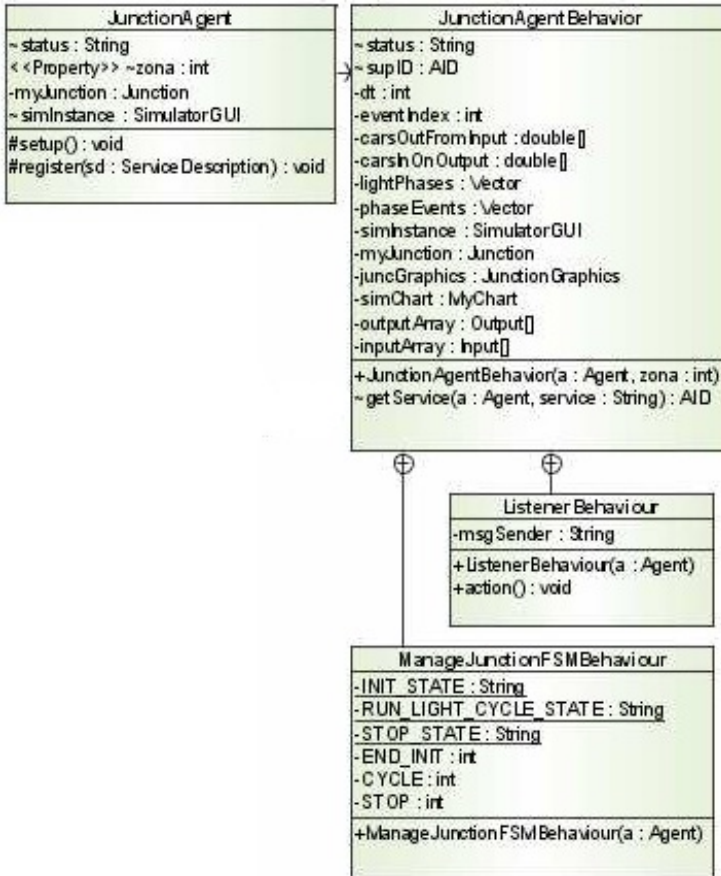


Fig. 2. Behavioural model of a junction agent

Building upon this idea, we propose a hierarchical agent-based architecture of control where bottom layer agents have the capability to monitor and apply local mechanisms of control in order to minimize the waiting queues of vehicles on the crossroad’s inputs. The overall structure of the proposed traffic management and control architecture is presented in Fig. 3.

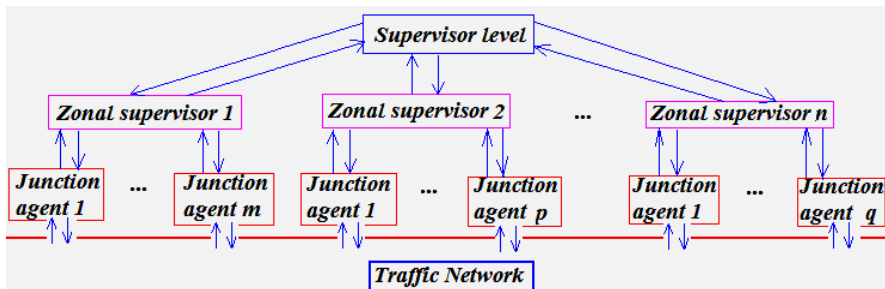


Fig. 3. Hierarchical agent-based control architecture

The bottom level of control consists of junction-agents, each of them being able to control the duration and succession of traffic lights for a given crossroad. Their basic model and functionality has been described in the previous section. Junction-agents are capable of both horizontal and vertical communication, exchanging traffic load information with peer agents and reporting to a zonal supervisor agent from which they receive updated control parameters.

Zonal supervisors on the middle layer are agents assigned to a set of junction agents in a given sub-region, implementing strategies to serve two main objectives:

- regional statistic estimation and prediction planning for zonal optimization;
- conflict resolution if mutual agreement on green light timings fails between any two junction agents on the bottom layer.

The top layer consists of a higher level supervisory agent which coordinates zone supervisors, basically establishing *scenario-based functioning regimes* for critical contexts that cannot be solved otherwise.

The architecture is conceived so as to allow the bottom layer to ensure a basic functioning even in situations in which communication links between agents are broken, either horizontally or vertically, by the basic contextual scenarios (activation profiles) that are stored in the world model of junction agents.

In terms of control, the bottom layer agents work in the timeframe of the real traffic environment, therefore their congestion prevention mechanisms must be based on fast response, direct methods derived from current measurements only. In this context, fuzzy logic is a plausible choice for processing a nondeterministic and non-linear problem, such as that of adjusting green light timings according to traffic load at any given moment. Also, traffic control can be easily expressed by rules, which makes fuzzy rule-based signal control a natural choice. Membership functions are defined for crossroad lane queue lengths using fuzzy terms such as “long-queue”, “short-queue”. More green time is allocated for a lane if there are many vehicles arriving in, and less green time otherwise. The fuzzy local congestion prevention mechanism provides a reasonable alternative when traffic load is not critical, or if the bottom layer becomes disconnected from superior command in the event of whatever system or communication failure.

The zonal supervisor works on a larger timescale, implementing a genetic algorithm (GA) for the search of an optimal tuple of green light timings for the crossroads in a sub-region. The degree of optimality of a solution is given by the waiting queue lengths for all. Further details on the GA implementation are given in section 5.

Prediction and planning is done by using real traffic data received from the junction agents and stored as traffic history over a period of four months (data used was from October 1st 2009 to January 31st of 2010). These data have been preprocessed in order to be used as model inputs for simulating over one hour in the future to obtain a forecast of the network behavior. Preprocessing implies various statistic operations on the data in order to derive the most relevant inputs according to the time of day and day of the week. These data manipulations ensure that the simulations take into account traffic load variations such as daytime vs. nighttime traffic, work shifts: morning time vs. evening, working days vs. weekends and holidays.

The scenarios stored by the top level supervisor are generally conceived offline, using data stored over time from the bottom and medium levels. A coarse classification of traffic scenarios can be made based on traffic load:

- extremely rarefied traffic – generally night and suburbs traffic: the traffic light cycle can be overridden and green light can be given for any direction where a vehicle is approaching, based on a *first arrived - first served* strategy.
- medium density traffic – generally not leading to congestion – weekend/official holydays traffic.
- dense traffic – possibly leading to congestions.
- saturated traffic – congestion beginning to affect the neighboring junctions.

4 The Software Platform

A software platform was implemented primarily as a flexible, open-system environment for testing various intelligent approaches for agent-based traffic control. The block structure of the software platform is presented in Fig. 4.

The software platform was implemented in Java, using agent programming concepts combined with classical object programming. The multi-agent system was created using the JADE infrastructure which offers specific facilities for agent creation and agent interaction. The concepts of *Junction*, *Input*, *Output*, *LightPhase*, etc. were modeled as OOP classes, which are integrated within the agent's world model.

The platform allows traffic network design based on a predefined agent library (see Fig. 5). The user can interconnect crossroads in the traffic network by using the graphical user interface and specify values for parameters that define each connection (such as link capacity, maximum load accepted, etc.) The user actions in the GUI are translated into agent connections and attribute initialization, thus reflecting the physical structural information of the traffic network within the agent population.

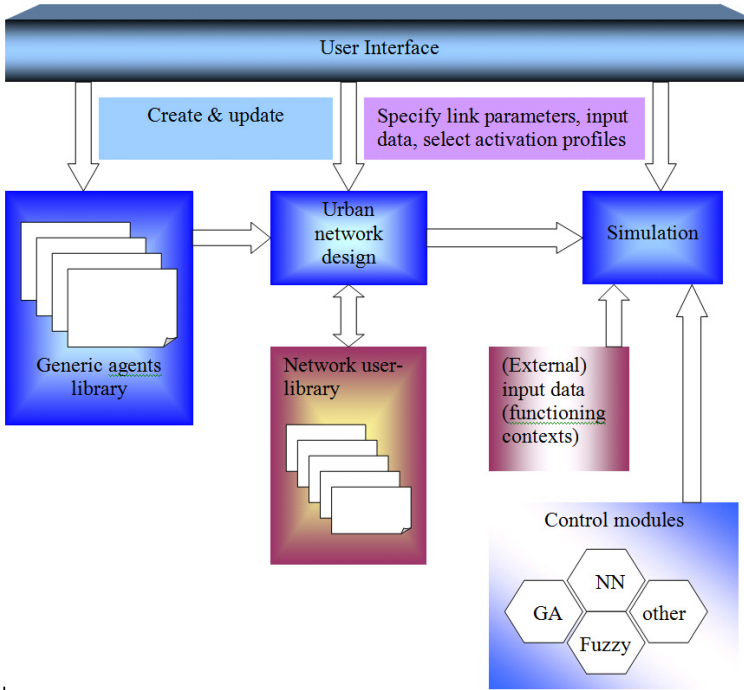


Fig. 4. Structure of the software platform.

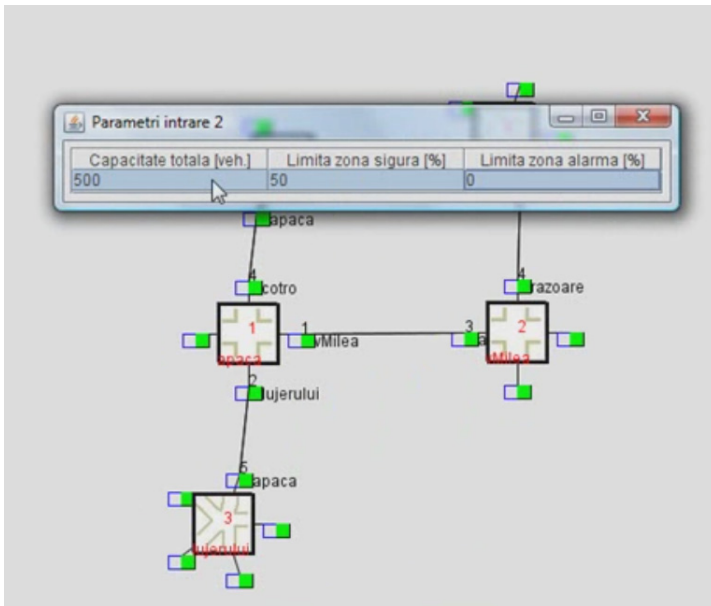


Fig. 5. Software platform screenshot: creating a traffic network and setting parameters

A traffic network functioning can then be simulated using input files of data. The user can see graphical representations of the evolution of the waiting queues on each crossroad input links (Fig. 6).

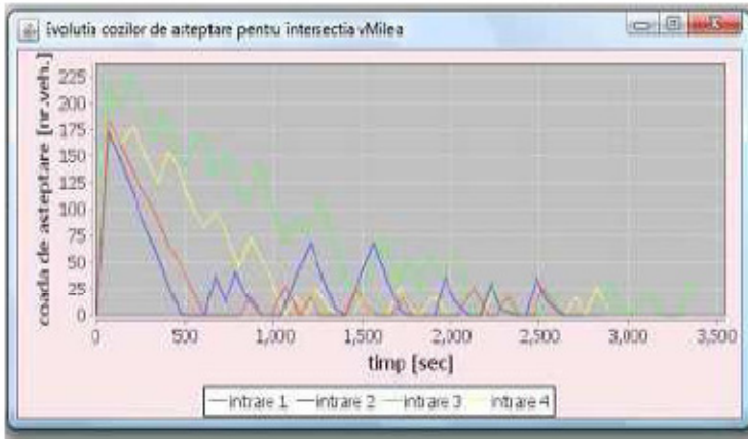


Fig. 6. Plot of vehicle queue length variation over time for all input links of a crossroad

At this point the supervisor agent performs GA optimization of the phase activation profiles only at user demand (by pressing a button on the graphical user interface).

5 Case Study

As a case study, we have chosen an urban area of three interconnected junctions located in the center of Bucharest, as depicted in Fig. 7. Under current traffic conditions and green light phase timings this small area is prone to developing congestive traffic states.

The optimization variables are the green phase timing for all (or a subset of) crossroads in the area of the supervisor agent. The total light phase duration is also an optimization variable, aiming towards a balanced traffic flow across the region.

The fitness function is evaluated to be inverse proportional to the weighted average of the waiting queue lengths on all road segments entering the crossroads.

The GA has been implemented in Matlab 2010a by the use of the Genetic Algorithms and Direct Search Toolbox. The GA was run using the following options:

- the default type double for chromosomes, within a certain lower bound and upper bound [lb, ub] derived from the time limits imposed on each phase;
- population size of 15 chromosomes;

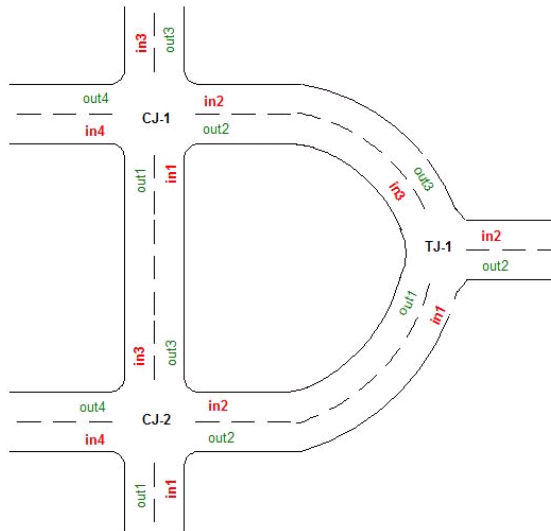


Fig. 7. Traffic network used in case study: two X crossroads and one T crossroad

- a maximum number of 15 generations as a stopping criterion or a maximum of 7 stall generations;
- stochastic uniform selection function;
- adaptive feasible mutation, which randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation

The GA algorithm Matlab code was run from the Java environment by using Matlab Server and passing model data in XML format. Also, assuming a multi-core cluster for the implementation of the supervisory level algorithms we have used the Parallel Processing Toolbox, running each junction model on a separate Matlab worker.

Simulation results show the variation of the waiting queues (in number of vehicles) over the time-span of an hour (see Fig. 8 a, b, and c). As can be seen, all input queues remain consistently below a certain threshold considered as a percentual limit of the total road segment capacity (marked by a red horizontal line in all Figures).

The computational and temporal requirements to reach a solution by genetic algorithms need also to be taken into consideration to verify the feasibility of the implementation in the real traffic environment.

The simulations presented above were done on a machine with an Intel Core2 Duo CPU at 2.6GHz. The total run-time was slightly below the time-span of the simulations with possible overhead due to other Windows processes running. Also, further code optimization may be possible.

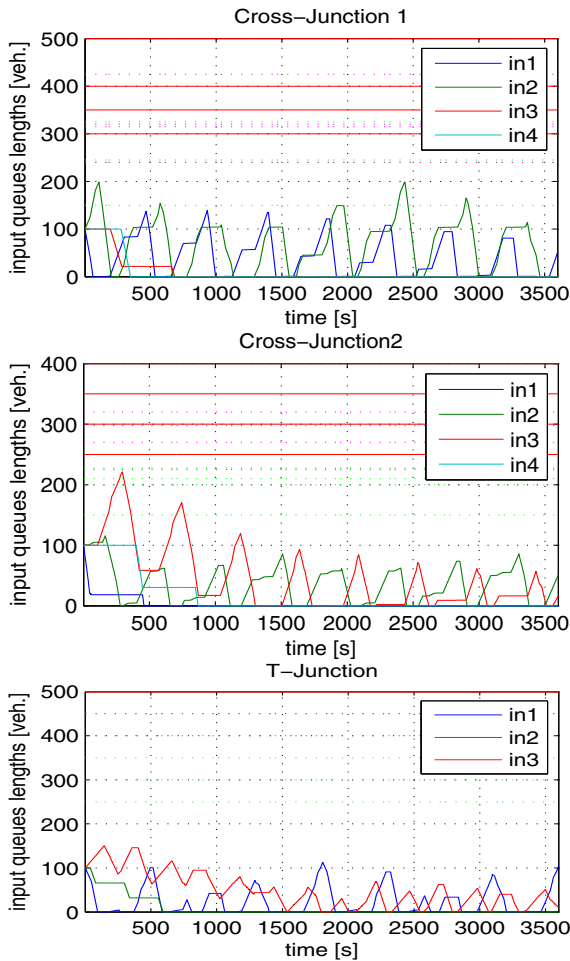


Fig. 8. Simulation results showing the time evolution of input links vehicle queues using the traffic light phase timings found by GA

6 Conclusions and Future Research

This paper presented an agent-based approach which allows a traffic region to be transposed into a community of agents. A software platform has been designed to test this approach using various types of agents. Our preliminary simulations confirm the adequacy of the multi-agent approach for the modeling and simulation complex, large scale systems. The implicit modularity obtained by the multi-agent implementation allows a functional decomposition leading to a reduction of complexity without loss of generality. The autonomous functioning of each unit combined with communication among working units results in a complex, emergent behavior of the overall system which could have hardly been modeled otherwise.

The test platform developed allows users to build their own test cases of traffic infrastructure to analyze, with or without extending the platform or designing new control strategies.

Time is a critical issue for the problem of congestion prevention, therefore we shall proceed in studying methods for reducing the runtime of the genetic algorithm in finding a solution. A possibility to achieve this objective is adopting a hybrid solution in conjunction with artificial neural networks (ANNs) [9]. Since the most time consuming element in running a genetic algorithm optimization is evaluating the fitness function at each step, we propose the offline training of a neural network model for the urban area under consideration which will then be used in evaluating the chromosomes produced by the GA. To ensure more accuracy for the ANN model of a traffic area a library of such models may be built to take into consideration the steep traffic load variations such as daytime vs. nighttime traffic, work shifts: morning time vs. evening, working days vs. weekends and holidays.

We also intend to use in the future the NCIT-Cluster, a high performance computing cluster infrastructure available at the Politehnica University of Bucharest to test the distributed implementation presented above.

Acknowledgments. The paper presents partial results of a CNMP budgeted project: "Intelligent Techniques for the Modeling, Analysis, and Optimization of Urban Traffic", contract no. 71108.

References

- [1] Daganzo, C., Geroliminis, N.: An analytical approximation for the macroscopic fundamental diagram of urban traffic. *Transportation Research Part B* 42, 771–781 (2008)
- [2] Hoogendoorn, S.P.: State-of-the-art of Vehicular Traffic Flow Mod-elling. Special Issue on Road Traffic Modelling and Control of the *Journal of Systems and Control Engineering* 215(4), 283–303 (2001)
- [3] Lämmer, S., Helbing, D.: Self-Stabilizing Decentralized Signal Control of Realistic, Saturated Network Traffic (2010) (working paper)
- [4] Sanchez Medina, J.J., Galan Moreno, M.J., Rubio Royo, E.: Evolutionary Computation Applied to Urban Traffic Optimization. *Advances in Evolutionary Algorithms*
- [5] http://www.nitrd.gov/subcommittee/hcss/documents/CPS_OSTP_ResponseWinningTheFuture.pdf
- [6] http://iccps2012.cse.wustl.edu/_doc/CPS_Summit_Report.pdf
- [7] Voinescu, M., Udrea, A., Caramihai, S.: On Urban Traffic Modelling and Control. *Journal of Control Engineering and Applied Informatics* 11(1)
- [8] Caramihai, S., Voinescu, M., Munteanu, C., Dumitrache, I.: An Agent-Based Approach for Urban Traffic Simulation and Control. In: *Proceedings of the 6th IFAC International Workshop on Knowledge and Technology Transfer in/to Developing countries: Automation and Infrastructure*, Ohrid, Macedonia (2009)
- [9] Issa, F.: Contributions to Optimizing Control Strategy Assignment in Collaborative Mobile Robots Applications. PhD. Thesis (2009)

Digital Motor Control Library

Radu Duma, Petru Dobra, Mirela Trusca, and Ioan Valentin Sita

Technical University of Cluj–Napoca, Automatic Control Department,
Memorandumului str. 28, 400114 Cluj-Napoca, Romania
{Radu.Duma, Petru.Dobra, Mirela.Trusca, Valentin.Sita}@
aut.utcluj.ro

Abstract. The paper present Rapid Control Prototyping (RCP) libraries for Matlab/Simulink, which are used to generate real-time C code for the Renesas M32C87 microcontroller. The libraries are integrated in the toolbox Target for Renesas M32C87. Simulink blocks are implemented for drivers and algorithms specific for the Renesas M32C87 microcontroller. The toolbox contains five libraries: I/O drivers, Digital Motor Control (DMC), Controller Area Network (CAN) and Serial Communication Interface (SCI). In this paper the DCM library is described in detail. A positioning application for a Computer Numerical Controlled (CNC) machine is implemented using the Renesas M32C87 microcontroller.

Keywords: rapid control prototyping, microcontroller, closed-loop control, CNC machine, PID control.

1 Introduction

Developing controllers for applications (electrical drive systems) means large expenditure, when performed with usual development methods. The workload comprises development of a mathematical model as well as algorithm design and implementation, off-line simulation, and optimization. The whole process has to be restarted on occurring errors or divergences, which makes the development process time consuming and costly [1].

Rapid Control Prototyping (RCP) is a way out of this situation, especially if the control algorithm is complex and a lot of iteration steps are necessary.

RCP requires two components: a Computer Aided Control System Design (CACSD) software and a dedicated hardware capable of running tasks in real-time (Fig. 1).

CACSD tools are extensively used to generate real-time code automatically. The graphical programming approach removes the need to write software by hand and allows the engineer to focus instead on improving functionality and performance. Complete system design is carried out within the simulation environment. With the great diversity of applications, a development environment must be flexible and provide exactly the functionality necessary for efficient problem solving.

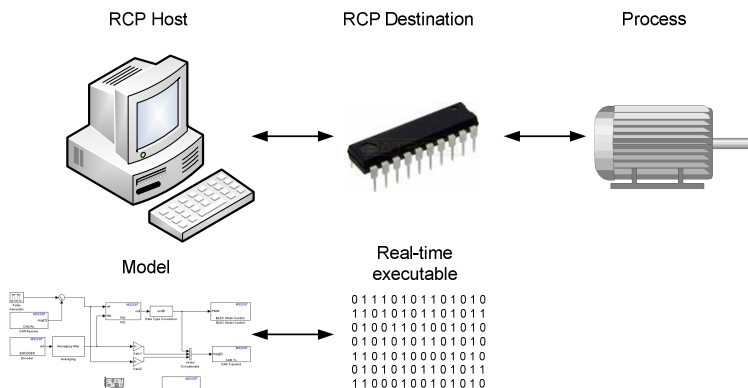


Fig. 1. General architecture of an RCP system

Mathworks' Simulink [2] software is such a graphical modelling tool. MathWork's developed toolboxes for some widely used targets: Motorola MPC555, Infineon C166, C2000 and C6000 DSP families from Texas Instruments. Rebeschies [3] developed the MICROS toolbox for standard 80C166 microcontrollers. A DSP based RCP system for engineering education and research using as CACSD tool Matlab/Simulink is presented in [4].

Hanselmann [5] from the dSPACE GmbH presented 'Total Development Environment' (TDE) for rapid control prototyping. TDE includes MATLAB, Simulink, powerful hardware, based on the DSP's, and an additional set of software tools for online data visualization (COCKPIT, TRACE). Controller boards like DS1104 and DS1103 are appropriate for motion control and are fully programmable from the Simulink environment.

Microchip [6] developed a Matlab RCP toolbox for the dsPIC33 Digital Signal Controllers (DSC), while Kerhuel [7] developed a toolbox which offers support for several Microchip microcontrollers and DSCs. Embedded RCP libraries with application for positioning system are presented in [8], while a low cost embedded solution for power quality measurement is presented in [9].

A free, open source, RCP solution is based on Scilab/Scicos [10] and Linux-RTAI [11], which uses the processor of a general purpose computer for executing real-time tasks. A real-time patch is applied to the standard Linux kernel. A modified version of the Scicos code generator, RTAI-Lab [11] generates hard real-time code compatible with Linux-RTAI. Acquisition cards can be directly integrated into the Scicos scheme and into the generated code using drivers provided by the COMEDI [12] project.

Ravn [13] implemented an adaptive control toolbox, for Scilab/Scicos, which can be used for RCP. A target supported in Scilab/Scicos is the Microchip dsPIC(R) DSC micro-controller [14]. Duma [15] presented a system identification and control RCP toolbox for Scilab/Scicos.

In Fig. 2 is presented the block diagram for the information flow (from concept to model) and for the data flow (between PC and the Renesas M32C87 microcontroller), for the implemented toolbox.

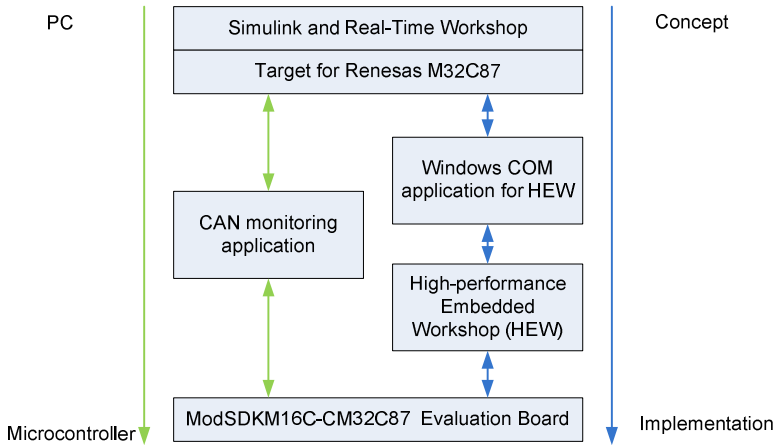


Fig. 2. Block diagram for the information and data flows for the *Target for Renesas M32C87* RCP toolbox

Control algorithms for the M32C87 microcontroller can be developed using blocks from the Target for Renesas M32C87 toolbox and predefined blocks from Simulink or user defined blocks.

After the validation of the model code can be generated using Real-Time Workshop. At the end of the code generation process the High-performance Embedded Workshop (HEW) Integrated Development Environment (IDE) is automatically started, a new HEW project is created, the generated code-source files are added to the project, the project is compiled and the binary file is downloaded to the target processor. The execution of the code can be observed in real-time using blocks from the CAN library.

2 Target for Renesas M32C87

The target board for the implemented toolbox is the modular development kit ModSDKM16C-CM32C87 (Fig. 3). It is composed of the generic main board ModSDK-Base and the CM32C87 CPU module.

The CPU module is equipped with a M32C87 processor member of the M16C microcontroller’s family. The processor is an enhanced version of the M16C kernel with an instruction set extended to 32 bits. The maximum operating frequency is 32MHz.

The development board is suited for digital motor application due to the following features: three-phase inverter motor control unit, output compare unit which can generate different PWM waveforms, 2-phase encoder input and time measurement (input capture) function. Other peripherals of the processor are: 10-bit A/D Converter: 34 channels; 8-bit D/A Converter: 2 channels; intelligent I/O.

The processor has several communication interfaces: 6 serial interfaces for synchronous and asynchronous communication, I²C, GCI and IrDa; 2 CAN channels.

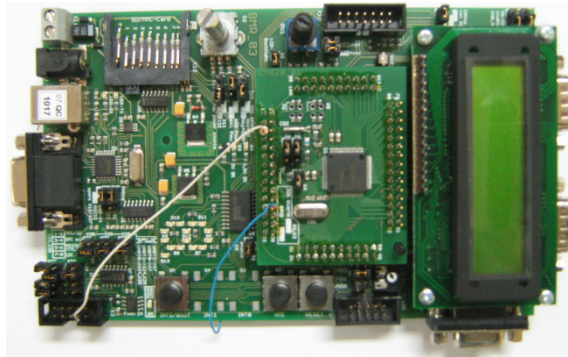


Fig. 3. The ModSDKM16C-CM32C87 development board

The Renesas microcontrollers are supported in several IDEs: HEW, IAR, TASKING. Each of this IDE has its own integrate compiler for the M16C family. The executables can be downloaded on the processor using the E8 or E8a debuggers or the Flash Development Toolkit and M16C Flasher software.

The features presented in this section make the ModSDKM16C-CM32C87 development kit an optimal solution for an RCP toolbox.

The Simulink library for the M32C87 processor is presented in Fig. 4. The toolbox contains other five libraries: *I/O drivers*, *Digital Motor Control (DMC)*, *Controller Area Network (CAN)* and *Serial Communication Interface (SCI)*.

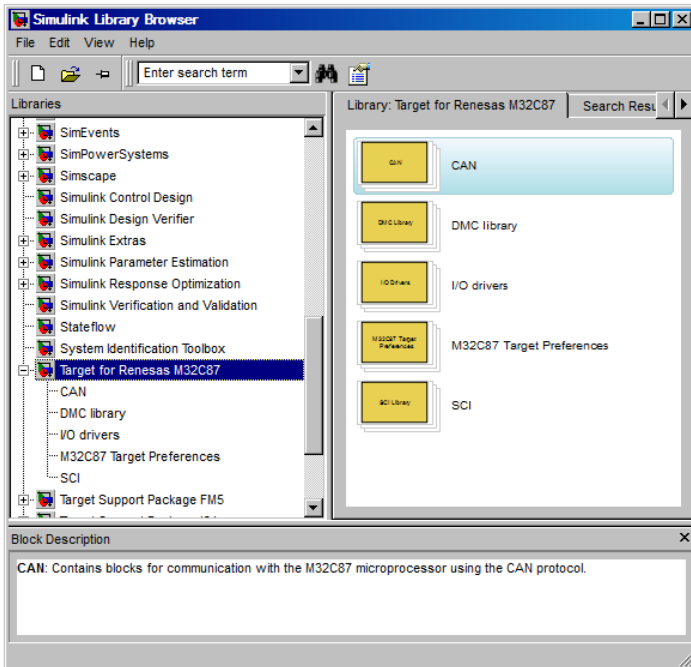


Fig. 4. Target for Renesas M32C87 RCP toolbox

3 Digital Motor Control Library

The *DMC* library (Fig. 5) contains blocks which implement digital motor control algorithms.

The implemented algorithms are optimized for the M32C87 microcontroller, which is a fixed point processor. The hardware multiplier is used and only integer variables are declared, in order to obtain the lowest sampling rate without breaking real-time constraints.

The library contains the following blocks:

- *M32C87 PID* – implements a PID controller with anti-windup. The structure of the controller is described in detail in subsection 1 of this section;
- *M32C87 Three-Phase Motor Control* – configures the M32C87 processor to generate six complementary PWM signals with dead-time. These PWM signals can be used for the control of three phase motors;
- *M32C87 BLDC Motor Control* – implements a trapezoidal six step control algorithm for a Brushless Direct Current (BLDC) motor. The behavior of the block is described in subsection 2 of this section.

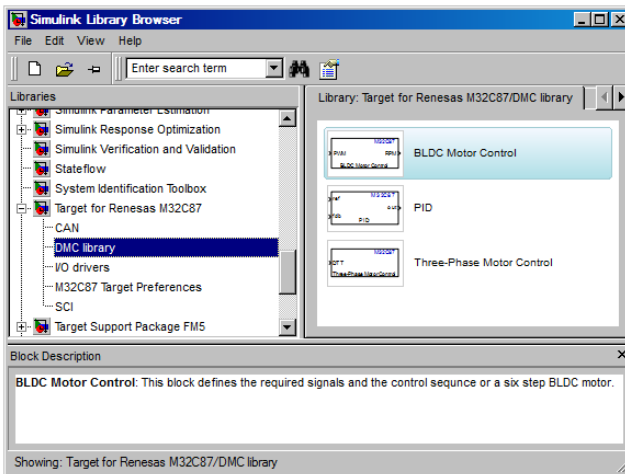


Fig. 5. DMC RCP library

3.1 M32C87 PID Block

More than 90% of the control loops are of PI and PID type [16, 17]. The reason for which the PID controller is so widely used is its simple structure which has proved to be robust with regard to many commonly control problems.

The standard equation of a PID controller is presented below:

$$u(t) = K_p \left(\varepsilon(t) + \frac{1}{T_i} \int_0^t \varepsilon(\tau) d\tau + T_d \frac{d\varepsilon(t)}{dt} \right)$$

where $\varepsilon(t) = y_{ref}(t) - y(t)$ is the error signal, y_{ref} is the set point and y is the output of the process.

In time several modifications have been done to the standard PID controller structure. A pure derivative gives a very large amplification of measurement noise. The gain of the derivative must thus be limited, by low-pass filtering the derivative term, resulting in a limited gain N at high frequencies. A fraction β of the reference signal acts on the proportional part, while the derivative component acts only on the output of the process.

The structure of the PID controller with integrator anti-windup is shown in Fig. 6.

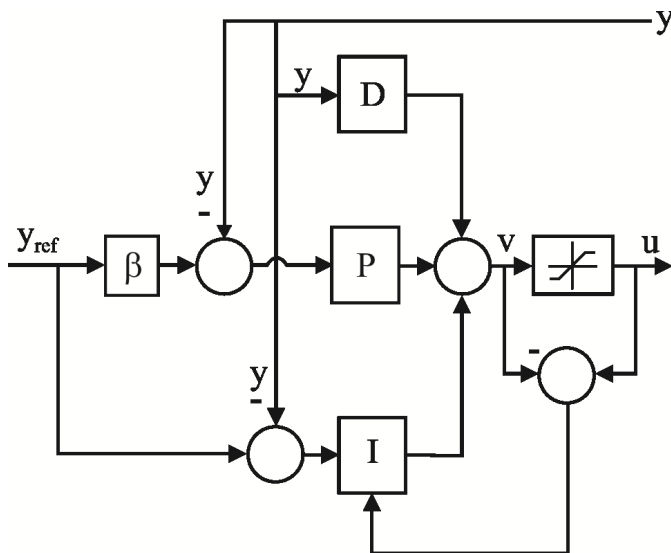


Fig. 6. The structure of the PID controller with anti-windup

The discrete equations used for the implementation of the PID algorithm are presented below:

$$\begin{aligned}
 P(n) &= K_p (\beta y_{ref}(n) - y(n)) \\
 D(n) &= \alpha_d D(n-1) + \beta_d (y(n-1) - y(n)) \\
 v(n) &= P(n) + I(n) + D(n) \\
 I(n+1) &= I(n) + \beta_i (y_{ref}(n) - y(n)) + \beta_i (u(n) - v(n))
 \end{aligned} \tag{1}$$

where $\alpha_d = T_d / (T_d + Nh)$, $\beta_d = (K_p T_d N) / (T_d + Nh)$, $\beta_i = K_p T_s / T_i$ and $\beta_t = T_s / T_t$.

Equations (1) are used for the target language compiler implementation of the *M32C87 PID* block.

In Fig. 7 is presented the *Function Block Parameters* window for setting the parameters of the *M32C87 PID* block.

For the PID controller the following parameters have to be chosen: K_p , proportional gain; T_i , integration time; T_d , derivative time; β , fraction of command signal; N , high frequency limiter of derivative action; u_{min} , minimum saturation value; u_{max} , maximum saturation value; and T_s , sampling time.

The block has two input ports: the reference and the feedback signal, and one output: the control signal for the controlled process.

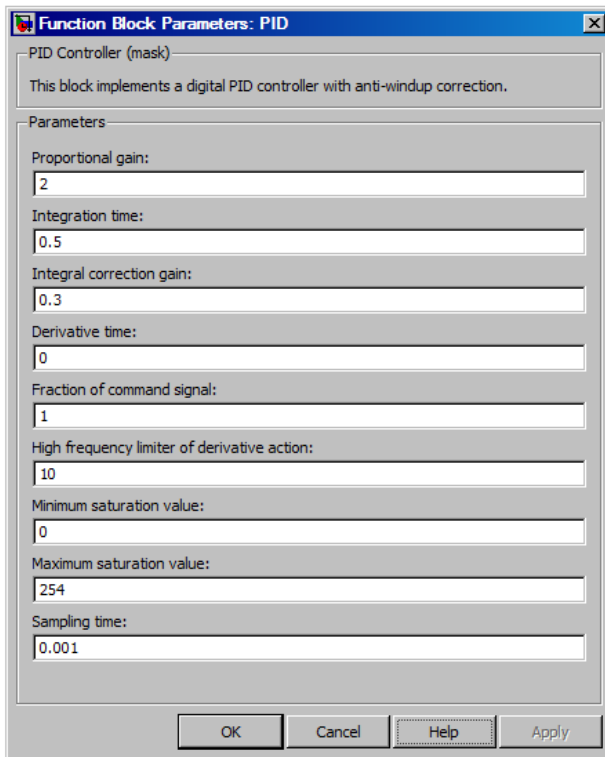


Fig. 7. Function Block Parameters window of the M32C87 PID block

3.2 M32C87 BLDC Motor Control Block

BLDC motors are one of the motor types rapidly gaining popularity. BLDC motors are used in industries such as appliances, automotive, aerospace, consumer, medical, industrial automation equipment and instrumentation.

BLDC motors do not use brushes for commutation; instead, they are electronically commutated. BLDC motors have many advantages over brushed DC motors and induction motors.

The *M32C87 BLDC Motor Control* block implements a trapezoidal six step control algorithm. The speed of the motor is controlled using PWM signals. The block uses as modulation technique the upper modulation. Only upper transistors of the three phase bridge are controlled using PWM signals while for the lower transistors arm transistors enable/disable signals are used.

The block has an input which represents the duty cycle of the PWM signals. The block can determine the speed of the motor using the signals generated by the Hall sensors. In this case, the block has an output port which represents the speed of the BLDC motor in rotation per minute. Also, for speed measurement an encoder can be used. In this case the block does not have an output port.

Fig. 8 presents the *Function Block Parameters* window for setting the parameters of the *M32C87 BLDC Motor Control* block.

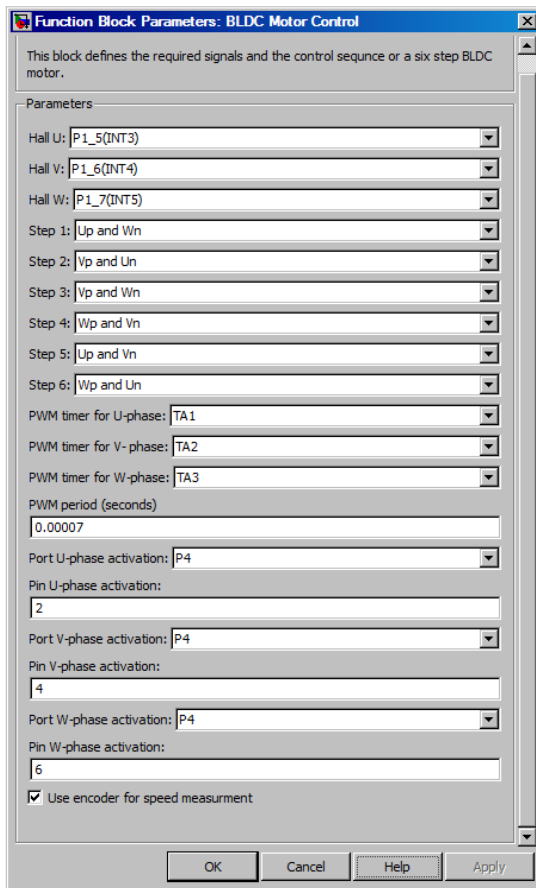
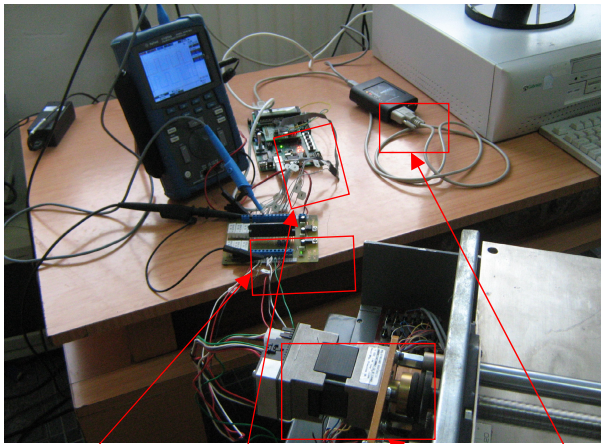


Fig. 8. Function Block Parameters window of the *M32C87 BLDC Motor Control* block

For the *M32C87 BLDC Motor Control* block the following parameters have to be specified: the input pins on which the signals generated by the Hall sensors are applied, the commutation sequence, the modules of timer *A* which are used for PWM signals generation, the period of the PWM signals, the pins which are used for enabling/disabling of lower transistors of the three phase bridge.

4 Application Example

A practical application for a Computer Numerical Controlled (CNC) machine is implemented. The stepper motors of an old CNC machine are replaced with BLDC motors, in order to obtain higher positioning accuracy. A ModSDKM16C-CM32C87 evaluation board, equipped with a Renesas M32C87 microcontroller is used as target processor. The motor which is controlled is a Hurst-Emerson BLDC motor [18]. In order to communicate between the embedded processor and the PC, an USB to CAN converter produced by Systec-Electronic [19] is used. The test setup is presented in Fig. 9.



Driver ModSDKM16C-CM32C87 BLDC Motor USB-CAN Converter

Fig. 9. Test setup

The closed loop Simulink model for the control of an axis of the CNC machine is presented in Fig. 10. The model of the control algorithm implements a cascade control topology. For the position loop a PI controller is used while for the speed loop a P controller is used. The parameters of the controllers are computed using the method introduced by Kessler [20]. The position reference is a rectangular signal, which is applied on the reference port of a *M32C87 PID* block. The output of the position controller sets the reference of the speed controller. The speed controller computes the duty cycle of the PWM signal which is used for the control of the BLDC motor mounted on axis of the CNC machine.

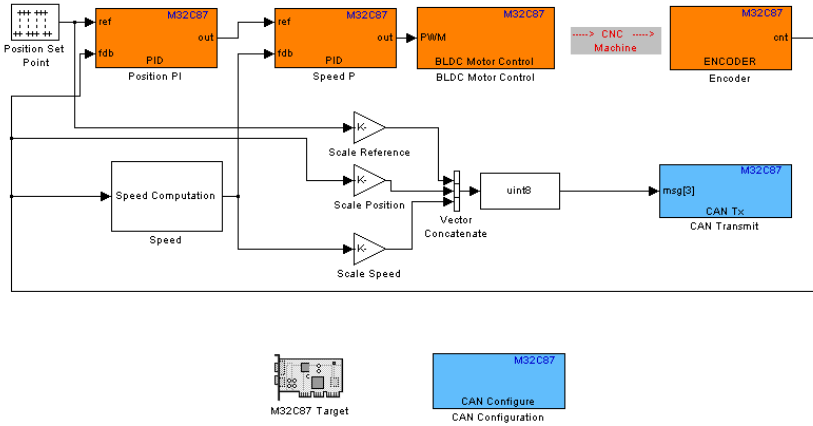


Fig. 10. Simulink block diagram for the control of an axis of the CNC machine

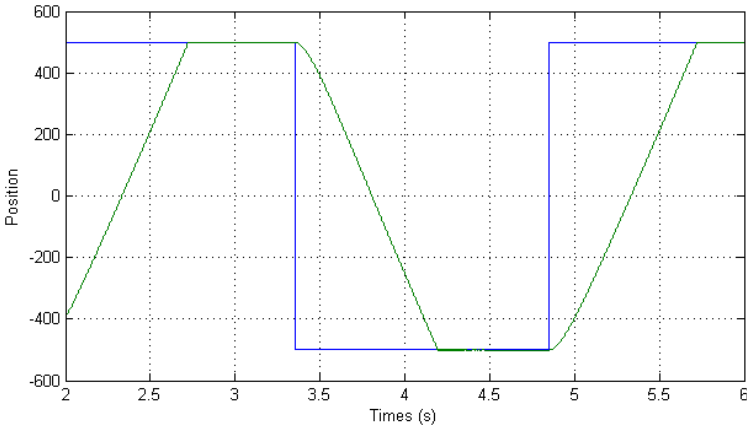


Fig. 11. The positing of the axis of the CNC machine

A configuration *M32C87 CAN Configure* block is added to the model. It does not connect to any other blocks, but stands alone to configure the selected CAN module. The reference position, the measured position and the speed are concatenated in an array and are sent over the CAN bus using a *M32C87 CAN Transmit* block.

A target preference block has to be added to the model. The *M32C87 Target* block does not connect to any other blocks, but stands alone to set the target preferences for the model. After the validation of the model the code generation process is invoked, and the generated executable file is downloaded to the Renesas *M32C87* processor

The positioning of the axis of the CNC machine is shown in Fig. 11 and the speed in Fig 12.

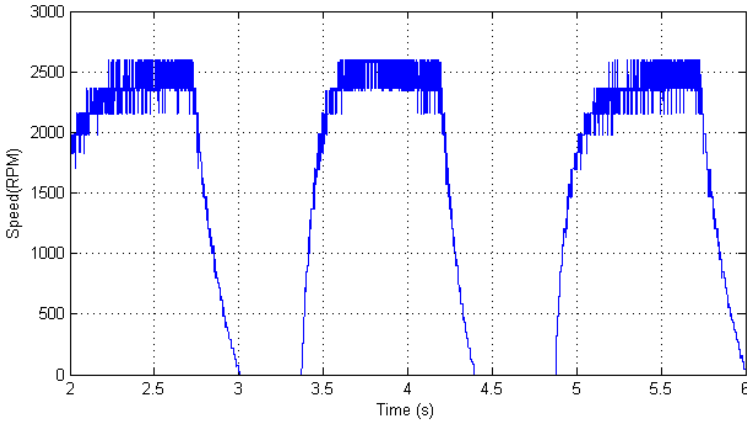


Fig. 12. Speed profile of the axis of the CNC machine

5 Conclusions

A contribution of the paper is the implementation of a RCP toolbox for the Renesas M32C87 microcontroller, using as CACSD tool Matlab/Simulink. The RCP approach removes the need to write software by hand and allows the engineer to focus instead on improving the performance of the control algorithm. The toolbox contains five libraries: *I/O drivers*, *Digital Motor Control*, *Controller Area Network* and *Serial Communication Interface*.

A practical application for the control of an axis of a CNC machine using BLDC motor is presented.

Acknowledgment. This paper was supported by the project “Progress and development through post-doctoral research and innovation in engineering and applied sciences – PRiDE - Contract no. POSDRU/89/1.5/S/57083”, project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

References

1. Hanselman, H.: Automotive control: from concept to experiment to product. In: Proc. IEEE International Symposium on Computer-Aided Control System Design, Dearborn, pp. 129–134 (1996)
2. MatWorks, <http://www.mathworks.org>
3. Rebeschies, S.: MIRCOS - microcontroller-based real time control system toolbox for use with Matlab/Simulink. In: Proc. IEEE Int. Symp. Computer Aided Control System Design, Kohala Coast, pp. 267–272 (1999)

4. Hercog, D., Curkovic, M., Jezernik, K.: DSP Based Rapid Control Prototyping Systems for Engineering Education and Research. In: Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, Munich, pp. 2292–2297 (2006)
5. Hanselmann, H.: DSP in control: the total development environment. In: International Conference on Signal Processing Applications, Boston, pp. 1647–1654 (1996)
6. MATLAB/Simulink device blocksets for dsPIC DSCs, http://www.microchip.com/downloads/en/DeviceDoc/MATLAB_Device_Blocks_51771b.pdf, <http://www.microchip.com>
7. Simulink - Embedded Target for PIC, <http://www.kerhuel.eu>
8. Duma, R., Dobra, P., Dobra, M., Sita, I.V.: Rapid Control Prototyping Libraries with Application for Positioning Systems. In: Proceedings of the International Conference on Control Systems and Computer Science, Bucharest, pp. 460–465 (2011)
9. Tomesc, L., Betea, B., Trusca, M.: Power Quality Measurements Using a Low-Cost Embedded Solution. In: Proceedings of the International Conference on Control Systems and Computer Science, Bucharest, pp. 100–104 (2011)
10. Scilab/Scicos, <http://www.scilab.org>
11. RTAI – Real Time Application Interface, <http://www.rtai.org>
12. Comedi, <http://www.comedi.org>
13. Ravn, O.: Adaptive control using the adaptive toolbox-TAT for Scilab/Scicos. In: 14th IFAC Symposium on System Identification, Newcastle (2006)
14. Evidence, <http://www.evidence.eu.com/>
15. Duma, R., Dobra, P., Trusca, M., Dumitrache, D., Sita, I.V.: Rapid Control Prototyping Educational Toolbox for Scilab/Scicos. In: Proc. of European Control Conference, Budapest, pp. 4611–4616 (2009)
16. Koivo, H.N., Tantt, J.N.: Tuning of PID controllers: survey of SISO and MIMO techniques. In: Proceedings of Intelligent Tuning and Adaptive Control (1991)
17. Yamamoto, S., Hasimoto, I.: Present status and future needs: the view from Japanese industry. In: Yamamoto, S., Hasimoto, I. (eds.) Proceedings of 4th International Conference on Chemical Process Control, Texas (1991)
18. Emerson, <http://www.emersonmotors.com>
19. SystecElectronic, <http://www.systec.com>
20. Voda, A.A., Landau, I.D.: A method for the auto-calibration of PID controllers. *Automatica* 31, 41–53 (1995)

From HTML to 3DMMO – A Roadmap Full of Challenges

Alin Dragos Bogdan Moldoveanu, Florica Moldoveanu, Victor Asavei,
Alexandru Egner, and Anca Morar

Faculty of Automatics and Computer Science, University "Politehnica" of Bucharest,
Romania

{alin.moldoveanu, florica.moldoveanu, victor.asavei,
alexandru.egner, anca.morar}@cs.pub.ro

Abstract. WWW is everywhere, embraced by humankind to share information and collaborate. The basic schema of hyperlinked web pages has evolved with revolutionary technologies, altogether with the devices used to access the web, but still carries on strong limitations. Virtual reality and 3D MMO, with their sheer power of representation and interaction, is seen by many as the future of human computer based interaction and collaboration and would, naturally, mix with the WWW model sooner or later. The complexity of the technology, from technical point of view but also with many interdisciplinary issues (medical, ethical, economical, socio-political etc.) would make this transition challenging and exciting. This paper classifies, identifies and proposes basic conceptual solution for these issues - of moving from interconnected web pages to interconnected 3D virtual spaces with massive number of interacting users.

Keywords: virtual reality, MMO, human computer interaction.

1 Introduction

Our world is not 2D. Yet, our Internet is still mostly 2D: the world wide web, seen as a huge network of interlinked web pages, is probably between 90 to 95 percents build from http and html. The technology, aging back to the beginnings of Internet, benefits from its simplicity but carries many strong limitations in terms of power of representation and interaction.

The two decades starting from 1991 when HTML was introduced were market of continuous improvements: dynamic generation of web pages, plug-in system, Java, JavaScript, technologies like Ajax, Flash and many others. Web pages are still getting more and more dynamic and appealing. But if we look at the big pictures, the feeling is at least a bit contradictory. All these technologies can and actually do integrate into the http-html backbones because of its inherent simplicity, but their main purpose is exactly to overcome its many limitations.

One radically different technology is becoming more and more popular: a branch of virtual reality known as online virtual worlds. Most of these are fully

3D, highly dynamic and populated by sometimes hundreds of thousands of users that interact in real time. These representation and interaction powers are way beyond the capabilities of classical or even improved web pages.

This power however comes with some costs and challenges, both from technological and other points of views. That's why these emergent technologies are not yet widely used outside the world of computer games. But, as they become more mature, we are just witnessing the advent of them being used as ground for a plethora of "serious" application.

In this paper we explore if these online 3D virtual worlds can actually substitute, somewhere in our near future, the current http-html based web.

We will show how virtual worlds can offer more than classical web pages.

We will see what still stops their expansion - both from technological and interdisciplinary point of view.

And finally, we suggest some possible solutions to some of these challenges, that would mark a roadmap of expansion of 3D online virtual worlds as new computer based collaboration technologies.

2 Limitations and Workarounds of the HTTP-HTML Paradigm

As one of its founders comments, HTML was "bad and incomplete" in the beginning, but there was really no time in the development process to improve it [1].

People see major limitations in terms of lack of structure. Of course, this is quite normal, as HTML was designed with a main goal of being as simple as possible and mainly just a mean of sharing and connecting information [2].

Meta-tags and other mark-up information structuring aides are starting to do a pretty good job of transforming the web from unstructured to structured information.

The most widely used schema, of presenting HTML pages through web browsers that fetch them through HTTP also brings major limitations into the equation - regarding the dynamicity of the content.

Improvements to HTML itself and the addition of plug-ins (such as real player), scripting languages (such as Java Script) and dynamic technologies (Java, Ajax) allow more power regarding the visual aspect of web pages and also their dynamicity.

But none of these technologies seem to be more than just workarounds for the fact that both HTML and to some extent HTTP were designed without envisioning the full scale of what online interaction and collaboration would become.

3 The Online Virtual Worlds

We can see some of the things that http-html is missing if we look at some alternative technologies.

One of them is that of the online virtual worlds. For a while, it was disregarded and seen by many just as a trivial branch of virtual reality meeting the computer games world. But is not the 1st time that gaming stimulates new technologies.

Nowadays, virtual worlds are approaching their maturity and many are starting to talk about them as a mean of unequalled power of interaction and collaboration over the internet.

An online virtual world is a computer simulated world, accessed and modified by its users through internet, in real time.

Virtual worlds are intended for their users to inhabit and interact within.

3.1 Content

Their content can correspond to a 2D or 3D environment. It can be anything that we can expect to see in a real or imaginary world: natural or artificial landscape, characters, weather etc.

Users are represented through avatars. Avatars are also used to navigate within the world, perform various actions and interact with the others.

Besides the visual representation, these worlds can include sounds (2D or 3D) and sometimes advanced forms of interaction, like force feedback and other haptics.

3.2 Semantics

From the semantic point of view, each world has its own concepts and goals.

Some rules are started in the virtual world policy and terms of service and enforced by the software that manages the virtual world.

For example, such rules can be the rules specific to a game, rules regarding the language interaction between players, etc.

3.3 Real-Time Massively Multiplayer

Some virtual worlds are single user, some allow a small number of multiple users (like 4 or 8 or 20), while the most important of them allow the simultaneous access of hundreds to hundreds of thousands of users.

Note that all these users interact in real time. For example, when one user is moving its avatar in the virtual world, this will be immediately seen by all the other users.

This real-time massively multiplayer aspect opens us unprecedented possibilities for users interacting and collaboration with each other.

3.4 Applications

Computer games creators have been the firsts to grasp the power of this technology.

MMO (massively multiplayer online) games lists nowadays hundreds of successful titles and make up for a global market projected to reach US\$14.4 billion by 2015 [3].

Games range from hardcore gaming involving fights between players to achieve various goals to social or casual gaming, which put accent on the social interaction between players or various types of entertainment.

Close to games are environments like Second Life. According to its website, Second Life is a free 3D virtual world where users can socialize, connect and create using free voice and text chat [4]. Second Life has nothing of a game, no competition aspect, the accent being on socializing. The content is mostly created by users with the help of some content creation tools. Despite being labeled as free, it generates huge revenue for its owners, as land inside the Second Life virtual world is sold or rented.

Other pioneer field that starts to note the potential or virtual worlds are education - interested in its potential for distance/online education but also of the new aspects of interactive and non classical teaching.

Online advertising, online conferencing and others are starting to see virtual worlds as a serious application domain. New software packages addressing the creating of such types of application have just been created in the last years.

3.5 Comparing to Web Pages

If we just look at the representation and interaction power, we can see that online virtual worlds are superior in many aspects to classical web pages.

Table 1. WebPages vs. Virtual Worlds

	Webpage	Virtual World
3D	no	yes
Haptic	no	sometimes
Representation power	limited	unlimited
Hyperlinks/Hyperjumps	yes	yes
Real-time	partially	yes
Immersive	partially	highly

Even from such an easy comparison, is easy to conclude that virtual worlds would offer way more beyond the possibilities of web pages in terms of representation and interaction power.

Basically a virtual world can represent almost everything we can have in a real world, while allowing its users to explore and interact in a very immersive manner. This degree of immersion increases more and more as advanced 3D rendering devices like head mounted displays or other interaction devices like gloves, wands, virtual spheres, etc. are getting accessible in terms of costs.

Then why are not yet widely adopted? As shown in next sections, *there is still a plethora of issues that limit the wide adoption of virtual worlds*. Some of these are related to the immaturity of the technology and to its inner algorithms regarding

the real-time MMO aspect. Other limitations are not technological but rather social, medical, economical or political. We will look at each of them and possible generic solutions.

4 Technological Challenges for the Expansion of 3D MMO Virtual Worlds

Main obstacles hindering the wide scale adoption of 3D massively multiuser online virtual worlds (3D MMO) regard their ease of creation, their scalability regarding number of simultaneous users and their interconnectivity - or shall we rather say the lacks in all these aspects.

4.1 Creation

The ideal would be that creating a 3D MMO virtual world should be as easy and accessible as creating a website.

Creation involves both the development of the software that runs the simulation and the creation of the artifacts that populate it.

There are nowadays many software packages for designing and running 3D virtual environments. However, very few of them have the MMO functionality. And those having it are very complex and hard to use to be accessible to the large public.

In fact, until few years ago, creating a MMO game for example was done in many cases almost from scratch and was estimated to cost 10-50 million \$. Of course, these costs were accessible only to big companies. Once a company has created such software, it had little or no interest to make it available for free reuse to others. However, the next step was inevitably the evolution of several dedicated frameworks, with 3D MMO functionality, but these were also expensive, according to various licensing schema. Eventually, *in the last years, we see such 3D MMO frameworks appearing also in the open source field*, some of them available with no real restrictions and costs to anyone. However, they still lack the maturity, being very complex and in many cases full of bugs. The software development effort, even based on them, is still high, but we can certainly say that improvements are done day by day.

Most of the artifacts from 3D MMO virtual worlds are 3D models: relief, vegetation, buildings, furniture, vehicles and characters - real or imaginary. They are, in all cases, created with specialized modeling software. The procedure differs by whether an imaginary model is created, or one that copies a real world model. But in both cases, there is a serious amount of handwork involved, which keeps this activity costly also.

A huge interest would be in the low-cost reproduction of real environments into virtual replicas, for example creating a virtual model of some campus or some city or factory or museum etc. Low costs would require the process to be almost completely automated and to use inexpensive equipments. *However, the existing methods*, based either on photogrammetry or laser scanning, either don't give yet very good results or are highly expensive, thus *are unusable or not scalable for most cases*.

4.2 Scalability

There is something about the nature of real-time MMO environments that makes their scalability one of the biggest challenges for software architects: each action made by a user will change the state of the virtual world; each such change must be immediately calculated and transmitted to all the other users involved. The effects of user actions by themselves can sometimes require some serious computational effort - like collision detection, spells or combat moves (in games) etc. Also they depend a lot on the position and state of the other users.

All these mark a big difference from most software systems, for example databases, when most user interactions with the system do not directly affect and do not directly depend upon the dynamic state of a big number of other users. *In fact, for most software systems, computational effort grows close to linear with the number of users, while in the case of 3D MMO the dependence is quadratic.*

Add to this the fact that the total time for calculating the effects of a user action and informing all the other affected users should be in the range of tens to hundreds of milliseconds - otherwise users will experience the phenomena usually called lag, which reduces a lot the immersion.

These two contradictory requirements - the ability to handle huge computational effort and short response time - make it a real challenge to design the system responsible for running such a virtual world.

There is a general algorithmic solution for this, having its origins in computer graphics, named *space partitioning*. Basically, the idea is to divide the whole virtual 3D space into several non-overlapping regions and to keep track of the users located in each region. Then, most of the calculations regarding the actions of a user can be reduced only to the other users in that region (and sometimes in adjacent regions) - resulting in reduced computational cost.

From an architectural point of view, this idea offers an additional benefit, as each region can be assigned to be managed by different processing unit (a processor or a computer). This allows for scalability beyond the computational power of a single computer.

There are many ways of dividing the space into regions: static or dynamic, manual or automatic, quad trees or BSP trees, etc. There are also even more methods and architecture for taking care of the implicit overhead that comes from the need of the region managers to cooperate to maintain the global state of the simulation.

Typically, from several practical reasons like economical or security, which we will skip here, the architecture of 3D MMO virtual worlds is client server: the server keeps the overall state of the virtual world and enforces the rules, while the client software is run the users to merely access the simulation, offering them the visualization and interaction functions. To handle the scalability issue with the subspace division approach and other ideas, *such servers are actually server farms, sometimes with tens or hundreds of computers, each computer acting as a region manager.* Of course, this translates in big operational costs for the corresponding virtual world. An architectural example is illustrated in Fig 1.

Some experiments try to handle the work division with peer-to-peer approaches. Naturally each peer would take some of the workload. This would reduce the costs for the central server, compared to the client-server architecture. But these approaches pose big issues concerning consistency and security; they are yet theoretical or experimental and didn't make it into the industry.

In conclusion, designing a cheaply scalable architecture for 3D MMO virtual worlds is still a challenge for software architects. *A general enough solution is needed, that would allow flexible dynamic scalability by number of users. This would, in turn, permit the economic creation of general purpose 3D MMO server farms, suitable to host dynamically any 3D MMO virtual worlds, similar to how web hosts services can host any distinct web sites. Then the operation costs for 3D MMO virtual worlds will start to get very low, as it is for websites today.*

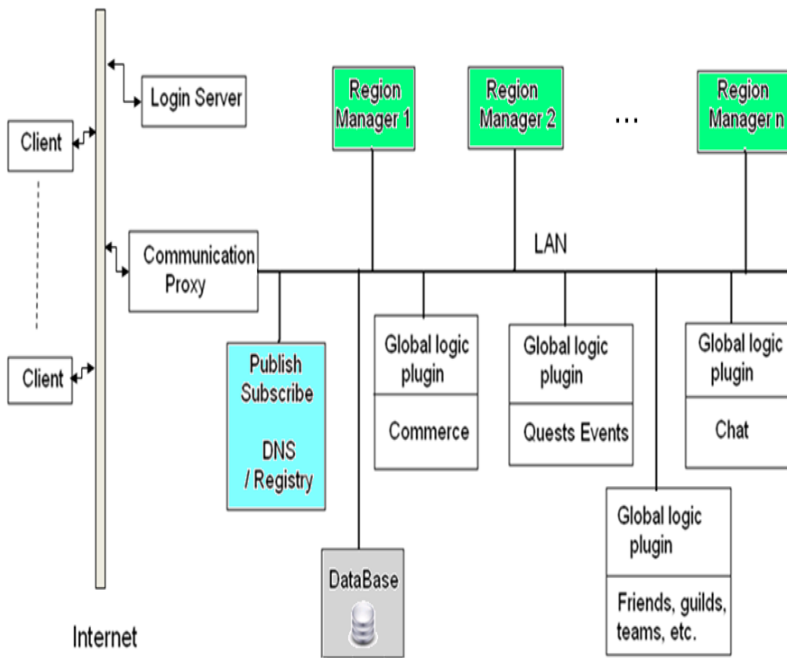


Fig. 1. General architecture of a 3D MMO server farm

4.3 Interconnectivity

Up to this point, most virtual worlds are totally independent one from another. There are no connections between worlds. The user avatar exists only within one world and can't be moved/transferred/be visiting to another world.

This is partially due to historical reasons: when the creating of the majority of 3D MMO worlds started, the technology was still in its beginning and no one really thought about interconnectivity and interoperability.

However, the big reason is in marketing/economics. We believe that most successful software companies developing such worlds are actually not interested in opening them up. They have 2 reasons to do this:

- They invested a lot in the development and don't feel like sharing any part of the technology.
- They prefer to milk as much profit as possible from the operation of their successful particular worlds, without risking having their users move to other worlds.

However, the technology for MMO finally made it into the open source. This was partially due to some initiative from the company that owns Second Life, which at some point wanted to open source. Later they changed their mind and policy but it was too late: people got the concepts and started to build some powerful open source solutions, like OpenSim.

None of these solutions is yet mature enough to be easily used for industry. However they are evolving fast and we predict they will reach such maturity in a few years.

With the open source came naturally the interest for interoperability of these virtual spaces. Common standards and protocols are defined and implemented. This is a huge step forward, but will still take some time until a network of interconnected and interoperable 3D MMO virtual worlds, where users can roam freely from one to another, would be established.

5 Interdisciplinary Challenges For 3d MMO Virtual Worlds

However free of mundane limitations they appear to be, virtual worlds actually bear strong connections to the real world - which in turn are subject to many issues. Their expansion would only make these issues more visible and important, and yearning for good solutions. In this paragraph we'll have a brief look at these controversial aspects and suggest general solutions for them.

5.1 Context

In technological advanced countries, many youngsters spend more time playing MMOs than playing outside with their real friends or neighbors. These virtual environments are part of their daily life and culture.

The degree of immersion can get very high, both in terms of time extension and focus.

Actions and behavior in such spaces is in many cases, subject to almost no restrictions.

Users' pool is quite large. In same space we can encounter people of many national, racial, social, economical and educational backgrounds.

All this freedom and diversity can and in fact has many positive effects, but sometimes leaves room for abuses and can create serious problems.

5.2 *Social*

In same MMO virtual space, users can differ a lot in terms of:

- age
- country of origin
- education
- cultural background
- financial status
- traits in the virtual space (abilities, guilds, teams etc.)

The immediate *positive effects of this diverse exposure of participants* can be accounted with:

- knowing and understanding people different from oneself, in terms of all the aspects listed above
- intercultural collaboration

However, *there can be also many negative aspects*, due to:

- the competitive nature of most MMO games
- the lack of well defined virtual policies and laws

Examples of explicit abuses can be [5]:

- aggressive language
- threats about the real life of some users, with the purpose of obtaining advantages in the virtual world
- racial issues
- sexual corruption of minors
- exposure through observation or dialog to any kind of negative influences.

An example of implicit abuse can be considered the fact that many MMO game designers strive to make their game as addictive as possible, sometimes using psychological techniques, even if their users are, in most cases, people of your of very young age.

5.3 *Medical*

The humoristic prototype of the gamer is a thin, weak, pale adolescent, speaking very fast and excited about last released games or add-ons. This is quite normal, as many of the game addicts spend that much time and energy in virtual worlds that they forget to take care of themselves in terms of health and hygiene.

There can be hard short or long term effects of such life style. For example, there are well known the cases of game addicts took directly from internet cafes in China to hospital.

Mental health of virtual spaces users can also be considered a very important issue. The competitive aspect often requires a lot of concentration, over long period of times, which sometimes exceed the possibilities of some users. In time, this can provoke chronic fatigue or losing much of the contact with, or adaptation to the real world.

Advances in virtual reality equipment will increase these phenomena even more.

5.4 Psychological

Virtual spaces (games) addiction is nowadays considered by many researchers as a specific psychological illness, at least to the extent that it indirectly affects the social life and the health of individuals. Many studies have identified similarities to autism [6], however many differences exist.

Many countries have already created special treatment facilities for internet or gaming addicts. Some countries, e.g. China, have also policies that limit the time spent online in MMO games.

All these tendencies to judge very hard the phenomena are mostly caused by the shocking nature of some extreme addiction cases encountered. *However, surprisingly, there is no scientific study yet to prove that MMO virtual spaces do have a negative effect on their users, with statistics.* On the contrary, some studies have pinpointed for example, lower values of indicators for introversion or timidity for MMO players then for other subjects [7].

We can't conclude yet on this matter, but *we can at least suppose that all the restrictive measures taken by some countries have other reasons then strict medical, more likely economical or political.*

5.5 Economical and Legal

Most virtual spaces include some own virtual economy, with concepts like virtual propriety, currency, exchanges and so on. All these concepts are well defined in the game but they lack the support of real world legislation. We identified 3 issues that need to be solved so that individual rights will be really protected in the virtual world and that virtual values can be integrated in the real world economy:

- de-virtualization of the virtual property rights
- legislation and taxing the virtual economical activities
- protecting of virtual economies

De-virtualization of the virtual property rights:

There is yet no legislation to define and protect the assets owned by a user in a virtual space.

In most cases, these are defined implicitly by a contract (usually Terms or Service) between the user and the company operating the virtual world. In most cases these ToSs are overprotective to the company and offer almost no protection to the user, specifying that all the assets are owned by the company and the user only had the right to use them according to the game rules. Most users are not even aware about these aspects, as they accept the ToSs without careful examination, during the sign-up process.

These ToSs do have some huge disadvantages, both for users and for the operators of virtual spaces [8]:

- users: have no absolute guarantee over their property of various virtual assets
- users: some ToSs are actually in contradiction with general law principles in use in some countries; however, they do operate as a de facto solution
- operators: ToS just enables some juridical protection, but the company can lose more if it fails to protect virtual assets of the users - because a lot of other users will leave that virtual space; therefore the companies constantly have to invest a lot in virtual spaces security and fight hackers, exploiters and scammers, without having proper backup from the law to help them punish such persons.

An ideal solution would be to *give to virtual assets all the rights and protection enjoyed by real properties.*

This does not appear impossible or even very hard. However some important aspects that need clarification are:

- defining the conditions when virtual assets cease to exist, because their virtual space cease to exist
- national vs. international legislation issues (due to the naturally global extent of most virtual spaces)

Legislation and taxes for virtual enterprises:

Many of the users are involved in the production and trading of virtual assets. As long as these do not produce real profit, they can be considered just entertainment and can't be subject to any taxes. But we also encounter the situation when a user pays real money to other user, in exchange for virtual assets like virtual gold, diamonds, special items, land, protection etc. This can occur sporadically or in a systematic and organized manner - in this last case being called "gold farming" (starting from the traditional activity of mining or gathering virtual gold).

Gold farmers usually produce:

- high level and powerful avatars
- virtual money or any quantifiable resource
- special objects, hard to obtain (like powerful weapons, etc).

Such virtual to real business are very popular nowadays. Usually buyers are people from rich countries, while gold farmers are youngsters from China,

Philippines, India and sometimes Egypt or some eastern European countries. The extent of the business is quite big, for example a rough number of 100.000 work in this in China only.

However, from the point of view of most virtual spaces operators, especially games, this activity is not beneficial, as it tend to affect the economy of the virtual worlds and in turn, some mechanism through which operators make real cash from the virtual economy. Basically it creates competition in turning virtual assets to real money for the operators themselves.

As a consequence, most operators try to prevent such activities with:

- banning the accounts/avatars involved in such trades
- stopping the trading of highly valuable objects etc.

However none of these measures works well enough and can't stop gold farmers to make profit.

Novel solutions adopted by some operators follow a different idea: *rather than forbidding the transactions, they encourage them and create safe mechanism for such transaction, while at the same time taxing them* or allowing direct conversion of virtual currency in real currency at a rate convenient for the operator (e.g. Second Life Linden Dollar).

Also some states are starting to see the economical value of such transactions. Accordingly, is looking for ways of giving them legal value and impose taxes.

Evolutions and issues:

More and more real world influences over the virtual spaces, together with the laws and protection of virtual property and taxation of profit making virtual to real enterprises will have many benefits. But at the same time it can pose some special issues. A consequence, for example, will be that it will *reduce the full control that virtual spaces operators currently have over their virtual space and, in particular, over the balance of virtual economies.*

This will open doors for manipulation of virtual economies, with techniques specific to early stock exchanges.

Legislative and technical solutions would have to be found to prevent such possibilities without compromising the dynamics and freedom specific to virtual spaces economies.

5.6 Conclusions

As for the technical factors, the expansion of virtual worlds makes these issues more and more important. All of them are nowadays looking for adequate legislative solutions, but these must be supported and enables the advanced technical infrastructure.

6 Virtual Worlds, Augmented Reality and the Future

Both these concepts are intertwined operating at the intersection of the real world and the worlds of computers and information.

Apparently they are competing paradigms when it comes to enhancing the users' life: whether to model something in a virtual world and benefit from all the freedom of imagination or whether to augment the real world with information generated by computers?

The competition is just apparent, as the two paradigms address complementary aspects. The answer to the question which one should be use can be both.

The advocates of augmented reality, or ubiquitous computing, excited by all of the new mobile devices and applications that have emerged in the last few years, tend to advocate it as a universal solution and an absolute winner [9]. However, there are at least two features that virtual worlds alone have as distinct from any augmented reality concept, features which make it impossible to take them out of equation:

- their free of limitations aspect - which fits and feeds an ineffable trait of human mind and consciousness - means they will always be preferred by many users for entertainment, training, experiments or for exploring the limits of human imagination
- more practical, their online nature means they can be accessed from anywhere, from any place in the real world, unlike the real world which requires physical presence

Not surprisingly we can see how the interest in these technologies evolves in internet websites content or searches.

The following figure compares average worldwide traffic for the two keywords in all years, according to Google collected data: virtual worlds, augmented reality:

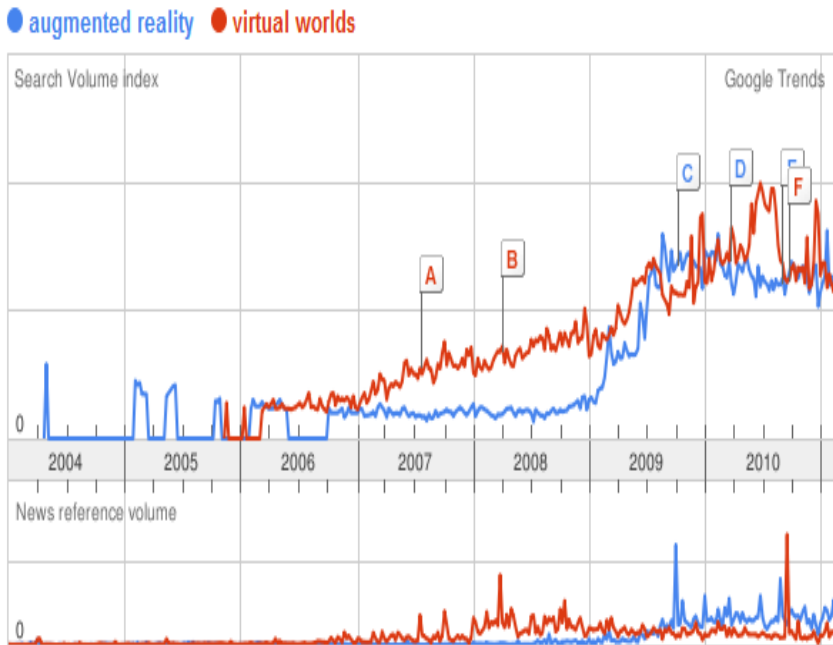


Fig. 2. <http://www.google.com/trends?q=augmented+reality%2Cvirtual+worlds> [10]

Acknowledgments. The research presented in this paper was supported by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

References

1. Cailliau, R.: Personal Interview (1998)
2. Berners-Lee, T.: Information Management: A Proposal. CERN, Geneva (1989)
3. Global Industry Analysts. Massively Multiplayer Online Games - A Global Strategic Business Report (2010), [http://www.companiesandmarkets.com/Summary-Market-Report/mmog-\(massively-multiplayer-online-games\)-a-global-strategic-business-report-317174.asp](http://www.companiesandmarkets.com/Summary-Market-Report/mmog-(massively-multiplayer-online-games)-a-global-strategic-business-report-317174.asp)
4. Second Life, <http://secondlife.com/>
5. Anderson, C., Dill, K.: Video games and aggressive thoughts, feelings, and behaviour in the laboratory and life. *Journal of Personality and Social Psychology* 78 (2000)
6. Arendt, S.: Study: Game Addiction Similar to Autism (2008), <http://www.wired.com/gamelifelife/2008/04/study-videogame/>
7. Lake, C.: Video game addicts are not just 'shy nerds' (2008), <http://www.addictioninfo.org/articles/2760/1/Video-game-addicts-are-not-just-shy-nerds/Page1.html>
8. McKee, M.: Real Jurisdiction in a Virtual World: Interconnecting Jurisdictional Doctrine with End User License Agreements. Spring Meeting Speaker Papers - AIPLA CLE Publication (2008)
9. Schonfeld, E.: Augmented Reality Vs. Virtual Reality: Which One Is More Real (2010), <http://techcrunch.com/2010/01/06/augmented-reality-vs-virtual-reality/>
10. Google Trends, <http://www.google.com/trends?q=augmented+reality%2Cvirtual+worlds>

Schedulability Guarantees for Dependable Distributed Real-Time Systems under Error Bursts

Huseyin Aysan, Radu Dobrin, and Sasikumar Punnekkat

Mälardalen University, Västerås, Sweden

{huseyin.aysan,radu.dobrin,sasikumar.punnekkat}@mdh.se

Abstract. In dependable embedded real-time systems, typically built of computing nodes exchanging messages over reliability-constrained networks, the provision of schedulability guarantees for task and message sets under realistic fault and error assumptions is an essential requirement, though complex and tricky to achieve. An important factor to be considered in this context is the random nature of occurrences of faults and errors, which, if addressed in the traditional schedulability analysis by assuming a rigid worst-case occurrence scenario, may lead to inaccurate results. In this work we propose a framework for end-to-end probabilistic schedulability analysis for real-time tasks exchanging messages over Controller Area Network under stochastic errors.

Keywords: real-time systems, task scheduling, CAN, dependability, reliability, fault tolerance, schedulability analysis.

1 Introduction

Real-time systems are computer systems in which the correctness of the system depends not only on the logical correctness of the computations performed, but also on which point in time the results are provided [1]. Delivering a result at a point in time beyond the latest possible, i.e., after its deadline, may result to catastrophic consequences in *safety critical real-time systems*. Examples of such systems are medical control equipment nuclear power plants, or vehicle control systems.

A real-time system typically consist of a number of a number of *resources* (e.g., processing nodes connected by communication mediums), a number of *tasks* typically communicating over a field buss, e.g., Controller Area Network (CAN) by exchanging *messages*, designed to fulfil a number of *timing constraints*, and a *scheduler* that assigns each task and message a fraction of the processor(s) time or bus bandwidth, according to a *scheduling policy*. Events like tasks or messages are usually *periodic* or *non-periodic* depending on their pattern of occurrences. While periodic events consist of an infinite sequence of invocations, called *instances*, non-periodic ones are invoked by the occurrence of another event. The choice of

tasks, messages and scheduling policy is made by the system designer to satisfy some original constraints imposed on the system. Consequently, tasks and messages are assigned a number of scheduling parameters, such as periods, deadlines or priorities, depending on the chosen scheduling policy.

Besides the real-time specific deadline constraint, the majority of safety critical real-time systems are typically characterized by dependability requirements. In essence, these systems have the major design objective to guarantee the properties of correctness and timeliness even under error occurrences. Further, these systems are typically built using several computing nodes that interact with each other in a distributed manner where reliable communication plays a crucial role for achieving overall system dependability. While the deadline constraint is addressed by using the response time analysis, the design of reliable end-to-end systems, involving task executing on processing nodes and exchanging messages over a network (i.e. CAN), requires usage of appropriate fault-tolerance (FT) mechanisms and analysis techniques jointly at node- as well as network level. However, the fundamental requirement for the design of effective and efficient FT mechanisms is a realistic and applicable model of potential faults, their manifestations and consequences.

In a large number of safety or mission critical systems, that typically employ the *preemptive* fixed priority scheduling (FPS) policy, real-time schedulability analysis techniques have been increasingly used in order to ensure that the strict timeliness requirements of the applications are met. The preemptive behaviour of FPS, although desirable to increase the schedulability bound, can on the other hand benefit of control mechanisms to address the potential high contexts switch costs [2]. The analysis has been also extended to CAN, where real-time messages are scheduled *non-preemptively* on the bus. CAN was designed in the 1980s at Robert Bosch GmbH [3] with a special focus on automotive real-time requirements and has been widely used in the automotive and automation industries due to its ease in use, low cost and provided reduction in wiring complexity. The most important feature of CAN from the real-time perspective is its predictable behaviour. Recent works have addressed the effect of the network delay on the performance of control systems [4]. CAN provides means for prioritized control of the transmission medium by using an arbitration mechanism which guarantees that the highest priority message that enters an arbitration will be transmitted first. This makes CAN amenable to response time analysis akin to those performed on fixed priority task sets. Volcano methodology used by Volvo [5] is an example of the acceptance of such analysis by the industry. The model underlying the basic CAN analysis assumes an error free communication bus, i.e. all messages sent are assumed to be correctly received, which may not always be true due to the interference from the operational environment or the faulty hardware components. These interferences cause errors in the transmitted data, which could indirectly lead to catastrophic failures. While in processor scheduling the designer is responsible to provide fault tolerance mechanisms, in CAN scheduling, to reduce the risks due to erroneous transmissions, CAN designers have provided elaborate error checking and error confinement features in the protocol. The basic philosophy of these features is to identify an error as fast as possible and then retransmit the affected

message. This implies that in systems without spatial redundancy of communication medium/controllers, the FT mechanism employed is time redundancy which addresses transient errors but could have an adverse impact on the latencies of message sets; potentially leading to violation of timing requirements. Furthermore, burst of errors typically affect several message retransmission attempts and contribute to potentially large response time that may deem the system unschedulable.

In this work, we propose an end-to-end schedulability analysis for real-time tasks executing on processing nodes and exchanging messages over CAN, under error scenarios.

2 End-to-End Probabilistic Fault-Tolerance Analysis (PFTA) under Error Bursts

In this section we present an end-to-end probabilistic fault-tolerance analysis (PFTA) for distributed real-time systems under error bursts. We assume a number of processing nodes executing real-time tasks that exchange messages over CAN. We first present PFTA for a single processor node where tasks execute scheduled under FPS. Then we show how PFTA is performed for message scheduling in CAN, and finally we introduce PFTA for transactions of tasks exchanging messages over CAN under error bursts.

2.1 PFTA for Processor Scheduling under Error Bursts

The approach begins with a performing a set of schedulability analyses that accounts for a range of worst-case scenarios generated by stochastic error burst occurrences on the response times of tasks scheduled under the fixed priority scheduling (FPS) policy. Then the probabilistic schedulability guarantees are calculated as a weighted sum of the conditional probabilities of schedulability under specified error burst characteristics.

In this subsection a single processor platform is assumed on which a sporadic task set is allocated which have deadlines equal to or less than their minimum inter-arrival times. Whenever an error is detected within a task, the affected task τ_i executes an alternate task with a worst-case execution time less than or equal to the original worst-case execution time of its primary, a deadline equal to the original deadline and a minimum inter-arrival time equal to the minimum inter-arrival-time of its primary. This alternate can typically be a re-execution of the same task, a recovery block, an exception handler, or an alternate task with imprecise computations. Errors are assumed to be detected just before the completion of the affected task instances.

The main sources of errors are assumed to be electromagnetic interferences (external faults), and transient hardware faults (internal faults) that affect, e.g. the sensors and the network systems. Examples to the considered errors are incorrect input values from sensors, or failure in delivering the output values via network messages. Errors that are propagated into tasks are detected at the end of task executions by observing, e.g., the out-of-range output values or omitted outputs.

Examples to the assumed error detection mechanisms are usage of sanity checks, range checks, checksums for the value correctness and the usage of watchdog timers for the time correctness. Watchdog timers are assumed to be implemented as simple hardware units that run in parallel with the tasks and interrupt in case of detected errors and the overhead of the value error detectors are included in the WCETs of the respective tasks.

Each fault may result in errors in the form of an error burst for a random duration. The distribution regarding the duration of the faults is very much domain specific, and, in this section, it is assumed that the information regarding the probability distribution of the fault durations is available. Other parameters assumed to be given related to the error model is the error rate λ and the mission time L .

In this work, it is assumed that no task can successfully finish between errors within a burst. Furthermore, any task instance scheduled even partially under the error burst will be considered as affected by the error.

Methodology Overview. The goal of this approach is to find the probability that the given task set is schedulable during a mission time L under the specified error model. This probability is dependent on the error characteristics (the minimum inter-arrival time between bursts, T_E , the possible values for the fault duration l_j , and the probability distribution $f(l)$) and can be derived from the conditional probabilities that the task set is schedulable under specific sets of values for these parameters.

The analysis begins with finding the maximum number of error bursts, n , that can hit any task in the task set. Considering the interplay between T_E and l_j a set of sensitivity analyses is performed to derive the minimum inter-arrival times between error bursts (T_E) for each possible combination of n fault durations by assuming the worst-case task executions and error overheads.

One should note that the derived minimum inter-arrival times are actually *upper bounds* which may never be reached. This is due to the nature of the inexact worst-case assumptions, such as the WCETs of the tasks, which correspond to upper bounds rather than exact worst-case values.

The fault duration combinations and the corresponding upper bound T_E values are then used to find the *conditional* probabilities of schedulability which are actually lower bounds for the exact probabilities. Finally, the lower bound probability of schedulability is computed as a cumulative sum of these individual conditional lower bound probabilities, i.e. by unconditioning the probability of schedulability with respect to the fault durations. The steps involved in the methodology are illustrated in Figure 1 and briefly described below.

- STEP 1. The analysis begins by finding an upper bound for the maximum number of error bursts that can hit any task in the task set while the task set is still schedulable.
- STEP 2. In this step, a set of sensitivity analyses is performed for each combination of n fault durations specified in the probability mass function $f(l)$ in order to derive the minimum inter-arrival time between bursts (T_E) under which the task set is still schedulable.

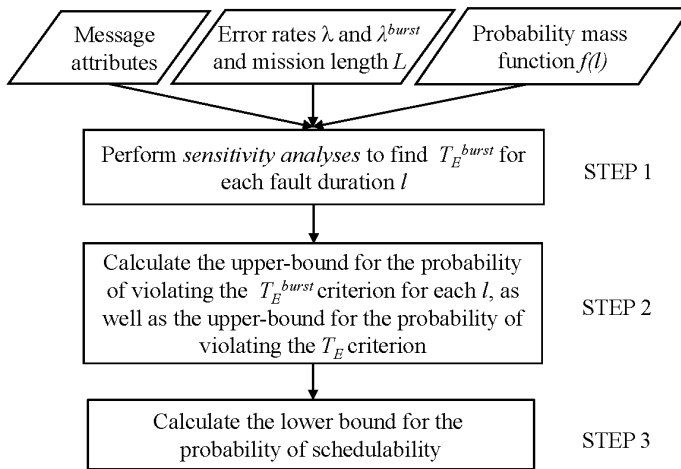


Fig. 1. Task scheduling – Methodology overview

STEP 3. The goal of this step is to derive the probabilities that the actual inter-arrival times between bursts will not be shorter than the calculated minimum inter-arrival times by taking into account λ and the mission time L .

STEP 4. Finally, based on the probability mass function $f(l)$, as well as the derived probabilities for each fault duration combination, the cumulative probability of schedulability is derived.

Worst-Case Task Response Time Analysis under Error Bursts. In this subsection, a worst-case response time analysis is presented that identifies whether a given task set is schedulable when affected by faults of *random durations* and a minimum inter-arrival time T_E . One should note that if the fault duration is greater than or equal to the minimum inter-arrival time between bursts, every burst can start before the end of the previous one, hence the bursts can potentially affect the whole mission time. If this is the case or if the fault duration is greater than the minimum inter-arrival time of the task whose worst-case response time is to be calculated, the schedulability of this task cannot be guaranteed.

The main differences between the error characteristics in the traditional single error model and the burst model are:

- An error burst may consist of multiple errors
- An error burst may affect multiple tasks

Hence, the worst-case scenario required for calculating the worst-case response times is not the same in case of error bursts as compared to the model introduced in [6].

The set of bursts interfering with a task τ_i , i.e., arriving after the release of τ_i , is denoted by $\{\beta_j | j=1,2,\dots,n\}$.

Definition 1. The *worst-case error overhead* E_i^j for a task τ_i caused by an error burst β_j is the largest amount of time required by the task alternates τ_k^{alt} , $\tau_k^{alt} \in \Gamma$, to recover from the effects of burst β_j , in the interval between the release time of task τ_i and its completion.

Remark 1.1. An important observation is that, while the worst-case error overhead accounts for all the alternates required for recovery (including the successful ones), it excludes all the primaries, since, although affected by errors, those are already taken into account in the traditional part of the response time analysis [7,8], i.e., C_i and $\sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$.

Remark 1.2. Another observation following Definition 1 is that, in the general case, a burst causes its worst-case error overhead when its interference (i.e., overlap) with the executions of the first and last task it affects is minimized. Hence, it has to start ε before the completion of the first affected task, and end ε after the start of the last affected task.

The worst-case error overhead for task τ_i caused by an error burst β_j under a burst with length l_j is:

$$E_i^j = \max_{k \in hep(i)} (C_k^{alt} + \sum_{m \in hep(k)} C_m^{alt} + \alpha) \cdot \quad (1)$$

where

$$\alpha = \begin{cases} 0, & \text{if } k \neq h \text{ and } C_h - (l_j - \varepsilon) \geq C_h^{alt} \\ l_j - \varepsilon + C_h^{alt} - C_h, & \text{if } k \neq h \text{ and } C_h - (l_j - \varepsilon) < C_h^{alt} \\ l_j - \varepsilon, & \text{otherwise} \end{cases}$$

and τ_h is the highest priority task in the task set Γ and ε is an arbitrary small positive real number.

The total interference I_i experienced by a task τ_i is the sum of the maximum interference caused by the higher priority tasks, I_i^{hp} , and the maximum interference caused by error bursts I_i^{err} .

$$\forall \tau_i \in \Gamma, I_i = I_i^{hp} + I_i^{err} \quad (2)$$

Note that I_i^{hp} is given by the traditional response time analysis [7,8]:

$$I_i^{hp} = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (3)$$

Consequently, the worst-case error interference that needs to be accounted for, in the response time analysis, is obtained by the summing up the worst-case error

overheads, E_i^j , of each error burst β_j that is assumed to interfere with task τ_i 's execution. In this case, the maximum interference caused by the error bursts with a minimum inter-arrival time T_E on task $\tau_i \in \Gamma$ in the interval $(0, R_i]$ is:

$$I_i^{err} = \sum_{j=1}^{\left\lceil \frac{R_i}{T_E} \right\rceil} E_i^j . \quad (4)$$

Hence, the equation that gives the worst-case response time for a task τ_i under error bursts is:

$$R_i = C_i + I_i^{hp} + I_i^{err} . \quad (5)$$

Probabilistic Schedulability Bounds. We assume that, during a mission, if the actual shortest interval between any two error bursts W is less than the derived minimum inter-arrival time between errors T_E , then the task set is unschedulable. Hence, the probability of schedulability for a T_E value derived for a fault duration combination $Pr(Ulcombo_i)$, is equal to $Pr(W < T_E)$, i.e., the probability of schedulability of a given task set is translated to the derivation of the probability that, during the mission time L , no two consecutive error bursts arrive with an inter-arrival time shorter than the derived T_E . Once the probabilities of schedulability (or the upper bound for the probability of unschedulability) for the T_E values derived for each fault duration combination is calculated, the probabilities of the fault duration combinations extracted from the probability mass function $f(l)$ are used to calculate the cumulative probability of schedulability.

2.2 PFTA for CAN Scheduling under Error Bursts

This subsection presents a schedulability analysis for real-time message scheduling on CAN, and a sensitivity analysis in order to derive accurate probabilistic schedulability guarantees for fault-tolerant real-time messages. The schedulability analysis presented in this subsection extends the existing CAN response time analysis [9,10,11,12,13] to cope with burst errors modeled with an improved accuracy that enables the specification of a range of new parameters including e.g., fault duration and intensity.

In this section a distributed real-time architecture is assumed that consists of sensors, actuators and processing nodes communicating over CAN. The communication is performed via a set of periodic messages. For the sake of generality, a message M_i is assumed to include N_i frames, hence the worst-case transmission time C_i of the message in an error-free scenario is:

$$C_i = N_i \times f^{\max} \times \tau_{bit} . \quad (6)$$

where f^{max} is the maximum frame size in number of bits, and τ_{bit} is the time it takes to transmit a single bit on CAN. However, the analysis presented in this section applies to the particular case of single frame messages as well.

While CAN communication is non-preemptive during the frame transmissions, messages composed of more than one frame can preempt each other at frame boundaries. Additionally, the non-preemptiveness of message frames may cause a higher priority message to be blocked by a lower priority message for at most one frame length, if the high priority message is released during the transmission of a lower priority frame. This priority inversion phenomenon can affect all messages except the lowest priority one, and only once per message period, before the transmission of the first message frame [14].

Each fault may affect the system for certain duration. Depending on the duration of a fault and the minimum inter-arrival time between errors within a fault, a fault can materialize into a burst of errors, only a single error, or no error at all during its length. However it is assumed that at least one error occurs during each fault exposure, since analysis assumes the worst-case scenario. For the sake of presentation, the term *error burst* is used for both error bursts and single errors. The duration of the faults is very much domain specific, and in this paper, it is assumed that the information regarding the probability distribution of the fault durations is available. Errors may occur any time during the fault as long as they satisfy the minimum inter-arrival time condition derived from the sensitivity analyses.

We assume that each error in each message frame is detected as soon as it occurs by the built in CAN error detection mechanisms and upon each error in a frame, an identical frame to the erroneous frame is scheduled for re-transmission following the error frame. Other error model related parameters that are assumed to be given are the rate that the observed system is hit by errors caused by independent faults λ and the mission time L of the system. This section assumes that at most one burst may hit any message instance hence T_E is equal to the largest period of all the messages in the message set.

Methodology Overview. The ultimate goal of this approach is to find the probability that the message set is schedulable under a given fault and error hypothesis. The methodology is outlined in the following steps, and illustrated in Figure 2.

- STEP 1. In this step, a series of sensitivity analyses is performed for each l in the probability mass function $f(l)$ in order to derive the minimum inter-arrival times of errors within error bursts, T_E^{burst} , for which the message set is guaranteed to be schedulable.
- STEP 2. In this step, first an upper bound for the probability of violating the minimum inter-arrival time requirement between errors within a burst, T_E^{burst} , for each fault duration l is calculated. Then, this probability bound on the *fault duration* is unconditioned and an upper bound for the probability of violating the minimum inter-arrival time requirement between errors within bursts under faults of random length, during the whole mission is derived. In this step, separately, an upper bound for the probability of violating the minimum inter-arrival time requirement between error bursts, T_E , during the whole mission is derived.

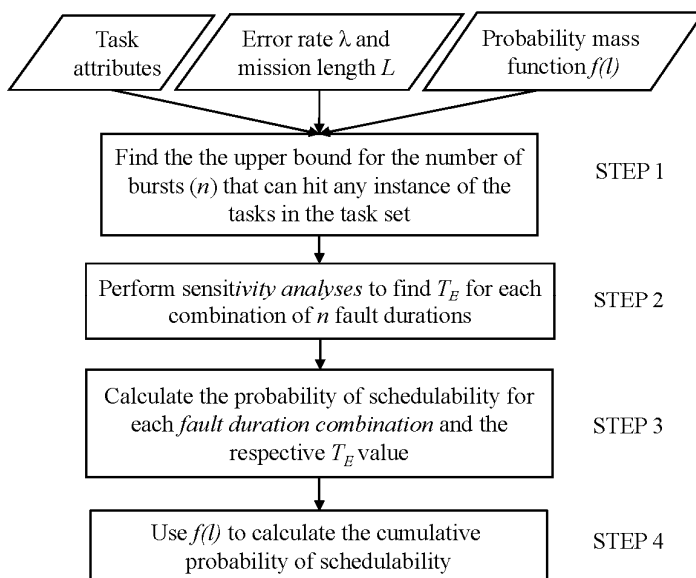


Fig. 2. Message scheduling – Methodology overview

STEP 3. Finally, in the last step, an upper-bound for probability of unschedulability, i.e. lower bound for the probability of schedulability, which is shown to be the union of the upper-bounds of the probabilities of at least one occurrence of any two bursts arriving less than T_E apart *or* at least one occurrence of any two errors within a burst, arriving less than T_E^{burst} apart during a mission of length L is calculated.

In the next subsection, a schedulability analysis under error bursts is presented which is the main tool to perform the outlined analysis.

CAN Response Time Analysis under Error Bursts. We present a worst-case response time analysis that identifies whether a given message set is schedulable when affected by error bursts caused by faults with a given duration l and a combination of error inter-arrival time thresholds (minimum inter-arrival time of error bursts T_E and errors within a burst T_E^{burst}). The presented worst-case response time analysis is based on the worst-case response time analysis of CAN under periodic messages and sporadic faults introduced by Tindell et al. [7]. In this work, we use the maximum error interference, I_i^{err} , in the equation used for calculating the worst-case response time of message M_i :

$$R_i = C_i + J_i + I_i^{err} + B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j. \quad (7)$$

Assuming the burst error model, the worst-case response time calculations will differ in the following cases depending on the minimum inter-arrival time of the errors within an error burst T_E^{burst} :

CASE 1. $T_E^{burst} \leq (e^{max} + f^{max})\tau_{bit} + l$: in this case, if the errors within an error burst occur with a separation of T_E^{burst} , it may not be possible to transmit any frame between any two consecutive errors during the burst. Therefore, the worst-case error overhead I_i^{err} in becomes:

$$I_i^{err} = (f^{max} + e^{max})\tau_{bit} + l. \quad (8)$$

The error overhead includes the transmission time of the largest frame in the worst-case scenario, i.e., when the first error in the burst hits its last bit. The other components of error overhead are the transmission time of the largest error frame and the whole duration of the fault, since in the worst-case, no frame can be transmitted during this time. The largest message frame and the largest error frame in Equation 8 are the frames before and after the error burst respectively.

CASE 2. $T_E^{burst} > (e^{max} + f^{max})\tau_{bit} + l$: in this case, one or more frames can successfully be transmitted between two errors within an error burst. Therefore only certain sections during the exposure to the fault may contribute to the error induced overhead. The worst-case error overhead, I_i^{err} , in this case, is given by:

$$I_i^{err} = (f^{max} + e^{max})\tau_{bit} + \left\lfloor \frac{l}{T_E^{burst}} \right\rfloor (e^{max}\tau_{bit} + (T_E^{burst} - e^{max}\tau_{bit} - \varepsilon) \bmod f^{max}\tau_{bit} + \varepsilon) + x \quad (9)$$

where $T_E^{burst} > 0$, $\varepsilon < \tau_{bit}$ and

$$x = \begin{cases} a, & \text{if } a \leq \left\lfloor \frac{l}{T_E^{burst}} \right\rfloor b \\ \left\lfloor \frac{l}{T_E^{burst}} \right\rfloor b, & \text{if } a > \left\lfloor \frac{l}{T_E^{burst}} \right\rfloor b \end{cases}$$

a and b are given by:

$$a = l \bmod T_E^{burst}$$

and

$$b = f^{max}\tau_{bit} - (T_E^{burst} - e^{max}\tau_{bit} - \varepsilon) \bmod f^{max}\tau_{bit} + \varepsilon$$

The error overhead in this case includes the transmission time of the largest frame, the largest error frame, and the error overhead *during* l . Note that in this case, the error overhead during l is strictly less than the fault duration l , however, Equation 9 is written in a general form and can be used for both cases. The first term $(f^{max} + e^{max}) \tau_{bit}$ in Equation 9 gives the worst-case error overhead caused by the first error in the burst and is equal to the sum of the largest message frame and the largest error frame.

The second term gives the worst-case error overhead caused by a single error during the burst (except the first error) multiplied by the maximum number of errors that can occur during the error burst minus one (the first error) assuming that the errors arrive with an exact inter-arrival time of T_E^{burst} . The product term $\lfloor l / T_E^{burst} \rfloor$ of the second term in Equation 9 gives the maximum number of errors that can occur during an error burst minus one. The product term $e^{max} \tau_{bit} + (T_E^{burst} - e^{max} \tau_{bit} - \epsilon) \bmod f^{max} \tau_{bit} + \epsilon$ of the second term includes the transmission time of the largest error frame and the largest message frame the error may hit. The last term x in Equation 9 gives the additional overhead caused by the errors whose relative arrival times are larger than (T_E^{burst}) . One should note that, the error overhead for a single error arrived with the minimum inter-arrival time (T_E^{burst}) , plus the additional overhead per error caused by late arrivals can at most be equal to $(f^{max} + e^{max}) \tau_{bit}$. Therefore, the worst-case value for x is equal to either the total amount of time that can be distributed to the error inter-arrival times for late arrivals (a), or the difference between the overhead assuming all errors hit the largest possible message in the last bit and the overhead assuming all errors arrive with the minimum inter-arrival time between errors within a burst $\lfloor l / T_E^{burst} \rfloor b$, whichever is smaller.

We have assumed that all successfully transmitted frames between two errors in a burst have the maximum frame size. If these frames are shorter than the maximum frame size, the error related interference may be larger than the value calculated by Expression 9. However, this increase in the error interference is bounded by the total sum of the differences between the actual frame sizes and the maximum frame size, i.e., the increase in the error interference is never larger than the cumulative reduction in the frame sizes. Hence, the worst-case response time analysis holds for the general case when message frame sizes are less than maximum, i.e., the analysis never calculates an optimistic value.

Probabilistic Schedulability Bounds. We assume that, during a mission, if the actual shortest interval between any two error bursts W is less than the minimum inter-arrival time between errors T_E , or if the actual shortest interval between any two errors within a burst W^{burst} is less than the minimum inter-arrival time between errors within a burst T_E^{burst} then the message set is un-schedulable. The T_E value is

assumed to be equal to the largest period in the message set and the T_E^{burst} value is derived for each fault duration in the probability mass function $f(l)$. Hence, the probability of unschedulability is equal to the union of two probabilities, (i) $Pr(W < T_E)$ and (ii) the probability of violating the minimum inter-arrival time requirement between errors within a burst caused by bursts of *random* length during the *whole* mission.

2.3 End-to-End Response Time Analysis

In this subsection we present a unified end-to-end response time analysis for fault tolerant distributed real-time systems consisting of tasks executing on nodes and exchanging messages over CAN network, under error bursts. The proposed analysis joins the results for processor scheduling with CAN message scheduling presented above, while taking into account the fault manifestations specific for each scenario as described in the subsections 2.1 and 2.2. Figure 3 illustrates the system model where tasks are executed on two nodes in an event-triggered manner under the FPS scheduling policy, and exchange messages on CAN network. Our goal is to determine the worst-case response time of an end-to-end *transaction* consisting of two tasks on different nodes exchanging one message on CAN.

The derivation of the end-to-end response time for a transaction is illustrated in Figure 4 where task *A* is executing on one node and sends a message *ml* to task *B* on a different node. What needs to be taken into account here is the *jitter inheritance* between nodes and network. In task (or message) scheduling, the *Response Time Jitter* - J_i is the maximum time distance between the response times for any two consecutive task (or message) instances τ_i^k and τ_i^{k+1} , and it is calculated as $J_i = \max |R_i^k - R_i^{k+1}|$.

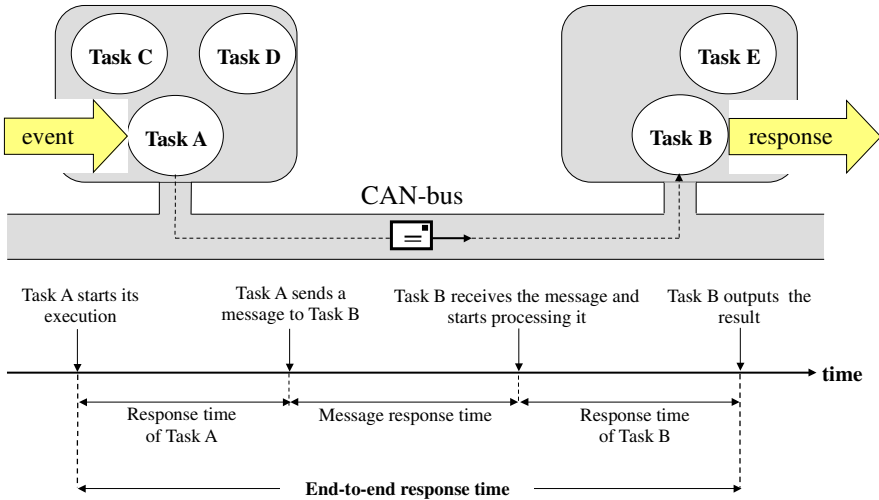


Fig. 3. System overview

Consequently, in the worst-case, the maximum response time jitter experienced by a task τ_i (or a message m_i) is given by:

$$J_i = R_i^{\max} - R_i^{\min}. \quad (10)$$

where R_i^{\max} is the worst-case response time of τ_i and R_i^{\min} is its best case response time given by [15]: $R_i^{\min} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{\min}}{T_j} - 1 \right\rceil C_j$.

Hence, the jitter inherited by the message m_l from the sending task A is $J_{m_l} = R_i^{\max} - R_i^{\min}$. Similarly, the jitter inherited by the receiving task B from message m_l is $J_B = R_{m_l}^{\max} - R_{m_l}^{\min}$. The error interference terms I_A^{err} , I_B^{err} and $I_{m_l}^{err}$ are derived as described in Equations 4 and 9 respectively.

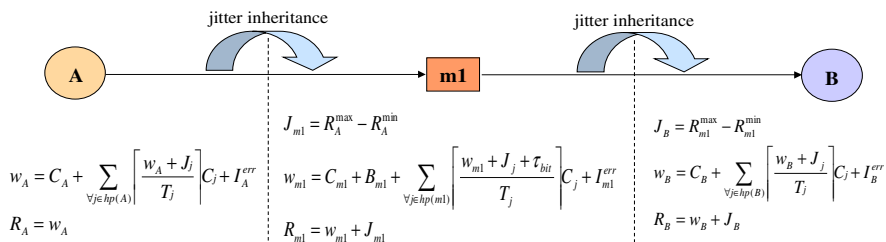


Fig. 4. End-to-end response time for one transaction A->m1->B

End-to-End Probabilistic Schedulability Bound. We assume that the schedulability of a task sets on separate nodes and the schedulability of messages on CAN do not affect each other's schedulability. Hence the end-to-end probability of schedulability of a transaction, $Pr(S_{transaction}) = Pr(S_{start-node}) \cup Pr(S_{CAN}) \cup Pr(S_{end-node})$, is equal to the multiplication of each individual probability of schedulability: $Pr(S_{transaction}) = Pr(S_{start-node}) Pr(S_{CAN}) Pr(S_{end-node})$.

5 Conclusions

Design of dependable distributed real-time systems demands advances in both dependability modeling and scheduling theory at node and network level in tandem, to provide system level guarantees that potential error scenarios are addressed in an effective as well as efficient manner. In this work, we have introduced a schedulability analysis framework for dependable networked real-time systems. We presented a sufficient end-to-end analysis that accounts for the worst-case interference caused by error bursts on transactions consisting of tasks scheduled on

different nodes under the preemptive FPS policy, and exchanging messages on CAN. The proposed analysis introduces significant improvements over existing works in many aspects, including a more elaborate and realistic error model that relaxes the previous assumptions. Further, we have outlined an overview on how to derive probabilistic scheduling guarantees from the stochastic behaviour of errors by performing a joint schedulability – and sensitivity analysis.

References

1. Stankovic, J.A., Ramamritham, K.: *Hard Real-Time Systems Tutorial*. IEEE Computer Society Press (1988)
2. Thekkilakattil, A., Dobrin, R., Punnekkat, S.: Preemption Control using CPU Frequency Scaling in Real-Time Systems. In: *18th International Conference on Control Systems and Computer Science* (2011)
3. Navet, N.: Controller Area Networks: CAN's use within Automobiles. *IEEE Potentials*, 12–14 (1998)
4. Pop, F., Baron, O.-A., Cristea, V.: Rescheduling Services for Reliable Distributed Systems. In: *18th International Conference on Control Systems and Computer Science* (2011)
5. Casparsson, L., Rajnak, A., Tindell, K., Malmberg, P.: *Volcano – a Revolution in On-board Communication*. Volvo Technology Report. 98-12-10 (1998)
6. Burns, A., Punnekkat, S., Strigini, L., Wright, D.R.: Probabilistic Scheduling Guarantees for Fault-Tolerant Real-Time Systems. In: *Dependable Computing for Critical Applications*, pp. 361–378 (1999)
7. Audsley, N.C., Burns, A., Richardson, M.F., Tindell, K., Wellings, A.J.: Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal* 8, 284–292 (1993)
8. Joseph, M., Pandya, P.: Finding Response Times in a Real-Time System. *The Computer Journal - British Computer Society* 29, 390–395 (1986)
9. Tindell, K., Burns, A., Wellings, A.: Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice* 3, 1163–1169 (1995)
10. Broster, I., Burns, A., Rodriguez-Navas, G.: Probabilistic analysis of CAN with faults. In: *23rd IEEE Real-Time Systems Symposium*, pp. 269–278 (2002)
11. Broster, I.: *Flexibility in Real-Time Communication*. Department of Computer Science, University of York (2003)
12. Navet, N., Song, Y.Q., Simonot, F.: Worst-Case Deadline Failure Probability in Real-Time Applications Distributed over Controller Area Network. *Journal of Systems Architecture*, 607–617 (2000)
13. Many, F., Doose, D.: Scheduling Analysis under Fault Bursts. In: *IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 113–122 (2011)
14. Di Natale, M.: Scheduling the CAN Bus with Earliest Deadline Techniques. In: *IEEE Real-Time Systems Symposium*, pp. 259–268 (2000)
15. Bril, R.J., Steffens, E.F.M., Verhaegh, W.F.J.: Best-Case Response Times of Real-Time Tasks. In: *2nd Philips Workshop on Scheduling and Resource Management*, pp. 19–27 (2001)

Improvements of Instruction Scheduling

Bogdan Ditu¹ and Nicolae Tapus²

¹ Freescale Semiconductor Romania, Tudor Vladimirescu 45,
Bucharest 5, Romania
bogdan.ditu@freescale.com

² “Politehnica” University of Bucharest, Computer Science Department,
Bucharest 6, Romania
ntapus@cs.pub.ro

Abstract. Instruction Scheduling as a compiler optimization is a very powerful technique to enable instruction level parallelism for many types of modern architectures. Instruction Scheduling can be used for making best fill of the micro-architecture pipeline (by minimizing the number of pipeline stalls) and is also of critical importance for keeping as busy as possible the multiple execution units for the architectures with parallel execution sets (like VLIW, EPIC or VLES architectures). This paper will present a set of improvements that can be brought to an instruction scheduling technique implemented in a real compiler for a VLIW architecture, where both pipeline aspects and multiple execution units are exploited. The improvements are based on practical and theoretical observations. They include a possible false-WAW dependence improvement, another improvement by considering inter-block latencies and also some improvements about hyper-block scheduling and IF-conversion integration with instruction scheduling.

Keywords: instruction scheduling, IF-conversion, inter-block latencies.

1 Introduction

Modern architectures are offering hardware support for increased instruction level parallelism. Increasing the hardware support for instruction level parallelism is only one side of the problem, the other side remaining how to make the code to take full advantage of the available parallelism. To achieve that, a supplementary mechanism is necessary. This mechanism is the instruction scheduling. Instruction scheduling can be implemented either as a hardware mechanism (optimizing the code while the application is in execution) or as a compiler technique (optimizing the instruction execution order during compile time). There will be no argue about which is better to use, as this is a decision taken early in the development phase of a hardware architecture and the dynamic and static scheduling do not usually apply together.

The instruction level parallelism that is commonly exploited by Instruction Scheduling as a compiler optimization is based on two architectural aspects:

- the micro-architecture (pipeline) of the processing unit - mechanism for executing in parallel more instructions on a single functional unit, each instruction being in a different stage of execution
- multiple functional units in the processing unit - mechanism to enhance instruction level parallelism by making possible the execution of multiple instructions in the same time on different execution units

The micro-architecture pipeline can negatively affect the executing code, by adding stalls in the execution, stalls that are considered "dead times" in code execution. These stalls can appear if an instruction is executed in a pipeline stage and its result is not available in the next pipeline stage, but in a later one. The distance between the pipeline stage the instruction is executed and the pipeline stage the instruction result is available is characterized as a pipeline stall, and it can be also called the instruction latency (for example, if by executing an instruction in the pipeline stage 1, its result is available only in the pipeline stage 4, meaning that for 2 cycles the pipeline is in stall, and that instruction adds a 3 cycles latency between it and an instruction that might need its result). In this situation, the instruction scheduler can rearrange the code instructions to minimize of pipeline stalls, meaning that while an instruction waits for its result, other instructions can be used to fill the stalls.

For multiple execution units, the more instructions can be grouped to be executed in parallel, the better. Building the instruction words on a VLIW architecture is exclusively the job of the instruction scheduler.

The rest of the paper will focus on presenting the improvements brought to instruction scheduling (section 3), after a brief presentation of instruction scheduling approaches (Related Work - section 2). The results of the improvements and some comments on these results are presented in section 4. Section 5 will add some concluding remarks on this work.

2 Related Work

In the past, instruction scheduling as a compiler optimization, was a very popular research subject. In the last decade it got even more interesting as the embedded systems became more eager for such an optimization in order to exploit parallel aspects of the architectures. As the instruction scheduling problem is NP-hard, many algorithms and heuristics were researched during the time, many of them having application only to specific types of architectures.

It is very important for the reader to know that this paper is continuing and slightly extending our previous work on instruction scheduling improvements ([1]). After a re-iteration of our conducted previous work, there will be a short overview of the extensions of this work so far and possible future work.

There are two major directions when considering the instruction scheduling: the general techniques (the ones that can be applied with success on any kind of region) and the loop region techniques (which are focused only on loops scheduling). The latest techniques were developed because the loops are in a program the

regions that take the most execution time, so the best improvements can be obtained from carefully scheduling one loop.

Each of the techniques mentioned here have different scopes: ones can be applied only at basic-block level, others can be applied to some increased types of regions, and others are global, being applied to the whole function to be optimized. The most used and the most cited by compiler literature instruction scheduling techniques are: list scheduling ([2], [3]) (for basic-block, or larger regions), trace scheduling ([4]) (for larger regions and fragments of flowgraph), percolation scheduling ([5]) (for cross-block scheduling) and software pipelining ([6], [7]) (for loop regions).

There are also many modern studies and research for improved instruction scheduling from which it worth mentioning [8], [9] for advanced remarks regarding instruction-level parallelism, [10], [11], [12], [13], [14], [15], [16], [17] for different types of heuristics and approaches for improving the instruction scheduling results for different purposes and scopes.

One of the most popular algorithms is list scheduling ([2], [3]). Due to its simplicity, applicability on basic-block level, and good results in practice, it was adopted in many compilers, especially in production compilers, which usually avoid dealing with experimental techniques (more used by prototype and academic compilers). The algorithm we propose for improving is also based on the list scheduling technique and is detailed as follows.

3 Improvements of Instruction Scheduling

There will be a short presentation of the used instruction scheduler, followed by the improvements details and what other opportunities of improvements were detected and will be implemented (kind of future work) and finally the results of these improvements. Starting from an implementation of this technique in a real compiler for a VLIW architecture (Freescale Starcore[®] DSP family of processors), some improvement observations were explored. This paper will further focus on presenting the improvements and analyzing their impact on real-life situations.

3.1 *Algorithm to Improve*

The improved instruction scheduler is an implementation of the classic local List Scheduling algorithm which works both on basic-block level and hyper-block level. This implementation was also adapted to exploit both the pipeline aspects and multiple execution units. The algorithm had good results for basic-block level, but there were still some opportunities to improve the results. At hyper-block level the algorithm did not have spectacular results, many of the test-cases having worse results than the basic-block scheduling. At the hyper-block level were the best opportunities to improve the instruction scheduler by integrating better the IF-CONVERSION (the hyper-block construction phase) and the instruction

scheduling. For hyper-block improvements, previous work was conducted in [18], [19]. Further hyper-block improvement opportunities are also detailed later on in the paper.

The instruction scheduling algorithm is not so complicated: first of all, the dependence directed acyclic graph (DAG) ([2]) is built to allow the analysis of inter-instructions constraints. The most important dependences (the ones that are of interest for the local instruction scheduling algorithm) are the data dependences. The dependences considered by the dependence DAG are the classical ones: *Read-After-Write (RAW)*, *Write-After-Read (WAR)*, *Write-After-Write (WAW)* and in volatile cases the *Read-after-Read (RAR)* dependences. The used local list instruction scheduling algorithm is presented in Fig.1 and will be detailed as follows.

```

compute dependence DAG of the instructions
mark all instructions in the list as unscheduled
CurTime = 0
while there are unscheduled instructions do
    get ready instructions
    if there are ready instructions
        create best packet
    endif
    if a best packet was created
        add best packet in scheduled packet list
        mark instructions in packet as scheduled
    endif
    increase CurTime according with the packet
enddo

```

Fig. 1. The local list instruction scheduling algorithm

The local *List Instruction Scheduling Algorithm* computes the dependence DAG, marks all the instructions as unscheduled and then initiates the current time *CurTime* to zero. The algorithm works as simple as possible: for each time slot (starting from zero and ending after all the instructions have been scheduled), the algorithm finds out which are the instructions that are ready to schedule - all their predecessor instructions (in the dependence DAG) are already scheduled and all the latencies that keeps one instruction to be scheduled are passed. After finding out which instructions are ready at a certain time (that moment's *CurTime*), the algorithm tries to build the best packet containing the instructions that are ready at that certain time. After building the best packet, it marks all the scheduled instructions as scheduled, adds the best packet to the list of scheduled packets and then proceeds to the next iteration (finding ready instructions and so on, even if not all the ready instructions at this particular time were scheduled). If at a certain time there are no ready instructions there is a pipeline stall and the *CurTime* is incremented successively until the pipeline stall is overcome. There are several things that can be considered when creating the *best packet*: the most convenient situation is to have all ready instructions fitting in this best packet, but this is not always the case. The "best" solution should be explored: the first criterion that

comes in mind is that the instructions inducing the longest latency to be scheduled at this time in the detriment of the instructions that have reduced latencies, but the "best" solution is probably a very difficult problem to solve (in the whole context), so several heuristics apply at this point.

3.2 False-WAW Improvement

The first of the improvements is based on the scheduled code analysis. While analyzing the scheduled code there were spotted instructions that could be grouped to execute in parallel, but they were not. At a closer look, after going through the dependence analysis results and after analyzing the instruction scheduling algorithm step by step, we found the thing that was preventing these instructions to be grouped for parallel execution. More instructions that could be grouped in parallel wrote the carry bit (since those instructions results could had overflow). Two instructions writing the carry bit is resulting in a *Write-After-Write (WAW)* dependency, which is correct. The algorithm was working fine, the problem being that this case was omitted to be analyzed when created the dependence DAG. One can argue on what is wrong in not grouping instructions with *WAW* dependency. There is nothing wrong, it is a must to block *WAW* dependency in the same execution set. Nevertheless *false-WAW* should not lock some grouping opportunities (Fig. 2 and Fig. 3).

The carry bit situation looks like a *false-WAW*, since there are truly two carry bit writes, one for each instruction, but these writes are not used anywhere. If there is at least one instruction that uses any of the carry bits, then this dependency would really be important. The use of the carry bit must come from its right definition, but if no instruction really uses the carry bit, then it does not matter which of the carry bit definitions will be available after scheduling both instructions.

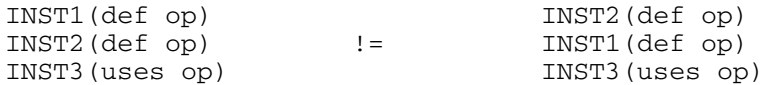


Fig. 2. True WAW dependence (instruction order must be strictly kept to have correct results)

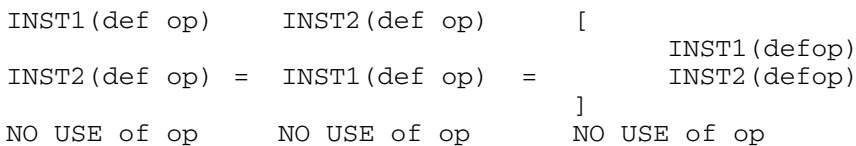


Fig. 3. False-WAW dependence (instruction order is not important, they can even be grouped in parallel)

From this observation to the point where two such instructions are decided to be grouped in the same execution set, there are lots of other constraints that must

be passed. When deciding that a *WAW* is not really a *WAW*, but it is a *false-WAW* (based on the presented observation), this *false-WAW* allows the instructions interchange, they order does not matter anymore (of course if there are no other dependences induced by other operands). But from the relative position freedom to the moment when these two instructions can be grouped, there is another hardware architectural step: the target architecture must allow multiple writes of the same operand in the same execution set, and must describe the exact behavior in this particular case (i.e. which of the writes will be considered). If this behavior is well defined, the *false-WAW* is less - only one of the writes should not be used anywhere. By grouping those two instructions, special care must be taken to allow the right instruction to write the needed operand. If the target architecture has an unspecified behavior when there are multiple writes in the same execution set, the *false-WAW* must remain a regular *WAW*. This optimization can cause several hardware damages if the hardware is not protected for this kind of behavior.

In the architecture the improved compiler was used for, there were two kinds of behavior (which is probably the case of most architectures):

- a register could not be ever written more than once in the same execution set - ignoring this constraint can cause unknown behavior in the hardware, possible leading to a hardware malfunction or even a chip burn
- the architecture allow multiple writes of flags in the same execution set, with the specification that the hardware has no clear behaviour about which of the instructions really affects the flag.

This was exactly the expected solution: the carry bit (the point from where all this observation started) is a flag, so there can be multiple writes to the carry bit, resulting in the possibility to group in the same execution set more instructions that writes the carry bit. As previously mentioned this observation can be refined and allow multiple writes, without the constraint for both definitions to be used, but just for one of the definitions to be used. This is applicable if and only if the target architecture has a clear way to treat this cases and its behavior is well known. For example, the target machine can state that multiple writes are allowed, but just the last instruction in the execution set is the one that really performs the write - this allows us to place the last instruction in the execution set the instruction for which the definition is used.

The results of this improvement are really spectacular (as will be shown in the results section), and the functionality of the improved code is the same, which validate the improvement as itself and the improvement's results as well.

3.3 *Inter-Block Latencies Improvement*

The second improvement of Basic-Block Instruction Scheduling is based on a theoretical observation. The presented basic-block instruction scheduling technique could take in consideration the effect of the scheduled block in the global context, without increasing the algorithm complexity and assuming that the scheduled block affects only its immediate neighbors.

For example, when scheduling a block, if all its predecessors are completely ignored, the scheduler can consider as ready instruction, an instruction that has latencies from the instructions in previous blocks, resulting in a schedule that begins with stalls. A first solution to keep the local instruction scheduling aware about the context it integrates in, is to consider the effect of the previous blocks in the current block. How can this be achieved? When starting to schedule a new block, for each instruction in that block, its maximum latency with any of the instructions in the predecessors (already scheduled) blocks is computed. The fact that the predecessors blocks are already scheduled helps to establish the exact latency and of course to establish how far up (starting from the bottom of the predecessor) to search the latency. It's well known that the maximum latency is the length of the pipeline, so, when searching (in bottom-up order) the instruction with possible latencies, the search is stopped when the maximum limit is reached (including the branches latencies).

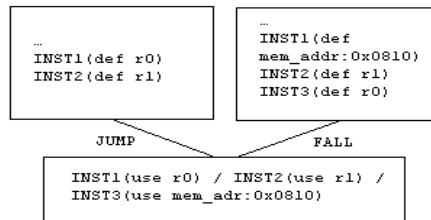


Fig. 4. Inter-block latencies

To clear things up, let's consider the example in Fig 4. The instructions in the block to be scheduled are instructions that use the registers defined in both of the predecessors (from the predecessors we consider only the instructions that are possible to affect the instructions in the successor block). In the figure are also marked the type of edges between the predecessors and the block to be scheduled. Let's consider that the following latencies are considered: the JUMP takes 3 cycles, the FALL takes 0 cycles, the distance between a register definition and a register use is 2 cycles and the distance between a memory location is written and the memory location can be read is 6 cycles. In this case, we start computing the latencies from predecessor blocks: for INST1 in the block to be scheduled - there is no latency between it and the instruction in the JUMP predecessor because the register-to-register latency is overcome by the other existing latencies (the JUMP latency and the distance between the def instruction and the bottom of the predecessor). There is a latency of 1 cycle between the INST1 in this block and INST3 in the FALL predecessor, meaning that if INST1 will be scheduled in the first cycle of the current block it will have a stall of one cycle while waiting for the

result of INST3 in the FALL predecessor. For INST2 in the current block, there are no dependences with the JUMP predecessor, same reasons as INST1. With the FALL predecessor there are no dependences, too, because the 2 cycles of register-to-register latency are overcome by the distance between the INST2 in the FALL predecessor and the bottom of the FALL block. For INST3 in the current block, there are no dependences with the JUMP predecessor, since the JUMP predecessor does not have any writes to the specified memory location and has a 3-cycle latency with the FALL predecessor, because from the 6 cycles that should pass from when the memory is written until it can be read are passed only 3 cycles (i.e. if INST3 from the current block will be scheduled anywhere less than 3 cycles from the beginning of the current block it will be stalled to have the available result from INST1 in FALL predecessor).

Next thing to decide is how one can consider using these computed inter-block dependences (or to be more correct the inter-block latencies). There are at least three solutions to this problem.

The first solution (the one that was also implemented as improvement) is to consider all the latencies as real latencies and when getting the ready instructions to consider these inter-block latencies as normal latencies, allowing an instruction to be scheduled only after its latency with the previous blocks is passed. At the first look this should be the real solution to the problem, but in practice the things did not look that well: as will be presented in the results section, this solution had some real improvements (not spectacular results, but improvements) and also had some worse results, too. Investigating the worse results the following conclusion was elaborated: if the latencies of each instruction from the current block with all the predecessors are computed and the maximum is chosen, with no regards which are the maximum latencies from each previous block, the following situation can happen (Fig. 5).

In Fig. 5 there is an example of worse case of inter-block latencies. This example is similar with the previous one (Fig. 4), so the latencies will be presented directly, without any detailed analysis. INST1 in the current block has 5-cycle latency with INSTn in the FALL predecessor block and no latency with any instruction in the JUMP predecessor block. If we consider just the maximum latency from all the predecessors and do not count the maximum latency from each predecessor, we will consider INST1 in the current block as ready after a first cycle in CurTime is passed. At a first look it really seems that this is a great deal and the 5-cycle stall is overcome by this new schedule (which, looking just to this block is worse than the schedule when not considering inter-block latencies). When looking on the resulting scheduled code off-line it looks better this way than before (as a global aspect), even if locally the schedule looks worse. The real problem was revealed when the code was executed and the profile trace was analyzed: the FALL edge was passed way fewer times than the JUMP edge, resulting in the execution of the worse schedule more many times without any effect from the FALL predecessor, so the performances decreases are justified.

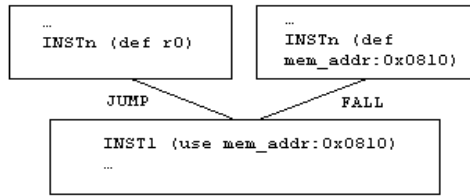


Fig. 5. Worse case of inter-block latencies

To repair this disadvantage, **the second solution** (which remains to be implemented as future work), needs to consider for each instruction, the maximum latency with each of the predecessors. When considering the instruction as ready to monitor the latency induced by each of the predecessors, each of them weighted by the frequency the predecessor has. The solution seems to be reasonable better than the previous one with the consideration that the frequency of a block is pretty hard to establish and it is usually information known just after the profile. It is true that the simplest way to have details about profile information is the compiler user giving hints to the compiler by placing pragmas for profile count, but as a real-life experience it is well known that most of the compiler users won't pass that information to the compiler, even if this information would help a lot the overall compiler result. There are for sure other ways to estimate the profile count for a block, but just to estimate because real information needed for profile counting is most of the times available only at runtime. This second solution is really the best solution, if and only if the profile count information is available. The work in [20] can really be used for implementing this solution.

The third solution for the inter-block latencies problem is a solution that does not provide the optimum solution but it is a middle solution. It tries to take advantage of the inter-block latencies information that is already computed. In this method (which also remains to be implemented as future work), when establishing which instructions are ready (at the beginning of the block to be scheduled), the scheduler should not wait for an instruction to have the inter-block latency passed, but just take into account when choosing instructions for the best packet that one instruction has a "possible" latency (possible, because the latency is not certified, since we don't know for sure if the latency happens most of the time or not). The scheduler should then consider another instruction that does have no latencies over this one. Theoretically, this should result in a better scheduling results than without considering the inter-blocks latencies at all, and probably it should result in a near to the previous solution scheduling.

This is for sure an interesting aspect of the inter-blocks interaction (of any kind), but especially when it comes to instruction scheduling, so this last solution will be for certain looked into in the near future, and the results of its implementation will be for certain of real interest for "optimal" results versus "complete" solutions.

3.4 *Improvements on IF-Conversion and Instruction Scheduling Integration*

When using block instruction scheduler, one of the best advantages is their reduced complexity, because they try to solve the scheduling problem locally, ignoring the global impact of the scheduling in the program's context. The local schedulers are simplified by the fact that they do not have to deal with control flow constructions, their only concern being what is happening in the block to be scheduled. The quantity of instructions to be scheduled is not really an issue for the local schedulers and does not affect their complexity (just the computational time), their complexity remaining most adequate compared with the global schedulers. Nevertheless, the larger the number of instructions in block, the better the scheduling solution, because there is an increased knowledge when deciding what and how something is scheduled. To combine the performances of local schedulers with the fact that it works better on larger blocks (with more instructions), and to overcome the fact that the control-flow path are of no interest for local instruction scheduling, there must be found a solution to build larger blocks by collapsing several control flow paths (and their blocks of course) into a single block. The best solution to do this is to eliminate branches, to replace conditionally branches tests with simple compare instructions which sets a certain predicate, and then to predicate all the instructions trough that branch with the predicate set by the compare instruction. This technique is called IF-conversion and it is used with target architectures that have predication support in order to improve some optimizations (which work better on larger blocks, without affecting their local scope) and especially to improve the instruction scheduling.

Both Instruction Scheduling and IF-conversion optimizations are very powerful and useful optimizations in a compiler. There are, anyway, some problems that can come from IF-conversion if the blocks to be IF-converted are not chosen right, resulting in an increased execution time. Why the IF-conversion and Instruction Scheduling are somehow related? The answer is pretty simple: the Instruction Scheduling can benefit from the IF-conversion result, unlocking lots of scheduling opportunities by transforming the control dependences into data dependences and by enlarging the Instruction Scheduler candidates; the IF-conversion for sure can benefit from the Instruction Scheduler, because the scheduler is the one that can best estimate the outcome of IF-conversion, and based on its estimation the conversion transformation can be applied or not.

There were some studies on combining Instruction Scheduling and IF-conversion ([18], [19]), with good practical results, but they did not look like have exploited to the maximum this combination of optimizations. Starting from these studies, further improvements of IF-conversion and instruction scheduling integration were sought. Based on the limitations and future work presented in [19] we tried to check how a better strategy would work, helped by profile information and probabilities of the branches in IF-THEN-ELSE situations.

The strategy we adopted was to add branch probabilities to the estimation of costs for the hyper-block construction. Instead of using exact profile information, we used a flavor of profile information which provided only information about the

branch that is most likely to be taken after each IF block. Even though this is not the best that can come from this solution, the results are encouraging for going further with this strategy on exact profile information.

4 Instruction Scheduling Improvement Results

All the presented improvements were implemented in an experimental compiler for Freescale Starcore® DSP. The results were measured using compiler specific benchmarks and DSP applications. To measure the real improvements, complete performance measurements were done before starting to improve the instruction scheduling, performance results which are the witness of the improvements that were really made by using the proposed strategies and observations. By complete measurements, we understand measurement of runtime performances using the basic-block instruction scheduler (*SCHED1*), using the hyper-block instruction scheduler from related work (*SCHED2_REL*), and using the new hyper-block instruction scheduler strategy (*SCHED2_NEW*). We also measured performance using a combination of basic-block instruction scheduler (*SCHED1*) and loop instruction scheduler (software pipelining) - which also can benefit from the presented improvements (in the performance reports we marked this combination as *ALL*).

4.1 Improvements Using Legal Multiple Writes in an Execution Set

These improvements are really impressive, lots of test-cases got better instruction scheduling by just unlocking the false-WAW dependences that existed in code, allowing multiple writes in an execution set (of course for the operands that the target architecture really allows to be written in parallel).

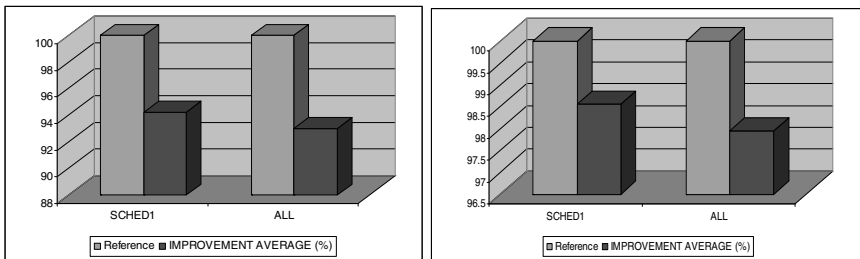


Fig. 6. (a)Benchmark1 and (b)Benchmark2 improvements using multiple writes in an execution set

Even if the improvements graphics are not scaled (Fig. 6 a,b), the average improvements are noticed, and if one must be convinced that these improvements are really good, the maximum cases for every benchmark could do the difference. Both the average and the maximum situations for the run benchmarks are presented in Table 1.

Table 1. Comparative results for using the multiple writes in an execution set improvement.

	SCHED1 % improv	ALL % improv
Benchmark1(avg)	5.790624907	7.015413263
Benchmark1(max)	33.58745352	36.27797862
Benchmark2(avg)	1.161607084	1.607651383
Benchmark2(max)	40.98325462	63.33399721

The maximum values of the improvements are really impressive, an overall percent of 63% improvement, even if it is just on a benchmark, it really is impressive, especially for embedded compilers.

4.2 Improvements Using Inter-Blocks Latencies

As commented when presenting the improvement opportunity, there are chances for this optimization to generate poor improvements because the branch probabilities that are considered when scheduling are not accurate. As the results will show, there are improvements, too, on the second benchmark, the average itself is positive, so if this opportunity will be further explored as presented in the theoretical part (which was considered as future work), the chances to get better results are real.

As expected, the overall results are not encouraging, but still, we are confident that this improvement can really work if the inter-block latency will be taken in consideration, but not as a primary criterion, like it is in these tests. The purpose of these tests was to reveal that improvements can be achieved, but the method of considering inter-block latencies is not the right one. Also, there are improvements in some cases, as shown in Table 2.

Table 2. Comparative results for improvements using inter-blocks latencies

	SCHED1 % improv	ALL % improv
Benchmark1(avg)	-0.04772288	-0.03954684
Benchmark1(max)	0.37087028	0.00212012
Benchmark2(avg)	0.0014252	-0.00194498
Benchmark2(max)	0.42034468	0.21281169

4.3 Improvements on IF-Conversion and Instruction Scheduling Integration

For measuring the improvements on the new strategy used in the IF-conversion and instruction scheduling integration we used a combination of compiler specific benchmarks and real DSP applications as we were very interested in the impact of this strategy on real life situations. The measurements we performed were for the new hyper-block strategy (*SCHED2_NEW*) versus the related work hyper-block strategy (*SCHED2_REL*), both of the improvements being reported with regards to the basic-block scheduling on the same examples (*SCHED1*).

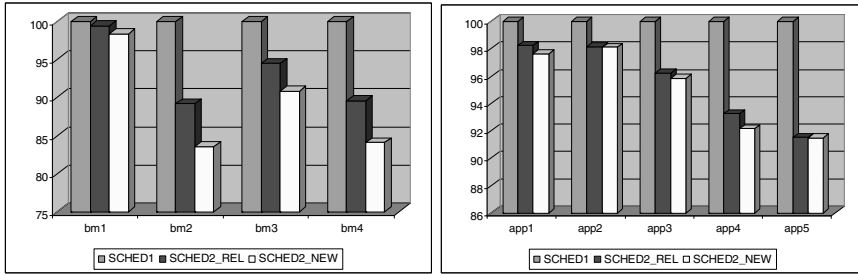


Fig. 7. (a)Benchmarks and (b)Applications improvements on hyper-block scheduling

Table 3. Comparative results for improvements on hyper-block scheduling

	SCHED2_REL improvement over SCHED1	% over SCHED1	SCHED2_NEW improvement over SCHED1	% over SCHED1
bm1	0.557117875		1.65747	
bm2	10.83623517		16.46412	
bm3	5.506216696		9.177028	
bm4	10.44618207		15.91385	
app1	1.768808448		2.357847	
app2	1.862348178		1.867658	
app3	3.780946073		4.156082	
app4	6.728061081		7.847981	
app5	8.482148124		8.502204	

The improvements we got over related work prove that the strategy of using the branch probabilities when deciding the costs of hyper-block constructions is a good step forward from the pessimistic approach. Although the improvements are not spectacular (Fig. 7 a,b and Table 3) the results may be determinant in deciding over the probability approach and even enhance it with exact profile information which probably would bring more benefits. Anyway an average of 2.18% and a maximum of 6.31% improvement (measured on the new strategy over the related work strategy) worth to be considered.

5 Conclusion

As mentioned in previous work ([1]), the results for basic-block improvements are very important in the context of compiler optimizations for exploiting parallelism. Even if in the case of inter-block latencies optimizations, the solution adopted is not always generating improvements, the fact that there are better cases proves that the optimization itself worth being explored further (probably by implementing the other two solutions presented as future work) – using the work in [20] for implementing the second solution is currently in progress. The improvements on IF-conversion and instruction scheduling integration are also important in proving

that branch probabilities can be considered with good results for hyper-block costs estimation. Even if the branch probabilities are not based on complete profile information, the results are motivating further investigations in this direction. Another interesting direction that worth being considered is having a retargetable implementation of the instruction scheduling algorithm and using architecture modeling in order to cover all target dependent aspects ([21]).

References

1. Ditu, B., Tapus, N.: Improvements of Instruction Scheduling. In: Proceedings of 18th Conference on Control Systems and Computer Science, pp. 545–552 (2011)
2. Muchnick, S.S.: Advanced Compiler Design and Implementation. Morgan Kaufmann Publishers (1997)
3. Gibbons, P.A., Muchnick, S.S.: Efficient Instruction Scheduling for a Pipelined Processor. In: Proceedings of the SIGPLAN 1986 Symposium on Compiler Construction. SIGPLAN Notices, vol. 21(7), pp. 11–16 (1986)
4. Fisher, J.A.: Trace Scheduling: A technique for Global Microcode Compaction. IEEE Trans. on Comps. C-30(7), 478–490 (1981)
5. Nicolau, A.: A Fine-Grain Parallelizing Compiler, Technical Report TR-86-792, Department of Computer Science, Cornell Univ., Ithaca, NY (1986)
6. Codina, J.M., Llosa, J., Gonzalez, A.: A Comparative Study of Modulo Scheduling Techniques. In: ICS 2002 (2002)
7. Rau, B.R.: Iterative Modulo Scheduling. An Algorithm for Software Pipelining Loops, MICRO 27, 63–74 (1994)
8. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers, Principles, Techniques, and Tools. Addison-Wesley (2007)
9. Chakrapani, L.N., Gyllenhaal, J., Hwu, W.W., Mahlke, S.A., Palem, K.V., Rabbah, R.M.: An infrastructure for research in instruction-level parallelism. In: Proceedings of the 17th International Workshop on Languages and Compilers for High Performance Computing (2005)
10. Malik, A.M., Russell, T., Chase, M., van Beek, P.: Learning heuristics for basic block instruction scheduling. *Journal of Heuristics* (2006)
11. Malik, A.M., McInnes, J., van Beek, P.: Optimal basic block instruction scheduling for multiple-issue processors using constraint programming. In: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (2006)
12. Malik, A.M., Russell, T., Chase, M., van Beek, P.: Optimal superblock instruction scheduling for multiple-issue processors using constraint programming. School of Computer Science, University of Waterloo (2006)
13. Russle, T., Malik, A.M., Chase, M., van Beek, P.: Learning Heuristics for Superblock Instruction Scheduling. School of Computer Science, University of Waterloo (2006)
14. Memik, G., Reinman, G., Mangione-Smith, W.H.: Precise Instruction Scheduling. *Journal of Instruction-Level Parallelism* 7 (2005)
15. Rangan, R., Vachharajani, N., Vachharajani, M., August, D.I.: Decoupled software pipelining with the synchronization array. In: Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques (2004)

16. Winkel, S.: Exploring the performance potential of Itanium processors with ILP-based scheduling. In: Proceedings of the International Symposium on Code Generation and Optimization. IEEE Computer Society (2004)
17. Parikh, A., Kim, S., Kandemir, M., Vijaykrishnan, N., Irwin, M.J.: Instruction Scheduling for Low Power. Department of Computer Science and Engineering, The Pennsylvania State University (2003)
18. Zane, A., Chircu, M., Costinescu, S., Palanciuc, V., Badea, D.: Integrated Instruction Scheduling and If-Conversion. In: International Signal Processing Conference (2003)
19. Ditu, B.: IF-conversion controlled by Instruction Scheduling. In: Proceedings of 15th Conference of Control Systems and Computer Science, pp. 706–711 (2005)
20. Ditu, B.: Automatic Optimizations based on Profile Information. In: Proceedings of Embedded World Conference (2010)
21. Ghica, L., Ditu, B., Tapus, N.: A Study of Architecture Modeling in the Context of Development Tools Chain. In: Proceedings of 18th Conference on Control Systems and Computer Science, pp. 308–313 (2011)

Re-scheduling Service for Distributed Systems

Florin Pop^{*}, Ciprian Dobre, Catalin Negru, and Valentin Cristea

University *Politehnica* of Bucharest, Computer Science Department,
Splaiul Independentei 313, Bucharest 060042, Romania
{florin.pop, ciprian.dobre, catalin.negru, valentin.cristea}@
cs.pub.ro

Abstract. The scheduling process in Grid systems is very complex. The resource heterogeneity, the dynamic environment, the variety of policies, the size and number of tasks, and the high number of constraints are some of the main characteristics that contribute to this complexity. Despite this, the increasing number of users and applications that use sustains its necessity. This thesis proposes a re-scheduling service for distributed systems, designed for Grid environment. The use of the proposed re-scheduling model in a real environment with real tasks and workflows represents the better way to validate the service architecture and algorithms. Also, a critical analysis of scheduling algorithms is performed.

Keywords: Re-scheduling, Distributed Systems, DAG, PBS, Grid.

1 Introduction

Scheduling in distributed systems means to take decisions involving resources distributed over multiple administrative domains. The main difference between a scheduler for distributed systems and a local one is that the first does not own his resources and has no control over them. For example, users do not know if other sets of tasks are planned on a resource which he considered. Therefore, distributed systems are a dynamic environment in which resources appear and disappear, which means that a scheduler must be able to detect and monitor resources.

Regarding the workflow in a distributed environment most common approach is that of modeling it as a simple task in a directed acyclic graph (DAG), where the order of the tasks execution, modeled as nodes, is determined by dependencies, which are modeled as directed edges. From these considerations a scheduling issue of a DAG is characterized by a combination of two steps, task assignment and scheduling (that decides the order for the assignment of tasks for each machine). These two phases can be referenced as a mapping of tasks to available resources. This will involve partitioning the application into interactive tasks, taking into account any control and data dependencies. The main purpose of scheduling is to construct an assignment of tasks on available resources and a scheduling

^{*} Corresponding author.

order to minimize computation and communication time (transfer data time) by properly allocating the nodes to the computing system without violating precedence constraints.

This paper proposes a re-scheduling service for distributed systems. In this context, distributed systems refer to multiple autonomous computer systems that interact with each other in order to achieve a common goal. The re-scheduling service is designed for Grid environment, a specific type of distributed system. In this project we focus on improving the service quality, respecting as much as possible the user demands. In Grid environment, although the failure rate is not high, reducing it might increase the client satisfaction and confidence. The overhead would be expressed in a reasonable time penalty for most users' point of view.

In centralized schedulers a single station is responsible for mapping a job to a resource. It only supports a uniform scheduling policy and is well suited for cluster management systems [1].

The decentralized model is characterized by multiple schedulers communicating with each other in order to make an informed decision of which resources to assign to the jobs about to be executed. Unlike the centralized model the decisions aren't made by a central component, thus making the system highly scalable and fault tolerant. On the down side, the scheduling solution might not be the optimal one since the information on the resources available is not centralized [2], [3]. A system based on a hierarchical scheduler is a system in which the processor is shared between several collaborative schedulers [4]. The hierarchical model is a hybrid one, combining the two described above. It defines a resource broker to communicate with the other schedulers. Thus each scheduler will still be able to enforce its own policy while the system will be aware of all the resources available leading to a better usage of resources [2], [5], [6], [7], [8], [9].

Secondly, scheduling algorithms can be classified as static or dynamic.

Static schedulers use a queue of jobs and their dependencies together with the resources available and make a plan of when and where the jobs should be run. All the characteristics of the parallel program (including its task execution times, task dependencies, task communications and synchronization) and the characteristics of the system (machines and networks specifications like CPU, memory, latency and bandwidth) are known *a priori* so the scheduling can be done offline [10].

Dynamic schedulers on the other hand, may make such a plan, but the final decision on the resource-job mapping will be done just before its execution. Other parameters like the current state of the system may be taken into account in a dynamic scheduler.

The most common way of developing a dynamic scheduler from a static one is to create the "plan" using a static scheduler repeatedly every so often. However creating a dynamic scheduler is different for every problem in turn.

Distributed systems are characterized by intensive data computing with large distributed datasets and distributed computing resources in cloud environment that can have long execution time. All of this causes a high probability of failures.

Building a trustworthy system is one of the main challenges of developers who are concerned with dependability issues. This is a vast domain that is in constant changes, regarding of the complexity of systems, the increasing number of uses, or

the nature of faults and failures. In such a dynamic environment, the need to deal with many types of threats is increasing. In nowadays, the computers are spreading fast into other domains, while the complexity of modern systems is growing, so that achieving dependability remains a central focus for system developers and users.

For start, we need to accept that errors will always occur in spite of all the efforts to eliminate faults that might cause them and try to reduce them as much as possible improving our fault tolerance techniques. From a fault-tolerance perspective, distributed systems have a major advantage. The way that they are designed makes them easy to provide redundancy, which is probably at the core of all fault-tolerance techniques. Although the rate of failures is low, about 1% of the computed tasks, reducing are important because it will improve the service quality, and might increase the client satisfaction and confidence.

The most important errors and also the most likely to happen in a Grid environment are the interaction errors and life cycle errors. So we focus our attention on them.

2 Fault Tolerance in Grid Environment

Fault tolerance is an important property for distributed programming in the context where resources within Grid reliability cannot be guaranteed. The easiest to detect errors, in the same time more frequent in such an environment are those related to timing, omission and interaction. Error detection will be done through monitoring of applications and nodes in the Grid by an error detection service. The easiest solution is data and application replication, in which, for example, data is sent for processing to more resources and then compared the results. Other options are to restart the application error handling from scratch, to save checkpoints of the application state periodically during execution without errors at which may return in the event of a defect in the last state saved. One of the most important components of a fault-tolerant system is the identification and categorization of the errors types that may occur in order to find clues about how it can be implemented or enhanced error detection and recovery from them.

2.1 *Types of Errors*

The implementation of error detection is very important to determine and analyze the types of errors that may occur. To identify these types, we could consider the following class:

- **Network errors** - are environmental errors caused by communication channel. This type of errors refers to package losses or corruption on the transmission path. In general, they are corrected by the network transmission protocol or the link is considered broken.
- **Timing errors** - are divided into two types of errors depending on the time of their appearance: at the beginning of the connection or during the

communication. The first category is due to the inability to establish a connection, and the second, which is considered a performance error, occurs when the response time exceeds the time in which the caller expects to receive a reply.

- **Response errors** - refers to errors caused by a service which returns values outside of the expected boundaries or that appear at the transition between system states.
- **Omission errors** - contains errors such as messages delayed or lost.
- **Physical errors** - includes critical conditions of physical resources like CPU errors, memory errors, storage errors. In this case, the resource is declared nonfunctional.
- **Life cycle errors** - can apply to component versioning. For example, updating a service while it's on. Clients expect that service to work properly according to its previous specification, but changes could affect it.
- **Interaction errors** - are caused by incompatibilities at the communication protocol stack level, security, workflows or timing. Because this conditions happen when the application is running and the context cannot be reproduced in testing phase, this type of errors are the most common one.
- **Byzantine errors** - are arbitrary errors that could appear during the execution.

2.2 *Failure Detection*

Fault detectors are a basic mechanism for fault-tolerant systems. Their need is evidenced in the process of activating recovery procedures or that allows reconfiguration of the system. They require additional messages in the system, resulting in increased network traffic that can lead to nondeterministic behavior of delays in the system. A simple algorithm for error detection, often used in practice, can be summarized as: at regular intervals, process p sends messages (heartbeat message) to process q; or if time expires before process q receives new message from p, q begins to suspect p.

Another solution is that the failure detector keeps a list of suspect processes and after a finite time, learning time, not to make any mistake, which defines in a certain extent a detector that uses a protocol adaptive too.

In order to analyze the efficiency of failure detection, some metrics were defined. These metrics evaluate the speed in fault detection and the detector ability to avoid mistakes. As an example we detail three of them:

- *Detection Time (TD)* is the time interval between the moment when the component fails and the moment the detector suspects it.
- *Mistake Duration(TM)* shows the time it takes to correct a mistake by the failure detector.
- *Mistake Recurrence Time (TMR)* measures the time between two consecutive mistakes.

2.3 Fault Tolerance Mechanisms

An extensive work in this field led to the description of fault tolerance mechanisms for preserving application execution despite of the presence of a resource, in particular a processor, failure. These mechanisms are classified into two major categories according the level at which errors are treated. The first one is at the *task level*, in which information about the task is sufficient to redefine the status of a failed task. The second one is at the *application level*, where much more information is necessary to redefine the entire state of application.

In the category of **task level mechanisms** we distinguish more strategies such as: retry checkpoint, alternate resource and task duplication. After detecting the failure the *retry approach* simply considers a number of tries to execute the same task on the same resource. The *checkpoint approach*, also called restart approach, saves the computation state periodically, such that it migrate the saved work of failed tasks to other processors. The *alternate resource strategy* chooses another resource for executing the tasks on the case of task failure. The *task duplication mechanism* selects tasks for duplication, hoping that at least one of the replicated tasks will finish successfully.

In the category of **application level mechanisms** we distinguish other strategies such as: rescue file, redundancy, user-defined exception handling and rewinding. The **rescue file mechanisms** consist of the resubmission of uncompleted portions of a DAG when some tasks failed. In this approach, if any task fails, the remainder tasks of the graph continue to be executed until no more forward progress can be made due to the dependencies in the DAG. At this point, it will be created a new input file, called a Rescue DAG that will contain the information about the all tasks of the DAG, but specifying the status - unfinished and successfully finished tasks. Then, the unfinished tasks are resubmitted. The *user-defined exception handling* allows users to give a special treatment to a specific failure of a particular task. The *rewinding mechanism* seeks to preserve the execution of the application. This is done by re-computing and migrate those tasks which is believed to disrupt the execution of following tasks.

3 Re-scheduling Service

This paper proposes a generic re-scheduling concept that allows the use of the designed re-scheduling algorithm with a wide variety of scheduling heuristics which are chosen in advance depending on the system structure. The re-scheduling component is called by a monitor who interrogates the system periodically.

Re-scheduling is more difficult to do on a Grid system than on a parallel system due to lack of control over resources. The operation of re-scheduling may require also significant delays that can affect overall system performance. Re-scheduling mechanism can divide into two categories: regular (periodically) or alerted by events. The mechanisms consist in the generation of regular requests for rediscovery of the available resources or recheck the availability of previously discovered resources. The other solution access immediately the re-scheduling mechanism

when is alerted by an event. Re-scheduling costs are bigger where there is data dependency than for independent tasks. This dependence can spread more easily an error that is not discovered in time.

To complete the workflow and to meet users requirements, re-scheduling during execution is required to meet the fault transparency and must be able to adapt to dynamic situations such as changes in resource availability due to an error. Generally, the key idea of re-scheduling policies for handling unexpected situations is to recalculate and adjust the order of execution of tasks. Therefore, re-scheduling mechanism is considered, in studies and analysis, not only for the occurrence of errors, but also to increase the performance in these systems. This is quite questionable because of the high cost that re-scheduling process brings in most cases. Generally speaking, re-scheduling adds an extra cost of scheduling and execution process. This cost can be explained by the cost of the reevaluations of the scheduling and the cost of transfer of tasks between the processing units. This cost can however be compensated by the fault transparency offered, if they occur, or lower cost of the execution, if the re-scheduling is used to optimize the initial scheduling (see Figure 1).

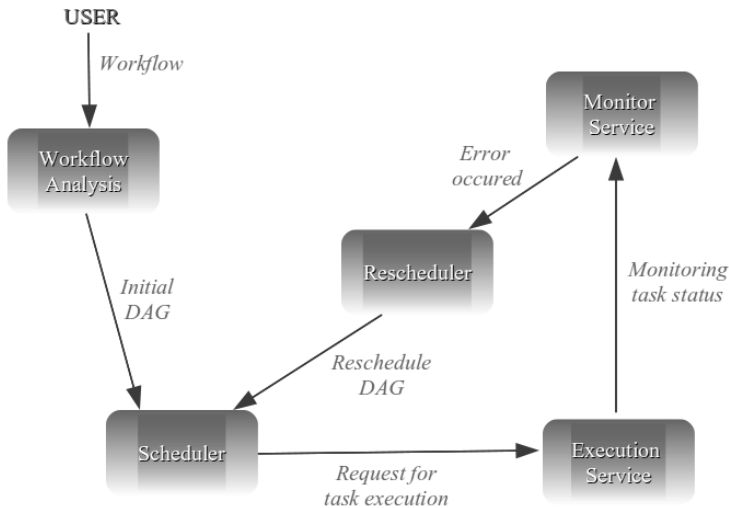


Fig. 1. Re-scheduling Service Architecture

The description of a scheduling request may specify some tasks requirements along with other additional information such as the task ID, path to the executable, the arguments, the input, output and error files, and the arriving time.

The requirements specified for each task may include:

- resource requirements such as CPU Power, Free Memory, Free Swap
- restrictions like Processing Time, Deadline Time
- number of executions that may occur
- priorities

The input for this service can be one task, a bag of tasks or a workflow of tasks. A bag of tasks is specified as a path to the directory where the configuration files are. A workflow represents a bag of tasks with dependencies. The input consists in a workflow file in a similar format with DAGMan, but simplified. This file is parsed and a directed acyclic graph is created to represent the tasks.

When all the information's are set, the schedule is called. First, the selected scheduling algorithm will provide nodes allocation and scheduling time to start. Then the executor will set a proper timer for each task and when the timer expires will establish the necessary context and will launch the task execution. Every time a task is send to compute, it is added to the monitor list.

Until it has successfully completed the entire graph, tasks are continuously monitored. When an error occurs at one of the scheduled tasks, it's coming out the necessity of re-scheduling it. Considering the task belonging to a directed acyclic graph (DAG) with dependencies between nodes we must analyze if the node where the error occurred can be reschedule alone or we should reschedule it with all the others nodes that forms together a dependency sub-graph having as root the current node.

Based on these assumptions, in terms of the set of tasks there are two types of input data for the re-scheduling process: one independent task and a sub-graph of tasks. Other input data that should be provided to the re-scheduling algorithm is the scheduling heuristic that should be used and the currently set of the available resources. After re-scheduling, the tasks execution order is reordered and new associations (task, processor, start time) are built and sent to execution in the same way as the initial schedule does. This represents output data that the re-scheduling algorithm returns. To easily insert the re-scheduling fault tolerance mechanism in any type of systems, we have designed a re-scheduling algorithm that can be used in combination with a wide variety of scheduling algorithm which are chosen in advance depending on the system structure (here we may consider factors like the number of existing processors, the structure of graph task that we want to schedule) to achieve optimal results. The proposed generic re-scheduling algorithm is described below:

```

H - heuristic used for re-scheduling
S - schedule
R - available resources

1. initialize schedule Scurrent a initial
   schedule of DAG
2. while (DAG unfinished)
   if (error detected)
   update R
   S = schedule(Scurrent, H)
   if (Scurrent:task_asoc_to_res !=
       S:task_asoc_to_res)
       Scurrent = S
   Execute Scurrent

```


The schedule procedure calls one of the scheduling algorithm (HLFET, CCF, ETF, HybridRmapper) to schedule the graph section remained unfinished, pointed by the heuristic H. At first, the characteristics of the system and the dependencies of the graph of tasks choose the appropriate heuristic and then, if an error occurs in the system, the re-scheduling procedure is called and the best heuristic is called on a particular subset of tasks. In this project we analyzed the following heuristics:

- CCF - a dynamic list scheduling algorithm providing good load balancing.
- ETF - an algorithm that aims at keeping the processors as busy as possible.
- HLFET - an algorithm based on both lists and levels scheduling heuristics.
- HybridRemapper - an algorithm specially designed for heterogeneous environments.

4 Resource Management Tool

This section presents the analysis of two important existing Grid scheduling tools: Condor and PBS. These are resource management tools.

Condor is a specialized resource management system developed for intensive computing tasks, which provides a task queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. The usage is as simple as possible, users submit their tasks to Condor, and it places them into a queue, chooses when and where to run the specific tasks based upon a policy. Also it deals with monitoring their progress, and finally informing the user upon completion. Condor can be used to manage a cluster of dedicated compute nodes. To use Condor inside a Grid environment Condor-G is needed because the Globus toolkit is used as the bridge between sites.

Another resource management tool is PBS, which offers Torque as an open source resource manager that provides control over batch jobs and distributed compute nodes. Torque is a centralized system, in which a controller is responsible for the system-wide decision-making and for estimating the state of the system. The controller mediates access to distributed resources by discovering suitable data sources for a given analysis scenario, suitable computational resources, optimally mapping analysis tasks to resources, deploying and monitoring task execution on selected resources, accessing data from local or remote data source during task execution and collating and presenting results.

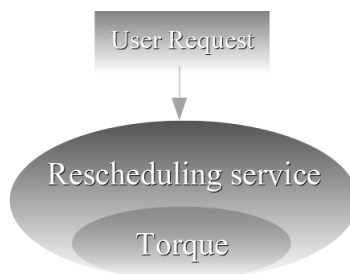


Fig. 2. Global Re-scheduling Architecture

The main motivation for the selection of Torque over Condor is the flexibility we've still got. This thesis adds a new layer over the resource management represented by a scheduling and re-scheduling service, like in Figure 2. Condor offers a lot of already implemented services and it is harder to add a new layer that has to overwrite some facilities.

5 Experimental Results

In this section we aim to evaluate the four DAG scheduling algorithms presented: CCF, ETF, HLFET and HybridRemapper. In particular we are interested in analyzing their performance on a realistic scenario. For this reason this experiment considered the existence of two available resources, directly connected to each other and a set of tasks with dependencies.

5.1 *Example of an Execution Flow*

Along the execution, a task can be in various states. When the user submits the request of executing the specific workflow of tasks, all the tasks will be assigned with the "initial" state. This service schedules the specified tasks and sends them to the processors on which they will be executed. At this moment the state of these tasks is changed to "submitted". When a particular task is effectively executing on one of the available processors, it is in the "running". The last state is "finished", and is reached when a task completes its job. At any time, an failure can occur and that will lead to the change of that specific task in "error" state.

The behavior of this re-scheduling algorithm in the case of failure detection and several stages of the graph development from the initial state in which is submitted for execution in Grid environment until all tasks are completed is shown in Figure 3. This example explains how the algorithm works on a graph with 12 tasks with dependencies between them and as available resources two directed connected processors.

As it can be seen, the first graph represents the initial directed acyclic graph of task. According to the schedule each task will be submitted at a certain moment of time. In the second graph, tasks C and D are already running and tasks B and E were just submitted. The third graph shows that an error occurred. At this moment, all tasks that are in "running" or "submitted" state will be canceled and rescheduled. The proposed service needs to construct a new graph with all the unfinished tasks and to send it to the schedule. Now the unfinished tasks will be executed according with the new schedule.

5.2 *Test Scenarios*

Functional test scenarios are divided into multiple categories:

- Test that contains a bag of independent jobs. These are useful in order to test the basic functionality and interoperability of all the project components: resource analyzer, scheduler, monitor and reschedule.

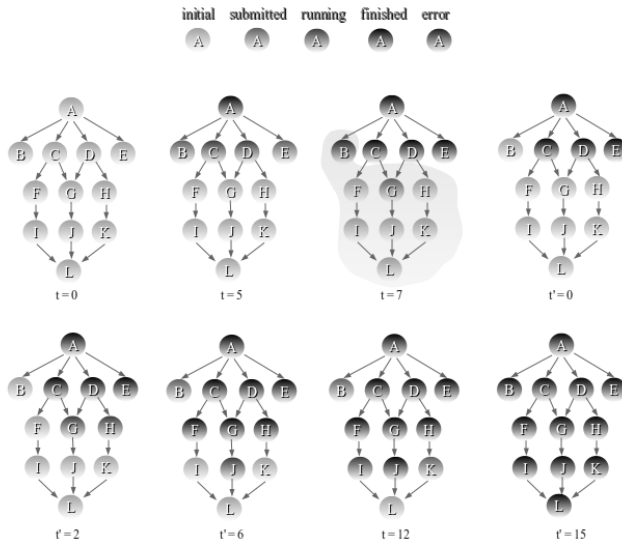


Fig. 3. Example of execution steps of the re-scheduling service using CCF

- Test that contains workflows with jobs that have interdependencies, but the output files of one task is not required for another job. The purpose is to test the scheduling algorithm, the correct launch of tasks on the nodes.
- Test that contains a workflow with jobs that have interdependencies also in the form of input-output files. In other words, the output file of a task will be the input file of another task. This test is different from the previous one because it refers to the sandbox synchronization. The output and input files must be in the same place for every node, and the sandbox should not suffer of synchronization issues. For example an output file is ready on node x where a task has finished, but is not present on node y where another task needs it because of the `sshfs` delay.
- Test that contains workflows with jobs that have interdependencies also in the form of input-output files, and a task will fail once. This is a basic test for the monitor and re-scheduler components. We must be sure that the monitor identifies correctly and in time the faulty job and calls the re-scheduler. Also, it is important what sub-graph is send to the re-scheduler.
- Test that contains the setting for the previous test, only that the error occurs with the frequency of 1-5%. This test aims to be as close as possible to a real life test.
- A comparison test that repeats the third test but using different scheduling algorithms. The purpose is to compare the performance of each algorithm on different workflows and to decide which the best on particular inputs is.

- Another set of tests try to determine several limitations of the project. The test is meant to determine the minimal time duration of a task, the maximal file that could be safely transferred without causing overhead and synchronization problems between sandboxes.

5.3 Performance Analysis of Scheduling Algorithms

For the experiments it was chosen a DAG of 12 tasks which contains a classic scenario of master-slaves type, with sequences of linear dependencies between nodes. This graph is presented in Figure 4.

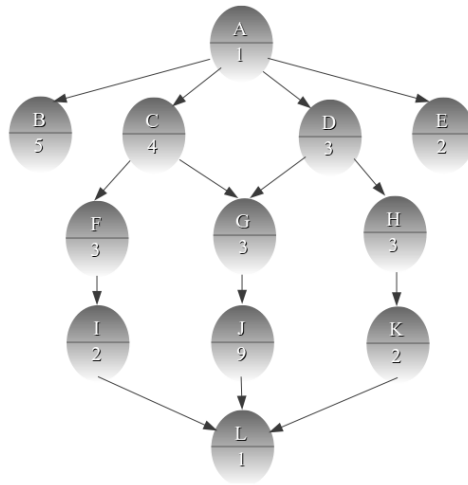


Fig. 4. Test Graph

A node represents a task identified by an id and the execution time and the edges indicate the dependencies between them. As we already specify, the configuration consists in two connected processors. Each edge has a cost with represents the communication time between processors if the tasks are executed on different resources. With the assumed conditions, the four experiments generated the following scheduling decision of task assignments that are presented in Figure 5.

As it can be seen, the CCF algorithm offers the best load balancing and the minimum running time. The HLFET algorithm produces a schedule that has the same execution time as CCF, while ETF and HybridRemapper algorithms generates a schedule that needs more time for completion.

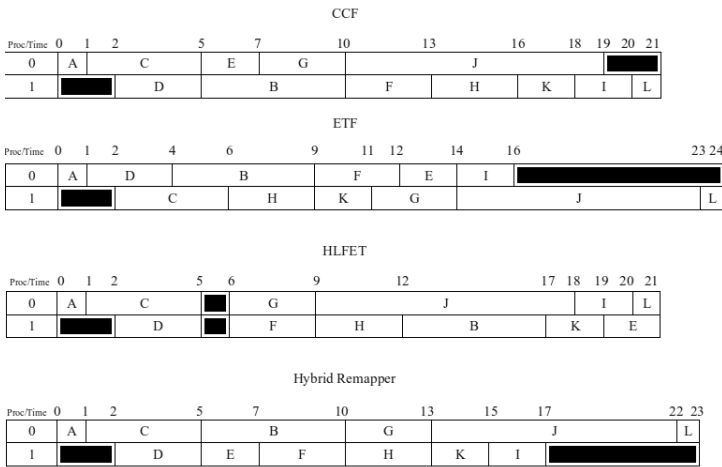


Fig. 5. Schedule decisions for all implemented algorithms

The next task allocation strategy through which a resource is allocated to a task give certain advantages to the scheduling policies. The HLFET scheduling algorithms, as shown in the experimental results, have a good time complexity and finish the scheduling operation in a very short time. The ETF algorithm brings a more complex approach and therefore is more time-consuming in allocating all resources to the correspondent nodes. In spite of that, sometimes, ETF has proven more efficient than HLFET.

Taking this into account, the CCF scheduling algorithm has the best performance regarding both the schedule length over all processors and the time spent for scheduling.

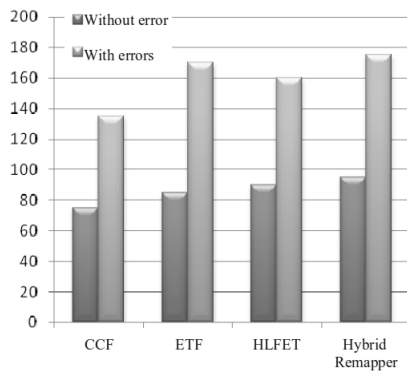


Fig. 6. Execution Time for all implemented algorithms

We must also take into consideration that the above results have been obtained for a small number of tasks and a reduced number of processing resources. Further analysis on a larger and more complex DAG could lead to different results regarding total processing time and load balancing.

As a conclusion we present a chart comparing these algorithms using non-error test and error test. In the graph shown in Figure 6 we notice that the CCF algorithm give the best times for re-scheduling. This is true for the total running time and also for the difference between the error case and the case without errors. The weakest algorithm from the error versus no error point of view is ETF. This means that in the case of reorganizing task execution order, the ETF algorithm will conclude in the worst results.

The hierarchy of the algorithms from the time of execution criteria is also presented in Figure 6. CCF offers the best results whether the system is error-prone or not. On the other hand, the second place it is occupied by ETF algorithm in case no error occurs and by HLFET in case of errors.

5.4 *Limitations*

Our applications present several limitations. First of all, if multiple workflows are submitted, they are scheduled one after the other. That means that resources are allocated for the first workflow. After completion of the first workflow, the second workflow is submitted. Thus, this does not necessary obtains the optimal scheduling, taking into consideration both workflows. Another thing is that we must have only one manager that does resource allocation. This could also provide a single point of failure. Furthermore, task must be accurately profiled in order to have an efficient scheduling. If a job takes longer to complete normally than the scheduled time, the workflow will never be successful (and will be inefficient because it will try resubmitting the job, not knowing the job does not actually ended with error).

Also, this implementation is designed to suit computational intensive tasks, not communication intensive tasks. Limitations like ssh transfer and sshfs update delay leads us to this conclusion. As test shows, it would be safe that files passed by tasks as input-output should not exceed a size of 4MB.

Another thing worth mentioning is the fact the tasks should have a time length of a minimal magnitude in tens of seconds. This is important for the way the monitor works: it issues a `qstat` command and parses the output, determining what tasks have finished. This is done with a frequency of one test per second. Thus, a very short task with duration of just a few seconds could be missed (improbable but possible).

5.5 *Conclusions and Future Work*

One of the major disadvantages of a Grid system is the lack of resource control. Resources enter and leave the system all the time. Failures in the system can be caused either by a single component's error or through the interaction between components. One of main fault tolerance methods is re-scheduling. Re-scheduling

isn't always the best solution, because it may cause important delays which can affect the performance of the system. Still, the re-scheduling technique remains a valuable tool. The re-scheduling algorithm proposed in this thesis has an important characteristic: it is a generic algorithm because it can be used with a large variety of scheduling heuristics. The proposed evaluation model is based on a series of metrics. This model led to a classification of the scheduling algorithms used together with the proposed re-scheduling procedure, after their performance in case of errors and re-scheduling. Assessments concluded with the observation that algorithms with the best performance if no errors occur tend to achieve the best scores also when re-scheduling is needed. The algorithm with the best results after our testing and analysis is CCF (Cluster ready Children First).

The main contribution of this thesis is the design and implementation of a re-scheduling service in Grid environment on top of the resource manager PBS, in his open source form, Torque. The purpose of this service is to improve the service quality of the scheduling in Grid environment, although the rate of failures is low, about 1% of the computed tasks, reducing it is important because it might increase the client satisfaction and confidence.

Other important contributions that have an important impact in the Grid scheduling domain perform a critical analysis of several scheduling algorithms for independent and dependent task scheduling. This analysis was the base for selecting and implementing the proposed re-scheduling algorithm heuristics. Other contribution is the validation of the re-scheduling service proposed by integration and testing it in a real environment.

The re-scheduling service proposed can be used with several re-scheduling strategies, whose classification will be developed according with the major types of graphs that needs to be rescheduled. By defining and implementing these classification methods it can analyze what combination of scheduling algorithms and re-scheduling strategies are the most appropriate to use, depending on the type of graph.

Acknowledgments. The research presented in this paper is supported by national project: "SORMSYS - Resource Management Optimization in Self-Organizing Large Scale Distributed Systems", Project CNCIS-PN-II-RU-PD ID: 201 (Contract No. 5/28.07.2010). The work has been co-funded by the Sectorial Operational Program Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement, POSDRU/89/1.5/S/62557

References

1. Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Marchal, L., Robert, Y.: Centralized versus distributed schedulers for bag-of-tasks applications. *IEEE Trans. Parallel Distrib. Syst.* 19(5), 698–709, <http://dx.doi.org/10.1109/TPDS.2007.70747>
2. Xavier, P., Lee, B.S., Cai, W.: A decentralized hierarchical scheduler for a grid-based clearinghouse. In: *IPDPS 2003: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*. IEEE Computer Society, Washington, DC (2003)

3. Pop, F.: Communication model for decentralized meta-scheduling in grid environments. In: Proceedings of The Second International Conference on Complex, Intelligent and Software Intensive System, Second International Workshop on P2P, Parallel, Grid and Internet Computing - 3PGIC 2008 (CISIS 2008), March 4-7, pp. 315–320. IEEE Computer Society, Barcelona (2008) ISBN: 0-7695-3109-1
4. Singho, F., Plantec, A.: AADL modeling and analysis of hierarchical schedulers. In: SIGAda 2007: Proceedings of the 2007 ACM International Conference on SIGAda Annual International Conference. ACM, New York (2007), <http://doi.acm.org/10.1145/1315580.1315593>
5. Goyal, P., Guo, X., Vin, H.M.: A Hierarchical CPU Scheduler for Multimedia Operating Systems. Technical Report. UMI Order Number: CS-TR-96-12, University of Texas at Austin (1996)
6. Regehr, J., Stankovic, J., Humphrey, M.: The Case for Hierarchical Schedulers with Performance Guarantees. Technical Report. UMI Order Number: CS-2000-07, University of Virginia (2000)
7. Pop, F., Dobre, C., Cristea, V.: Performance analysis of grid dag scheduling algorithms using monarc simulation tool. In: ISPD 2008: Proceedings of the 2008 International Symposium on Parallel and Distributed Computing, pp. 131–138. IEEE Computer Society, Washington, DC (2008), [doi: http://dx.doi.org/10.1109/ISPD.2008.15](http://dx.doi.org/10.1109/ISPD.2008.15)
8. Ma, Z., Catthoor, F., Vounckx, J.: Hierarchical task scheduler for interleaving subtasks on heterogeneous multiprocessor platforms. In: Proceedings of the 2005 Conference on Asia South Pacific Design Automation ASP-DAC 2005, January 18-21, pp. 952–955. ACM, New York (2005), <http://doi.acm.org/10.1145/1120725.1120765>
9. Regehr, J., Stankovic, J.A.: Hierarchical schedulers, performance guarantee, and resource management. SIGOPS Oper. Syst. Rev. 34(2), 31 (2000), <http://doi.acm.org/10.1145/346152.346216>
10. Forti, A.: Dag Scheduling for Grid Computing Systems (PhD Thesis). Ph.D. thesis. UNIVERSITY OF UDINE - ITALY, Department of Mathematics and Computer Science (2006)

Java Reflection Performance Analysis Using Different Java Development

Cătălin Tudose¹, Carmen Odubășteanu², and Serban Radu²

¹ ITC Networks, Calea Floreasca 167, Bucharest, Romania,
catalin_tudose@yahoo.com

² Department of Computer Science, "POLITEHNICA" University of Bucharest,
Spl. Independentei 313, 77206 Bucharest, Romania
{carmen.odubasteanu, radu.serban}@cs.pub.ro

Abstract. The reflection technique is used by programs that need to examine or modify the runtime behavior of applications in the Java Virtual Machine. This is a powerful instrument, but its use should bring some concerns related to performance overhead, security problems, exposing the private fields - which are not accessible through non-reflective code, losing benefits of compile-time type checking, reflective code being clumsy and verbose. This paper concentrates upon the performance overhead. Tests were made in order to analyze objects creation and method calls using primitive and reference arguments, both by direct and reflective invocation.

Keywords: Java, reflection performance, JDK, object creation, method call, direct invocation, reflective invocation.

1 Introduction

Reflection is the process by which software can get information about itself and modify its own structure and behavior. The programming paradigm driven by reflection is called *reflective programming*. It is a particular kind of meta-programming.

Brian Cantwell Smith's 1982 doctoral dissertation introduced the notion of computational reflection in programming languages ([1], [2]).

Reflection can be used for getting information and/or modifying program execution at runtime. A reflection-oriented program component can monitor the execution of pieces of code. It can also modify itself according to a desired goal. This is accomplished at runtime, by dynamically changing program code.

The programming paradigm that relies on reflective language features have been object of study for more than 15 years [3]. Reflection can also be used to dynamically adapt a given program to different situations. For example, consider an application that uses two different classes X and Y interchangeably to perform

similar operations. Without reflection-oriented programming, the application might be hard-coded to call method names of class X and class Y. However, using the reflection-oriented programming paradigm, the application could be developed to use reflection in order to invoke methods in classes X and Y without hard-coding method names. Reflection-oriented programming almost always requires additional knowledge, framework, relational mapping, and object relevance in order to take advantage of more generic code execution. Hard-coding can be avoided to the extent that reflection-oriented programming is used.

Reflection is a powerful technique, but should not be used extensively. Operations that may be performed directly should be preferred to the ones using reflection. There are some concerns that a programmer should know about when writing reflective code ([4], [5]):

- **Performance Overhead:** Reflective operations require types that must be dynamically resolved and the Java Virtual Machine should load classes and information at runtime. The non-reflective code loads classes and information at compile time. Consequently, reflective operations have slower performance than the non-reflective ones. If possible, a programmer should avoid calling frequently this kind of operations, as they are time-consuming.
- **Security Restrictions:** Reflection requires runtime permissions which may not be present when running under a security manager. This is an important consideration for code which has to run in a restricted security context, as an applet.
- **Exposure of Internals:** Reflection allows code to access private fields and methods, a behavior that is not present in the non-reflective code. This may result in dangerous effects. Reflective code breaks encapsulation and may change behavior with upgrades of the platform.
- **Benefits of Compile-Time Type Checking Are Lost:** Exception checking is also lost. If a program attempts to invoke a nonexistent or inaccessible method reflectively, it will fail at runtime unless the programmer has taken special precautions.
- **Reflective Code Is Clumsy and Verbose:** It is tedious to write and difficult to read

2 Reflection Model View

The starting point for using reflection is always a `java.lang.Class` instance. If the program works with a predetermined class, loaded at design-time, the Java language provides an easy shortcut to get the `Class` instance directly: `Class cls = Triangle.class;`

If an instance of an object is available, then the simplest way to get its `Class` is to invoke `Object.getClass()`. Surely, this only works for reference types which all inherit from `Object` and does not work for primitive types. As an example for the `String` class: `Class c = "string".getClass();`

If the program needs to read the class name at runtime from some external source, the approach is different. The program needs to use a class loader to find the class information.

```
// "name" is the class name to load
Class clas = null;
try {
    //call the class loader
    clas = Class.forName(name);
}catch(ClassNotFoundException classNotfoundEx) {
    // handle exception case
    classNotfoundEx.printStackTrace();
}
```

2.1 Constructors by Reflection

The Class object gives all the basic means for reflection access to the class metadata. This metadata includes: information about the class itself, such as the package and superclass; modifiers (as `public`, `protected`, `private`, `final`, `static`, `abstract` and `interface`); interfaces implemented by the class; details of the constructors, fields and methods defined by the class.

These 3 types of class components - constructors, fields and methods – will be the one to be used for the experiments detailed into this paper. The `java.lang.Class` provides 4 separate reflection calls to access information in different ways. All the calls follow a standard form. For the constructors, they are [6]:

- Constructor `getConstructor(Class[] params)` - Returns a Constructor object that reflects the specified public constructor of the class represented by this Class object
- Constructor[] `getConstructors()` - Returns an array containing Constructor objects reflecting all the public constructors of the class represented by this Class object
- Constructor `getDeclaredConstructor(Class[] params)` - Returns a Constructor object that reflects the specified constructor of the class or interface represented by this Class object
- Constructor[] `getDeclaredConstructors()` - Returns an array of Constructor objects reflecting all the constructors declared by the class represented by this Class object. The `java.lang.Class` class defines a special method to be used to create an instance of a class with a *no-argument* (or default) constructor.
- Object `newInstance()` -- Creates a new instance of the class represented by this `java.lang.Class` object

The class is instantiated as if by a `new` expression with an empty argument list. The class is initialized if it has not already been initialized.

In order to call a constructor with arguments, the sequence goes as following:

- Use the `getConstructor` method, get the public constructor using the specified parameter types. As example:

```
Constructor<?> constructor = cls.getConstructor(double.class, double.class, double.class);
```
- Call the `newInstance` method from the constructor with a specified set of parameters. This uses the constructor represented by the `Constructor` object to create and initialize a new instance of the constructor's declaring class, with the specified initialization parameters. Individual parameters are automatically unwrapped to match primitive formal parameters and both primitive and reference parameters are subject to method invocation conversions as necessary. As example:

```
Triangle reflectionTriangle = (Triangle)constructor.newInstance(a, b, c);
```

2.2 *Fields by Reflection*

The `Class` reflection calls to access field information are similar to those used to access constructors, with a field name used in place of an array of parameter types [4]:

- `Field getField(String name)` - Returns a `Field` object that reflects the specified public member field of the class or interface represented by this `Class` object
- `Field[] getFields()` - Returns an array containing `Field` objects reflecting all the accessible public fields of the class or interface represented by this `Class` object
- `Field getDeclaredField(String name)` - Returns a `Field` object that reflects the specified declared field of the class or interface represented by this `Class` object
- `Field[] getDeclaredFields()` - Returns an array of `Field` objects reflecting all the fields declared by the class or interface represented by this `Class` object

The first two methods return information for public fields that can be accessed through the class - even those inherited from an ancestor class. The last two methods return information for fields declared directly by the class - regardless of the fields' access types. As constructors are not inherited, this makes a difference to the similar methods that are getting the constructors.

2.3 *Methods by Reflection*

The `Class` reflection calls to access method information are similar to those used for constructors and fields, with a method name used in place of an array of parameter types [4]:

- Method `getMethod(String name, Class[] params)` - Returns a `Method` object that reflects the specified public member method of the class or interface represented by this `Class` object
- Method[] `getMethods()` - Returns an array containing `Method` objects reflecting all the public *member* methods of the class or interface represented by this `Class` object, including those declared by the class or interface and those inherited from superclasses and superinterfaces
- Method `getDeclaredMethod(String name, Class[] params)` - Returns a `Method` object that reflects the specified declared method of the class or interface represented by this `Class` object
- Method[] `getDeclaredMethods()` - Returns an array of `Method` objects reflecting all the methods declared by the class or interface represented by this `Class` object

The first two methods return information for public methods that can be accessed through the class - even those inherited from an ancestor class. The last two methods return information for methods declared directly by the class - regardless of the methods' access types.

In order to call a method, the sequence goes as following:

- Get a `Method` object, with a given name and a given list of argument types

```
Method method = cls.getMethod("setSideNames", String.class, String.class, String.class);
```
- Invoke the method with a particular object and a given list of parameters

```
method.invoke(reflectionTriangle, aSide, bSide, cSide);
```

2.4 Reflection Using Cautions

As detailed in the introduction part, there are cautions that a programmer should keep in mind when using the reflection. These ones include performance overhead, security restrictions, exposure of internals, losing benefits of compile-time type checking, reflective code being clumsy and verbose. This paper extensively studies the performance overhead, by testing and evaluating the reflective code execution using different JDKs.

Previous evaluations – general ones or detailed ones - have been made. Joshua Bloch [4] writes: “Reflective method invocation is much slower than normal method invocation. Exactly how much slower is hard to say, because there are so many factors at work. On my machine, the speed difference can be as small as a factor of two or as large as a factor of fifty.” Also, Dennis Sosnoski has developed a series of articles and tests in 2003, having as purpose to explain the Java class loading, to evaluate reflection performances and applications [7]. Since 2003, new JDKs have been released and performances have changed. Consequently, our testing evaluate the present-day performances of the reflection technique, using Java Virtual Machine (VM) release 5 and 6.

3 Building The Reflection Testing Environment

3.1 *Classes Used for Reflection Testing*

In order to make the testing, two classes have been developed: one that will create objects whose constructors and methods will be tested both by direct and reflective access (the `Triangle` class); one that will create and manage these objects both directly and through reflection (the `TrianglesReflection` class).

The piece of code below illustrates one of the methods that is created and invoked in order to test the reflection performances. The following steps are followed:

- the class is obtained using the `Class.forName` method. It returns the a reference `cls` to the class `Triangle`;
- the constructor is obtained using the `cls.getConstructor` method, with the list of arguments. It returns the constructor `constructor`;
- the constructor is reflectively invoked through `constructor.newInstance`. It returns a `Triangle` object.

```
private static void ObjCreationReflection(double a,
    double b, double c) {
    try{
        beforeReflectionInvocation = System.nanoTime();
        Class<?> cls =
            Class.forName("reflection.Triangle");
        long afterClassLookupTime = System.nanoTime();
        classLookupTime += afterClassLookupTime-
            beforeReflectionInvocation;
        Constructor<?> constructor =
            cls.getConstructor(double.class,
                double.class, double.class);
        long afterConstructorLookupTime =
            System.nanoTime();
        constructorLookupTime += afterConstructorLookup
            Time-afterClassLookupTime;
        Triangle reflectionTriangle =
            (Triangle)constructor.newInstance(a, b, c);
```

```
        afterReflectionInvocation = System.nanoTime();
        constructorInvocationTime +=
            afterReflectionInvocation -
            afterConstructorLookupTime;
        reflectionTime += afterReflectionInvocation -
            beforeReflectionInvocation;
    }
    catch (Exception exc) {
        exc.printStackTrace();
    }
}
```

3.2 *Tests Accomplished for Reflection Performance Evaluation*

The objectives followed by the tests are:

- Evaluate the object creation with primitive arguments through direct invocation and through reflection.
- Evaluate the object creation with reference arguments through direct invocation and through reflection.
- Evaluate the methods invocation with primitive arguments through direct call and through reflection.
- Evaluate the methods invocation with reference arguments through direct call and through reflection.
- Evaluate the time spent by the reflection mechanism into its phases: class lookup time, constructor/method lookup time, constructor/method invocation time.

Execute all tests above with different JDKs, in order to evaluate the differences from one version to another.

3.3 *Effective Testing*

A number of 10 tests have been executed, each of them creating 100.000 objects. The execution has been made both by direct and by reflective invocation.

For object creation with primitive the constructors use double primitive arguments. The results are illustrated in Table 1.

For object creation with reference arguments the constructors use Point.Double reference arguments. The results are illustrated in Table 2.

For method invocation with primitive arguments The method uses double primitive arguments. The results are illustrated in Table 3.

For method invocation with reference arguments the method uses reference String arguments. The results are illustrated in Table 4.

Table 1. Object creation with primitive arguments

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
jdk1.6.0_11											
Direct time (ms)	115.03	112.22	106.00	107.37	106.22	112.33	109.09	105.84	107.77	107.21	108.91
Reflection factor	7.56	8.00	8.38	8.18	8.79	8.22	8.28	7.38	7.51	8.16	8.04
jdk1.6.0_18											
Direct time (ms)	108.54	110.11	135.72	109.38	107.37	107.99	108.68	107.49	110.07	105.74	111.11
Reflection factor	8.01	7.84	7.91	7.96	8.18	7.81	7.91	8.03	7.58	8.11	7.93
jdk1.5.0_06 - reflection											
Class lookup	41.83%	41.58%	41.66%	41.49%	39.39%	41.98%	42.32%	41.74%	41.86%	42.59%	41.64%
Constructor lookup	37.68%	38.31%	37.95%	38.52%	42.39%	37.91%	38.24%	37.99%	38.05%	37.76%	38.48%
Constructor invocation	20.49%	20.11%	20.40%	19.99%	18.22%	20.11%	19.44%	20.27%	20.09%	19.65%	19.88%
jdk1.6.0_11 - reflection											
Class lookup	45.01%	45.66%	46.20%	45.36%	42.65%	45.11%	45.15%	45.24%	42.34%	44.65%	44.74%
Constructor lookup	32.20%	32.19%	31.67%	32.24%	34.68%	32.34%	32.73%	32.69%	32.70%	32.68%	32.61%
Constructor invocation	22.79%	22.15%	22.13%	22.40%	22.67%	22.55%	22.12%	22.07%	24.95%	22.67%	22.65%
jdk1.6.0_18 - reflection											
Class lookup	46.80%	47.60%	47.32%	47.08%	48.15%	47.35%	46.84%	47.24%	46.10%	46.49%	47.10%
Constructor lookup	33.73%	34.29%	34.11%	33.88%	33.59%	33.70%	34.02%	33.62%	35.44%	34.36%	34.07%
Constructor invocation	19.47%	18.11%	18.58%	19.04%	18.26%	18.95%	19.14%	19.14%	18.46%	19.15%	18.83%

Table 2. Object creation with reference arguments

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
jdk1.5.0_06											
Direct time (ms)	134.25	136.18	133.26	137.33	135.09	138.10	134.20	140.95	141.35	142.06	137.28
Reflection factor	7.13	7.53	7.20	7.16	7.20	7.31	7.17	7.43	6.55	7.06	7.17
jdk1.6.0_11											
Direct time (ms)	139.52	145.19	137.95	139.46	139.20	142.57	138.32	136.96	135.52	141.98	139.67
Reflection factor	6.04	5.84	6.03	5.34	6.09	4.91	6.08	5.82	5.80	5.84	5.78
jdk1.6.0_18											
Direct time (ms)	139.04	147.99	139.31	137.82	143.65	143.04	141.68	138.20	147.45	141.02	141.92
Reflection factor	6.01	6.26	6.54	6.64	6.34	6.18	6.28	6.31	6.11	6.38	6.30
jdk1.5.0_06 - reflection											
Class lookup	45.68%	45.55%	45.68%	45.85%	45.56%	45.46%	45.95%	44.73%	44.38%	45.22%	45.41%
Constructor lookup	35.41%	35.39%	35.25%	35.21%	35.66%	35.33%	35.36%	36.46%	37.22%	36.06%	35.74%
Constructor invocation	18.91%	19.06%	19.08%	18.94%	18.79%	19.21%	18.69%	18.81%	18.40%	18.73%	18.86%
jdk1.6.0_11 - reflection											
Class lookup	45.11%	43.75%	42.79%	44.50%	43.28%	37.49%	43.04%	44.15%	43.87%	43.84%	43.18%
Constructor lookup	34.58%	35.86%	35.04%	35.12%	36.11%	47.65%	35.33%	35.06%	35.17%	35.72%	36.56%
Constructor invocation	20.31%	20.39%	22.17%	20.38%	20.61%	14.86%	21.63%	20.79%	20.96%	20.44%	20.25%
jdk1.6.0_18 - reflection											
Class lookup	44.02%	43.53%	44.07%	43.78%	43.85%	43.21%	43.05%	43.12%	43.57%	43.38%	43.56%

Table 2. (continued)

Constructor lookup	35.96%	36.66%	34.70%	36.30%	36.34%	36.58%	36.53%	36.90%	36.39%	35.63%	36.20%
Constructor invocation	20.02%	19.82%	21.23%	19.93%	19.81%	20.21%	20.42%	19.98%	20.04%	20.99%	20.25%

Table 3. Method invocation with primitive arguments

jdk1.6.0_11	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
Direct time (ms)	111.29	109.45	109.77	122.88	115.49	109.70	104.70	110.32	107.29	108.02	110.89
Reflection factor	4.40	2.95	3.43	2.98	3.13	3.23	2.97	3.01	3.13	3.25	3.25
jdk1.6.0_18											
Direct time (ms)	118.39	103.78	104.48	100.48	103.42	102.23	102.09	106.03	100.70	104.42	104.60
Reflection factor	3.47	3.32	3.47	3.61	3.39	3.07	3.54	3.44	3.39	3.41	3.41
jdk1.5.0_06 – reflection											
Class lookup	27.10%	27.74%	27.48%	28.69%	27.75%	27.92%	27.94%	28.02%	27.68%	28.89%	27.92%
Constructor lookup	22.56%	23.76%	22.91%	22.52%	23.24%	22.96%	22.81%	23.36%	23.42%	22.60%	23.01%
Constructor invocation	15.68%	13.58%	13.26%	13.01%	13.73%	12.72%	13.87%	12.44%	12.45%	12.33%	13.31%
Method lookup	25.94%	25.82%	27.08%	26.79%	26.34%	27.46%	26.65%	27.17%	27.44%	25.68%	26.64%
Method invocation	8.72%	9.11%	9.26%	8.98%	8.94%	8.94%	8.73%	9.02%	9.02%	10.50%	9.12%
jdk1.6.0_11 – reflection											
Class lookup	31.84%	29.36%	31.81%	32.49%	31.19%	33.81%	28.03%	35.66%	27.42%	31.12%	31.27%
Constructor lookup	21.29%	20.58%	20.19%	25.95%	15.35%	19.90%	17.24%	21.67%	26.43%	21.39%	21.00%

Table 3. (continued)

Constructor invocation	14.54%	15.15%	12.53%	10.73%	16.47%	14.30%	14.92%	13.09%	12.42%	13.61%	13.78%
Method lookup	23.16%	24.52%	25.62%	22.98%	22.42%	23.14%	26.49%	19.89%	22.54%	23.80%	23.46%
Method invocation	9.17%	10.39%	9.86%	7.85%	14.58%	8.85%	13.33%	9.68%	11.18%	10.09%	10.50%
jdk1.6.0_18 – reflection											
Class lookup	30.62%	30.26%	30.59%	31.44%	31.44%	31.72%	31.17%	31.07%	31.18%	31.51%	31.10%
Constructor lookup	21.81%	22.49%	21.74%	22.23%	21.91%	21.61%	21.60%	21.54%	21.42%	21.63%	21.80%
Constructor invocation	13.85%	12.69%	13.02%	11.79%	11.94%	12.01%	11.90%	12.36%	11.96%	12.03%	12.36%
Method lookup	23.58%	24.55%	24.03%	24.00%	24.22%	24.28%	24.43%	25.10%	24.83%	24.71%	24.37%
Method invocation	10.14%	10.01%	10.62%	10.54%	10.49%	10.39%	10.90%	9.94%	10.61%	10.13%	10.38%

Table 4. Method invocation with reference arguments

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Average
jdk1.5.0_06											
Direct time (ms)	107.03	109.28	102.67	107.73	106.94	108.98	107.05	113.29	103.40	111.94	107.83
Reflection factor	5.21	4.67	4.94	4.98	4.95	5.18	4.89	4.67	5.05	4.92	4.95
jdk1.6.0_11											
Direct time (ms)	111.58	119.67	117.10	110.34	107.04	110.36	112.72	124.47	107.84	105.91	112.70
Reflection factor	4.07	3.53	3.56	3.61	3.75	3.59	3.61	3.59	3.59	3.73	3.66
jdk1.6.0_18											
Direct time (ms)	107.47	116.75	120.08	103.27	106.69	103.13	103.82	106.58	113.17	103.49	108.44
Reflection factor	4.03	3.77	4.01	3.74	4.09	4.02	4.30	3.99	3.84	3.93	3.97

Table 4. (continued)

jdk1.5.0_06 – reflection										
Class lookup	27.61%	27.24%	27.25%	27.64%	27.40%	22.88%	27.73%	28.10%	25.94%	26.94%
Constructor lookup	22.59%	23.02%	22.75%	22.66%	21.64%	33.65%	22.66%	22.54%	22.13%	23.70%
Constructor invocation	13.12%	13.10%	12.94%	13.15%	15.08%	11.35%	13.04%	13.37%	12.86%	13.13%
Method lookup	26.90%	27.11%	27.84%	27.02%	26.25%	23.97%	27.03%	26.60%	30.15%	26.90%
Method invocation	9.78%	9.53%	9.21%	9.53%	9.73%	8.16%	9.55%	9.40%	8.93%	9.34%
jdk1.6.0_11 – reflection										
Class lookup	30.99%	31.66%	27.02%	31.19%	30.92%	31.25%	31.07%	30.52%	31.17%	30.79%
Constructor lookup	20.65%	20.94%	26.25%	21.27%	21.20%	20.95%	21.02%	20.24%	20.95%	21.54%
Constructor invocation	13.52%	13.41%	13.17%	13.40%	13.91%	13.49%	13.43%	13.09%	13.74%	13.29%
Method lookup	24.08%	23.53%	23.37%	23.77%	23.72%	24.12%	23.95%	25.18%	23.58%	24.05%
Method invocation	10.76%	10.46%	10.19%	10.37%	10.26%	10.18%	10.52%	10.97%	10.55%	10.34%
jdk1.6.0_18 – reflection										
Class lookup	30.98%	31.68%	30.99%	31.87%	32.48%	31.62%	31.55%	31.43%	31.44%	31.51%
Constructor lookup	20.66%	20.64%	20.44%	20.33%	20.16%	20.80%	20.52%	20.54%	21.50%	20.62%
Constructor invocation	12.00%	11.45%	11.10%	11.65%	11.69%	13.17%	11.68%	11.63%	11.53%	11.73%
Method lookup	25.42%	25.88%	27.88%	26.00%	25.73%	24.40%	26.33%	26.21%	25.55%	26.08%
Method invocation	10.95%	10.35%	9.58%	10.15%	9.63%	10.00%	9.93%	10.18%	9.97%	10.07%

4 Conclusions

The Java reflection technique may be used for dynamically load and use classes at runtime. Using it in various cases may be of great help for the programmer. In tandem with the very much-used generics it may offer at runtime information that is unknown at design time. It is a powerful tool, but comes with a pretty high price, as its usage is time-consuming and may seriously affect program performances.

Performance improvements have been made between JDKs, but they are not spectacular and the performance caution of reflection usage should be kept in mind.

The testing environment was a machine with an Intel Core 2 Duo processor, 2 GHz, 4GB of RAM and Windows XP Service Pack 3 operating system.

For object creation using primitive arguments, between `jdk1.5.0_06` and `jdk1.6.0_18` the reflection factor (as compared to direct object creation) decreases from about 8.9172 to 7.9326. For 100,000 objects, the difference is from 1.024 seconds to 881 mili-seconds. It is a 16% improvement.

For object creation using reference arguments, the performances are better for `jdk1.6.0_11` than for `jdk1.6.0_18`. It seems that the reflection performance has not been a preoccupation for the developers. Creating 100,000 objects with `jdk1.6.0_11` took 807 mili-seconds, while creating them with `jdk1.5.0_06` took 984 mili-seconds. This is a 22% improvement. The reflection factor (as compared to direct object creation) decreases from 7.1742 to 5.7791.

Method invocation with primitive arguments takes 512 mili-seconds for 100,000 calls in `jdk1.5.0_06`, while it takes 357 mili-seconds for `jdk1.6.0_18`. It is a 43% improvement. The reflection factor (as compared to direct method invocation) decreases from 4.0965 to 3.2482.

Method invocation using reference arguments takes 533 mili-seconds for 100,000 calls in `jdk1.5.0_06`, while it takes 413 mili-seconds for `jdk1.6.0_11`. This is a 29% improvement. Again, performances are better for `jdk1.6.0_11` than for `jdk1.6.0_18`. The reflection factor (as compared to direct method invocation) decreases from 4.945 to 3.6638.

After the 4 conclusions above, we may say that the performance improvements really exist, but it is not spectacular – the best one being of 43%. Besides this, the most optimistic report factor between reflective invocation and direct invocation is at least 3.2482.

For `jdk1.6.0_18`, the usage of constructors and methods with primitive arguments improves compared to `jdk1.6.0_11`, as well as it grows worse for constructors and methods with reference arguments.

Most of the time spent in reflective operations consists of the lookup part. Class lookup takes more than 40% of the time, while constructor lookup takes more than 30%. Reflective invocation of the constructor takes less than 25%. The programmer must avoid frequent lookup operations. The class, the constructor or the method lookup should not be included into loop operations. The lookup should be made only once, and the reference obtained should be used in the entire program.

Instructions like:

```
Class<?> cls = Class.forName("reflection.Triangle");
Constructor<?> constructor =
    cls.getConstructor(double.class, double.class,
        double.class);
Method method = cls.getMethod("setSideNames",
    String.class, String.class, String.class);
```

should be taken outside loops. Thus, the performances will get much closer to direct invocations.

References

1. Cantwell Smith, B.: Procedural Reflection in Programming Languages, Department of Electrical Engineering and Computer Science. Massachusetts Institute of Technology, PhD Thesis (1982)
2. Cantwell Smith, B.: Reflection and semantics in a procedural language. Technical Report MIT-LCS-TR-272. Massachusetts Institute of Technology, Cambridge, Mass. (January 1982)
3. Java Sun Tutorial
4. Java™ Platform, Standard Edition 7 API Documentation
5. Sobel, J., Friedman, D.: An Introduction to Reflection-Oriented Programming. Indiana University (1996)
6. Bloch, J.: Effective Java. Addison Wesley (2008)
7. Sosnoski, D.: Java programming dynamics (2003)

Multi GPGPU Optimizations for 3D MMO Virtual Spaces

Victor Asavei, Florica Moldoveanu, Alin Moldoveanu, Alexandru Egner,
and Anca Morar

Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest,
Romania

{victor.asavei, florica.moldoveanu, alin.moldoveanu,
alexandru.egner, anca.morar}@cs.pub.ro

Abstract. The constantly growing demand for 3D Massively Multiuser Online (MMO) virtual spaces has led to an increase of the load that current 3D MMO server architectures need to cope with in order to accommodate a large user base that interacts inside the virtual space at a given moment of time. The traditional methods that are employed to solve this computational challenge by the current 3D MMO server architectures have inherent drawbacks regarding scalability, reliability, redundancy, cost and the realism of the virtual world. In the last few years, the graphics hardware has evolved in a spectacular manner in terms of computing power and architectural design, switching from a “single-core” to current “many-core” architecture. In this paper we propose an innovative approach to tackle the heavy computational operations that are executed by the simulations of 3D MMO Virtual Spaces using Multi GPGPU (General Purpose programming on Graphical Processing Units).

Keywords: MMO, GPGPU, CUDA, Virtual Spaces.

1 Introduction

Massive Multiplayer Online (MMO) games and also other types of virtual spaces such as virtual museums, virtual expositions, etc., are nowadays more and more popular and the number of worldwide users that access such virtual worlds is increasing constantly [1].

MMO virtual spaces are available online over the Internet and have as support a persistent virtual world which is accessed at the same time by hundreds or even thousands of users (Fig. 1).

The main objective of a 3D MMO virtual space is to create a high degree of immersion for the users that access the world. The prime feature that distinguishes between 3D MMO virtual spaces and other single-user or multiuser applications, that also simulate virtual worlds, is the fact that the first allows the interaction of a great number of users at the same time inside of a large virtual world.

However, the growing demand for 3D MMO virtual spaces has led to an increase of the load that current 3D MMO server architectures need to cope with in order to accommodate a large user base that interacts inside the virtual space at a given moment of time.

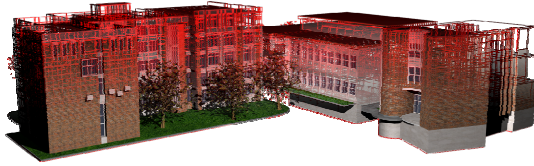


Fig. 1. Virtual space for the Faculty of Automatic Control and Computers

The traditional methods that are employed to solve this computational challenge by the current 3D MMO server architectures have inherent drawbacks regarding scalability, reliability, redundancy, cost and the realism of the virtual world.

This huge load is due not only because of the large number of tasks (directly proportional with the number of online users) but also by their inherent complexity that is closely related with the particularities of the virtual world.

In this paper, we start in the next section with an overview of the current architectures used by 3D MMO virtual spaces and we discuss some of the operations that are time and resource consuming. In section 3 we present why the GPGPU approach has been chosen for the implementation. In section 4 we propose a solution to implement some of the operations directly on the graphics hardware using a multi GPGPU approach and section 5 concludes the paper.

2 Current Solutions for 3D MMO Server Architectures

Even though some necessary requirements in order to use a MMO application in optimal circumstances are becoming more and more accessible for the normal home user (such as bandwidth), this is not sufficient to solve the scalability problems that MMOs have because of the necessity to accommodate as many users as possible at the same time.

The scalability problems are mainly due to the necessity to maintain the realism and consistency of the virtual world. Breaking the consistency requirements can lead for example to visual artifacts that don't have long term consequences but also can cause more serious problems such as the loss or duplication of objects during financial transactions.

Traditionally, the majority of MMO applications are implemented using a client-server architecture that has inherently advantages such as security and centralized control but also the following important disadvantages:

- Cost : because a large number of servers is used for the simulation of the virtual world, the maintenance costs after the launch can reach up to 80% of the revenues
- Reliability : servers can be points of failure in the system
- Scalability : in order to achieve horizontal-scalability a large number of servers is used to manage the virtual world but this approach after a certain point can also represent a performance bottleneck

As mentioned previously, an important aspect when designing the architecture is that MMOs must simulate virtual worlds that have a large physical span. This has led to the development of methods to split the virtual space into several distinct sub-spaces.

2.1 Zoning

Currently there are two different approaches to split the virtual world into sub-spaces. The simplest, from the implementation viewpoint, is to strictly partition the entire world into static zones that are small enough to be managed by a single server. The borders for these zones are very well determined, and a user that crosses into another zone will also connect to the server that manages the destination zone.

In the second approach the existence of borders, server zones and the transitions between them are transparent to the user (Fig. 2). Although it is not explicitly visible, the logical separation in zones for the virtual world still exists. Like the previous approach, important neighboring zones are managed by different servers. The main difference is represented by the existence of “transition zones” that have the role to “transport” the users between adjacent zones.

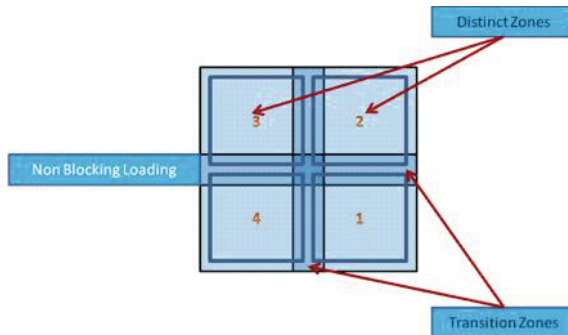


Fig. 2. “Seamless zoning”

2.2 Client-Server Model

Most of the MMO applications use Client-Server architecture (Fig. 3). The clients first access the virtual space through a Login Server that redirects them to a Shard Server.

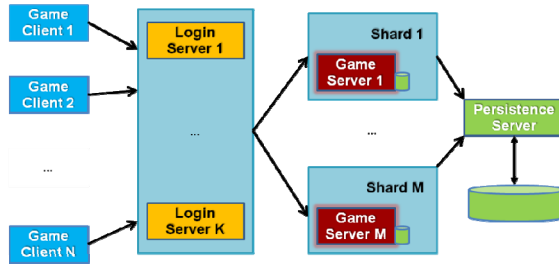


Fig. 3. Client-Server Shard Architecture for MMO Servers

The shards are independent versions of the same virtual world that run in parallel. This method of splitting the user base is used to achieve some degree of scalability. Usually the shards are not synchronized and users that are on different shards cannot interact. This fact contributes to a decrease in the realism of the simulated virtual world.

Some MMOs (for example the MMO Eve Online) try to implement “shardless” solutions that don’t split the user base and to allow all the users to be inside of a single huge virtual world. For the moment such approaches are very strongly dependent of the characteristics of the virtual world they try to simulate [2].

Taking into account all the challenges mentioned, it is a necessity to explore new solutions for the architectures used by MMOs and also to find new methods to optimize the operations executed inside the simulations of the virtual world in order to eliminate or at least to reduce to a certain degree the scalability problems.

3 Why GPGPU

In the last years, the computing hardware, CPU and GPU, have evolved dramatically in terms of computing power and architecture transforming into a very versatile hardware using parallel multi-core architectures [3]. These architectures are designed to prioritize operations that can be executed in parallel over a large quantity of data compared to the classic CPU architecture that prioritize single task operations that have low latency.

Type	Processor	Cores/Chip	ALUs/Core
GPU	AMD Radeon HD 5870	20	80
	NVIDIA GeForce GTX 480	15	32
CPU	Intel XEON Westmere	6	2
	CELL	8	4

Fig. 4. Modern GPU and CPU architectures

All the modern GPUs try to remain as efficient as possible by using multi-core designs that use hardware multithreading and SIMD processing. These techniques are not unique for the graphical processing units, but when compared with CPUs, the GPUs design takes these architectures to the extreme (Fig. 4).

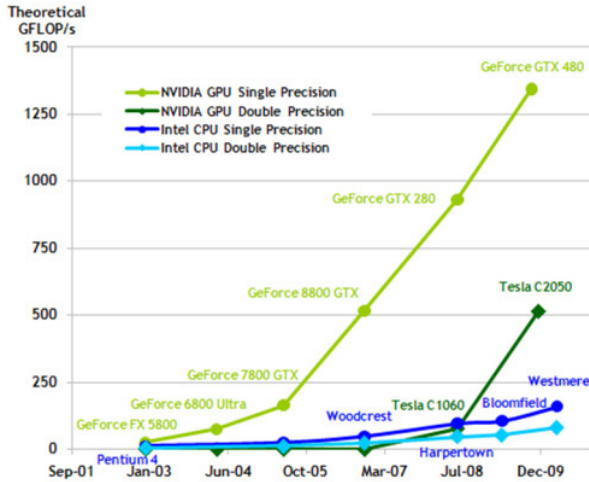


Fig. 5. Comparison between NVidia GPUs and Intel CPUs. Source: NVidia CUDA SDK 3.2

4 Our Solution

Our solution tries to achieve scalability, both vertical and horizontal, and to reduce the limitations of the traditional 3D MMO server architectures by implementing an architecture with the following main important characteristics:

- Offload heavy computational operations from the CPUs of the servers and implement them as GPGPU programs
- It is designed to be massively parallel by supporting the scheduling of operations at multi GPGPU level
- Using an architecture that is event and task driven and that is centered around the location and actions of the user entities in the virtual world because they are the ones that consume computational power and not the data of the virtual world by itself

In our implementation we have used NVidia hardware and the CUDA development toolkit to implement the GPGPU operations [5]. CUDA (Compute Unified Device Architecture) is a hardware and software architecture for running general purpose computations directly on the graphics hardware in a parallel manner without using the graphics API.

Using CUDA, the GPU is available as an additional computational unit to the main CPU (the host). In this way, parts that are computationally intensive and that are suitable to parallel execution are offloaded from the application that runs on the host directly on the GPU in order to increase the overall performance [6].

This part of the application is structured as a function (kernel) that is executed simultaneously on a large number of execution threads resulting in a SIMD processing of the operations.

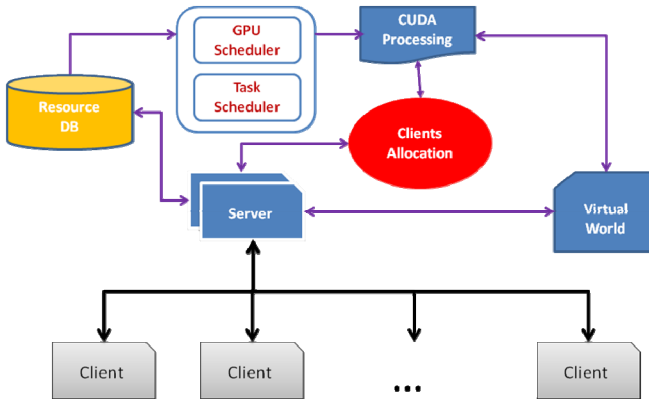


Fig. 6. Adaptation to the server architecture to run multi GPGPU

The following operations have been implemented using GPGPU techniques:

- The operations needed to simulate realistic physics for the virtual space

In order to accommodate the GPGPU computations in the MMO server architecture, the following modules have been implemented and are used by the virtual world server (Fig. 6):

- Client Allocation Module : allocation of the clients on the GPU using several heuristics criteria
- Scheduler module :
 - GPU Scheduler: allocation of tasks on the GPUs available
 - Task Scheduler : creation and scheduling of the CUDA / CPU tasks
- CUDA Processing Module : execution of the tasks as GPGPU programs on the graphics hardware; propagation of the results

4.1 Clients Allocation Module

This module is responsible with the allocation of clients and of the necessary resources to execute the computations at the server level.

For the allocation, the following three criteria / factors are used:

1. **Spatial positioning** of the clients that are online at a given moment of time inside the geography of the virtual space. Using this factor, if in a certain logical zone there is a greater number of users compared with another logical zone of the virtual space, more computational resources will be allocated to the first one in order for that zone to be able to process the increased number of operations that emerge from the complex user interactions with the virtual space.
2. **Type and number of operations** that are executed by the clients of the virtual space. In this way, the allocation for the processing of the tasks generated by the clients is done taking into account an “operations weight cost” that is associated to every user that is online inside the virtual space. Thereby, the clients that generate an increased number of operations (either taking into account the total number or the complexity of individual operations that can generate additional computations) will be able to benefit from additional resources that will be allocated for them in order to make sure that the system will process their operations in real-time.
3. **Type of user** that generates computations that will be processed by the system. Using this criterion, a different allocation can be made for the “human” users (that will have priority) and the NPC (non player characters) users that are controlled by the system.

4.2 Scheduler Module

This module is responsible with the creation and scheduling of the tasks that will be executed by the 3D MMO virtual space servers.

In our architecture, this module has to main components:

1. **Task scheduler**: this component is responsible with the creation of the computational tasks that are associated to the operations executed inside the virtual space. Thus, at each iteration (*tick*) of the main loop of the simulation, this component processes the actions of the entities from the virtual space and creates the tasks that emerge from these actions putting them inside the global task queue.
2. **GPU scheduler**: this component is responsible with the scheduling and sending of the tasks to the GPU for the actual execution. For the GPGPU implementation, using the cost associated to each task, the scheduling of the tasks that will be executed by the streaming multiprocessors of the GPU is done in the following way :
 - a. First, establish the number of computational resources (GPU's) available in the system;
 - b. Taking into account the required internal *tick* and the number of computational resources, establish the total number of tasks that can be executed at each iteration of the main loop of simulation;

- c. Using the total number of tasks that can be processed, compute the size of the grid and blocks of threads that is necessary to execute the tasks;
- d. Extract from the global task queue, those tasks that will be scheduled in the current *tick*;
- e. Create the task array to be executed using the tasks determined at the previous step and send it to the GPU for actual execution;

4.3 CUDA Processing Module

This module receives the GPGPU tasks from the scheduling module and it is responsible with their execution on the processors of the graphics hardware.

In this module the following operations are executed:

- Implementation of the kernel functions that will be executed by the threads of the GPU scalar processors
- Memory transfer of the data from the host device to the GPU
- After all the tasks have been executed by the GPU, the reverse transfer of the memory (that now contains the results of the execution) from the GPU to the host device is being done
- Synchronization operations that are necessary in order to be sure that all the individual threads of the GPU have finished the processing

4.4 Results

For our implementation we have used the following software/hardware components:

- Visual Studio 2008
- Boost library
- CUDA Development Toolkit 3.0.
- Intel Core2Quad 6600 Processor
- 2 x 8800 Ultra NVidia graphics cards

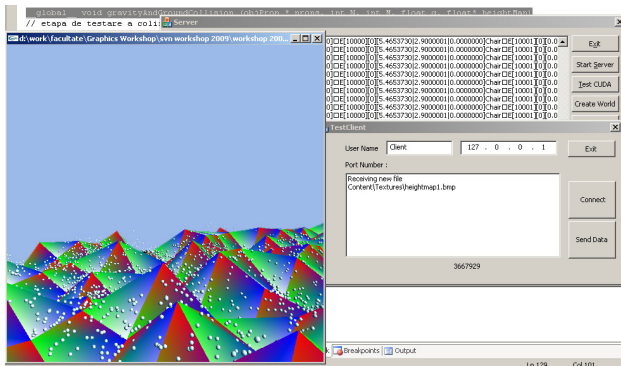


Fig. 7. Server running GPGPU physics tasks for 4096 users simulated as spheres

Using the above prototype architecture and modules we have achieved encouraging results, running simulations that behaved with real-time response for a great number of users (Fig. 7) on a single and multi GPU NVidia cards (8800 Ultra).

Table 1. Test results for the server running GPGPU physics tasks

Device	Server internal tick	Users
CPU Quad Core 4 Threads	60 FPS / 16 ms	<= 256
NVidia 8800 Ultra 128 SPUs GPGPU using CUDA	60 FPS / 16 ms	<= 4096
2xNVidia 8800 Ultra 128 SPUs GPGPU using CUDA	60 FPS / 16 ms	<= 8192

As can be seen from Table 1, in comparison with the CPU implementation, using the GPGPU approach has allowed for a far greater number of simulated users to be accommodated by the virtual world server architecture while maintaining the internal refresh tick of 60 FPS.

Another interesting and important result that can be seen from Table 1 is the fact that the multi GPGPU implementation scales very well when another card is available for processing.

5 Conclusions

Currently, Client-Server architectures provide the necessary functionalities required by a MMO application but with a high cost that limits practically the scalability of the virtual space forcibly splitting the world into several instances / shards.

In order to achieve a performance level that will satisfy a large number of users, the producers of virtual spaces are confronted with a significant financial cost to maintain the infrastructure of the system.

Our approach has proved that it is feasible to offload heavy computational operations from the virtual world servers as GPGPU programs reducing the overall effort that is necessary to run 3D MMO applications.

In terms of future scalability, taking into account the current evolution of GPUs, using GPGPU programs for 3D MMO servers can achieve an important degree of vertical scalability.

References

1. Carless, S.: Interview: Screen Digest on Subscription MMO Growth, Blizzard's Next (2009), http://www.gamasutra.com/view/news/23003/Interview_Screen_Digest_On_Subscription_MMO_Growth_Blizzards_Next.php (accessed February 14, 2011)

2. Eve Online (2003), <http://www.eveonline.com> (accessed on February 14, 2011)
3. Breitbart, J.: Case studies on GPU usage and data structure design. Dept. of Computer Science and Electrical Engineering. Universitat Kassel (2008)
4. Intel XEON Westmere Processors, <http://www.intel.com/products/workstation/processors/> (accessed on February 20, 2011)
5. CUDA Programming Guide, http://www.nvidia.com/object/cuda_develop.html (accessed on February 18, 2011)
6. Che, S., et al.: A Performance Study of General Purpose Applications on Graphics Processors. *Journal of Parallel and Distributed Computing* 68(10), 1370–1380 (2008)

Analyzing Social Networks Using Non-Metric Multidimensional Scaling

Florin Radulescu and Cristian Turcitu

Computer Science Department, University „Politehnica“ of Bucharest
060042 Bucharest, Romania
{florin.radulescu,cristian.turcitu}@cs.pub.ro

Abstract. Multidimensional scaling is a set of statistical techniques used for finding the underlying structure of high-dimensional data. In the non-metric approach, the data is ordinal, meaning there is a logical ordering between the input values. In this paper we will use non-metric scaling to discuss about how are online social networks perceived nowadays. A social network is a structure made of individuals which have common goals, hobbies, social status or are connected in some other way. We will concentrate in this article on social networks that are popular in Romania among people aged between 20 and 30.

Keywords: MDS, social network, proximity, similarity.

1 Introduction

Multidimensional scaling (MDS) consists of a multitude of techniques used for geometrically representing the relationships between entities having high dimensionality. It has its roots in the 50' in psychophysics ([1]) and psychometrics ([2]), being used to understand how people perceive similarity between different sets of objects. MDS is now used in several areas, like marketing, physics, social science and biology. Some aspects about multidimensional scaling and their use for analyzing social networks were presented in [10] and [11].

Based on a matrix (called proximity matrix) obtained from the dissimilarities or similarities between input objects, the aim of a MDS processing is to find a configuration of points which can map the initial proximities to distances of a lower dimension.

Given N – the number of input objects, $D = (d_{ij}) \in \mathbb{R}^{N \times N}$ the matrix of similarities or dissimilarities between objects, solving a MDS problem means finding a configuration of points $m < N$ dimensional points x_i such that:

$$d(x_i, x_j) = \|x_i - x_j\| = d_{ij}. \quad (1)$$

where $d(X_i, X_j)$ represents the distance between the points $X_i(x_{i1}, x_{i2}, \dots, x_{im})$ and $X_j(x_{j1}, x_{j2}, \dots, x_{jm})$ in space.

Distance can be defined as:

$$d(X_i, X_j) = \left(\sum_{a=1}^m |x_{ia} - x_{ja}|^p \right)^{1/p}. \tag{2}$$

where $p \geq 1$ can have several values. For $p=1$ the formula above is known as the Manhattan distance, for $p=2$, we have Euclidean distance and for $p > 2$ we talk about the p -norm Minkowski distance ([7]). In this paper we will refer to Euclidean distance when using the term “distance” to compare different objects.

Based on the input proximity matrix, the MDS algorithms can be classified as metric and non-metric. In metric MDS the input data is quantitative (interval or ratio data), while non-metric MDS uses ordinal data for the proximity matrix. We will focus on non-metric scaling in the following sections due to its wide usage in various fields.

In practice, it is quite hard to find the points $X_i \in R^m$ which can satisfy the relation (1), so the MDS algorithms try to minimize the difference between the proximities between input objects and the distance of the resulting points. In other words, the purpose of a MDS algorithm is to minimize the function:

$$\sigma(X, D) = \sum_{i < j} w_{ij} (d_{ij} - d(X_i, X_j))^2. \tag{3}$$

where w_{ij} is a weight function that can be used in some types of MDS.

1.1 Non-metric Multidimensional Scaling

When rank order between dissimilarities is more important in processing data than the numerical value of dissimilarities non-metric MDS can be used.

In non-metric scaling the proximities d_{ij} are replaced by disparities, which are sometimes called pseudo-distances because they are not real distances, they are obtained from dissimilarities using a function f :

$$f(d_{ij}) \approx \widehat{d}_{ij}. \tag{4}$$

which is monotonic increasing, meaning that for a pair of points a, b , if $a < b$, then $f(a) \leq f(b)$.

For non-metric MDS the function from (3) becomes:

$$\sigma(X, \widehat{D}) = \sum_{i < j} w_{ij} (\widehat{d}_{ij} - d(X_i, X_j))^2. \tag{5}$$

where \widehat{D} represents the matrix of disparities \widehat{d}_{ij} .

Therefore, finding a non-metric MDS solution comprises two steps:

- finding a monotonic increasing function which transforms the dissimilarities into disparities;
- minimizing the stress function from (5);

For solving the first step, in [3] and [4] are proposed the use of pooled-adjacent-violator algorithm (PAVA). For further details and improvements on this algorithm, article [5] can be consulted.

Minimizing the stress function is a complex problem which requires proper algorithms. According to [6], one common approach is SMACOF (Scaling by Majorizing a Complicated Function) – an algorithm that uses iterative majorization to reach a satisfactory solution.

The main idea of iterative majorization is to replace a complicated function with another one, for which the minimization process is simpler (see also [8]).

SMACOF algorithm for non-metric scaling has the following main steps:

- choose an initial configuration of points and calculate the stress function;
- find an update for the initial configuration;
- calculate the disparities \hat{d}_{ij} for the actual configuration;
- evaluate the stress function: if satisfies the desired criterion then stop, otherwise continue from the second step;

There are various implementations and upgrades for the SMACOF algorithm – one of them is the PROXCAL module implemented in the statistical software of IBM: SPSS.

In the following section we will use the SPSS tool for an evaluation of some social networks which are popular these days.

2 MDS Utilization in Social Networks

A social network can be described as being a group of persons who are connected by at least one common link and they know one another. They may have same passions, same goals, or they may be fighting for a common cause – either way, it's easy to identify social networks around us.

Social networks on the Internet are web sites built with the same purpose: of creating groups with similar interests that can communicate easily, no matter where are the people from. The trend of being affiliated to a social network is well known these days, especially among the youngsters. Using non-metric MDS we will discuss about how a social network is perceived nowadays.

2.1 *Specifying the Input Data*

We have selected for our experiment a number of seven social networks that are well known in the geographical area where we live. The social networks that we have chosen are: Twitter(Twt), Netlog(Ntg), Linkedin(Lkn), Hi5(Hi5), Friendster(Fst), Flickr(Fcr) si Facebook(Fbk).

As one can see, we have chosen an abbreviation for every social network – these will be used in the following sections, if necessary. For the input objects we have created a survey with each pair of social networks and we asked participants to rate their appreciation on the dissimilarities between the objects of each pair. For rating we used a scale from 1 to 5, a grade closer to 5 meaning that the social

networks compared are quite different, while a mark closer to 1 would mean that the compared objects are perceived as being similar.

At the experiment participated 25 persons, each having to grade a number of $7(7-1)/2 = 21$ pairs of social networks with marks from one to five.

2.2 Using SPSS PROXSCAL for Analysis

For interpreting the input data we will use the module PROXSCAL of the last version of the IBM software: SPSS 19. SPSS is one of the computer programs that are widely used for statistical analysis in social sciences (Wikipedia, 2011b).

We have selected PROXSCAL module because it implements the SMACOF algorithm that we previously discussed. PROXSCAL has also several functionalities that helped us to specify the number of iterations, the initial configuration of the solution matrix, or the value of the stress at which the algorithm would stop.

The module uses several types of input: full symmetrical matrix, lower or upper triangular matrix, rectangular matrix. For our example, a lower triangular matrix will be created from the data provided by every participant in order to have the appropriate input for the PROXSCAL procedure.

For the input data two types of analyses will be made. The first one is an individual analysis, for every participant at the experiment. After that, we will proceed to a replicated analysis, with all the input matrices of the participants.

2.3 Individual Processing of Data

In this section we will show how the subjects participating at the experiment differentiate between the social networks specified.

At first we have to select an input matrix from the 25 available, let's say the one from the 11'th participant. The proximity values for this subject can be visualized in Table 1 below. As we stated before, we have a lower triangular matrix, with zeros on the main diagonal.

Now we have to establish how many dimensions the solution will have. For this purpose we will use the scree plot of theoretical data.

Table 1. Input matrix for the eleventh participant

	Twt	Ntg	Lkn	Hi5	Fst	Fcr	Fbk
Twt	0						
Ntg	1	0					
Lkn	5	5	0				
Hi5	3	3	5	0			
Fst	1	5	5	3	0		
Fcr	3	4	5	4	3	0	
Fbk	2	4	5	4	2	2	0

As shown in figure 1, the stress decreases as the dimensionality of the solution increases. For a value of stress below 0.02, a two-dimensional solution can be chosen. After four dimensions, the value of stress has no major reduction.

For the two-dimensional space the distribution of objects for the 11'th participant can be observed in figure 2. The first observation is that LinkedIn is isolated from the others – this might mean that the source believes this object is quite different from the others. In the main cluster, Netlog and Hi5 are quite close, revealing the fact that they are perceived as being rather similar.

Two others social networks, Friendster and Flickr, are really close even though there are some differences between them: Friendster focuses on online games while Flickr is known for its video and images uploading facilities. One possible explanation might be that the participant is not very familiar with these two social networks, so he perceived them as being similar.

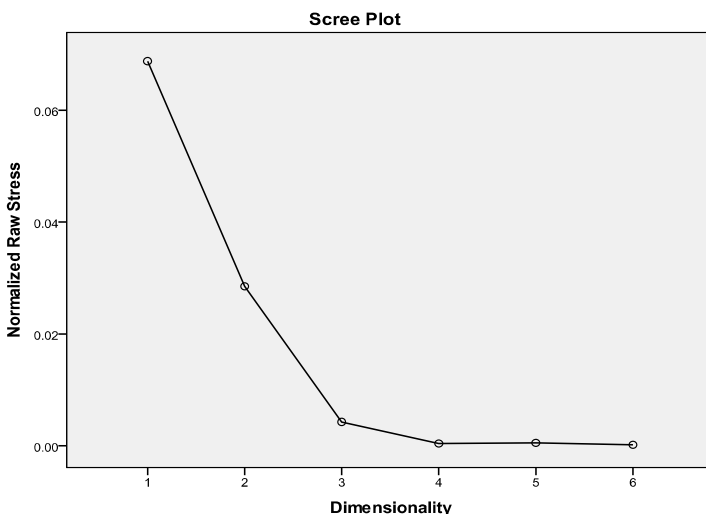


Fig. 1. Scree plot for eleventh subject

The interpretation above is valid for that subject based on the input matrix. If the information is provided from another source, the results may be different.

In the following section we will use the input data from all sources in order to interpret the results.

2.4 Replicated MDS Processing

Apart from individual scaling, PROXSCAL program can perform replicated analysis too. Replicated MDS is a technique that uses several proximity matrices as input data simultaneously.

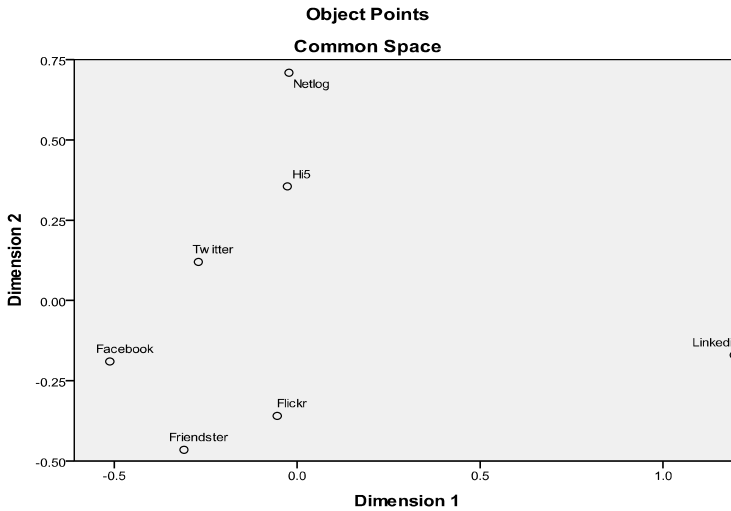


Fig. 2. MDS representation for the eleventh participant

RMDS assumes that the final dimensions are the same for the input matrices, and that every subject had the same stimuli when he participated at the experiment.

In RMDS, the stress formula becomes:

$$\sigma(X, \hat{D}) = \frac{1}{s} \sum_{k=1}^s \sum_{i < j} w_{ijk} \left(\hat{d}_{ijk} - d_k(X_i, X_j) \right)^2 \tag{6}$$

where s represents the number of input sources (proximity matrices), w_{ijk} is the weight between objects i and j from the k -th input matrix, and \hat{d}_{ijk} is the dissimilarity between objects i and j , in the matrix number k .

In our example, we combined the 25 proximity matrices and used them as an input for the PROXSCAL algorithm.

As we can see, for a stress value around 0.03, a two-dimensional solution is appropriate. The solution of RMDS can be observed in figure 4.

We can see that the replicated solution differs from the individual one – in this case there is a uniform distribution of the objects. A first observation might be that the social networks LinkedIn and Flickr are far apart on the second dimension, which means the participants perceive this objects as quite different.

Three networks, Facebook, Twitter and Netlog are slightly grouped together, meaning that the general perception is that they are seen as having similar functionalities. Friendster and Hi5 can also be grouped together according to the figure below; this might be a consequence of the fact that both have a platform for online games and this facility helps users to associate this social networks.

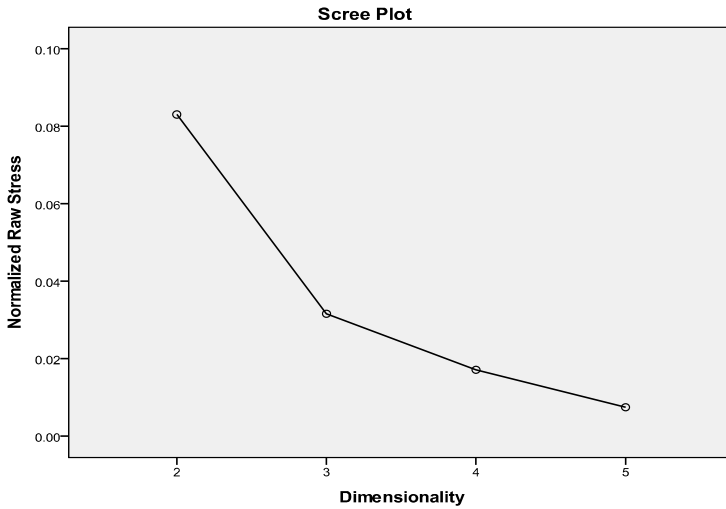


Fig. 3. Scree plot for RMDS

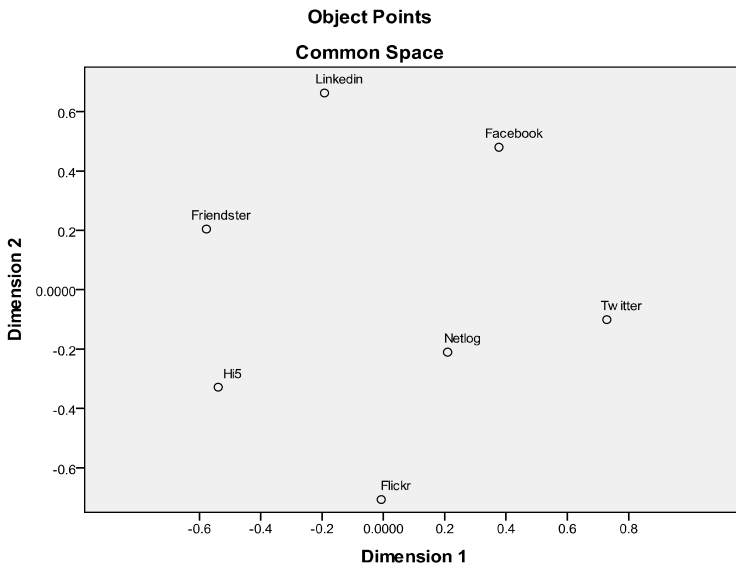


Fig. 4. Replicated MDS representation

Comparing the two methods of interpreting MDS data, we could say that replicated scaling provides a more accurate solution than the individual one. Individual scaling should be used when studying a specific problem, while replicated MDS may be used to analyze a more general view of the problem.

3 Some Inconveniences of Non-metric Scaling

One of the first disadvantages of using MDS is the large amount of data to be gathered and then processed. For example, if a study on N objects has to be made, a dissimilarity matrix for one source must have at least $N(N-1)/2$ values if the matrix is symmetric, or N^2 if it is not. Multiply this value by the number of participants at an experiment and the result is impressive.

Another disadvantage is that in many cases the input data is provided by human sources and they might make more or not deliberate mistakes, which can lead to abnormal results.

A final important issue is to search and exclude the proximity matrices if they do not contain any information in the data, meaning all dissimilarities are equal.

4 Conclusions

MDS is an exploratory set of techniques that can be used in various fields. In this paper we used non-metric scaling to analyze the perception of youngsters on social networks.

SPSS PROXSCAL implementation is a very flexible approach, having various options to choose from before starting the processing of data. The interpretation of the output is challenging and highly subjective – other persons would have probably found slightly different approaches to the resulting solutions.

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. *J. Mol. Biol.* 147, 195–197 (1981)
2. Guttman, L.: A new approach to factor analysis: the radex. In: Lazarsfeld, P. (ed.) *Mathematical Thinking in the Behavioral Sciences*, New York, pp. 258–348 (1954)
3. Young, G., Householder, A.S.: A note on multidimensional psycho-physical analysis. *Psychometrika* (1941)
4. Stephard, R.N.: The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. *Psychometrika* 27, 125–140 (1962)
5. Kruskal, J.B.: Nonmetric multidimensional scaling: a numerical method. *Psychometrika* 29, 115–129 (1964)
6. Zhou, W.: A review and implementation of some approaches to metric clustering (2004)
7. Borg, I., Groenen, P.: *Modern multidimensional scaling: theory and applications*, 2nd edn. Springer, New York (2005)
8. Wikipedia, Distance in Euclidean space (2011), <http://en.wikipedia.org/wiki/Distance>
9. Wikipedia, Stress majorization (2010), http://en.wikipedia.org/wiki/Stress_majorization

10. Wikipedia, Statistics Program (2011),
<http://en.wikipedia.org/wiki/SPSS>
11. Turcitu, T., Radulescu, F.: Using non-metric multidimensional scaling for analysing social networks. In: Procs. of the 18th Intl. Conference on Control Systems and Computer Science, Bucharest (2011)
12. Radulescu, F., Boicea, A., Ungureanu, D.: Using Fastmap for Distance Estimation in Web Search Engines. In: Procs. of the 17th Intl. Conference on Control Systems and Computer Science, Bucharest (2009)

Domain-Knowledge Optimized Simulated Annealing for Network-on-Chip Application Mapping^{*}

Ciprian Radu and Lucian Vințan

Advanced Computer Architecture & Processing Systems Research Lab
“Lucian Blaga” University of Sibiu, Romania
{ciprian.radu, lucian.vintan}@ulbsibiu.ro

Abstract. Network-on-Chip architectures are scalable on-chip interconnection networks. They replace the inefficient shared buses and are suitable for multicore and manycore systems. This paper presents an Optimized Simulated Annealing (OSA) algorithm for the Network-on-Chip application mapping problem. With OSA, the cores are implicitly and dynamically clustered using knowledge about communication demands. We show that OSA is a more feasible Simulated Annealing approach to NoC application mapping by comparing it with a general Simulated Annealing algorithm and a Branch and Bound algorithm, too. Using real applications we show that OSA is significantly faster than a general Simulated Annealing, without giving worse solutions. OSA proves to be feasible for Networks-on-Chip with more than 100 nodes. Also, compared to a Branch and Bound technique, it gives better solutions, as the problem size increases, while in terms of speed and memory consumption the two algorithms are comparable.

Keywords: Network-on-Chip (NoC), application mapping, clustering, optimization, evaluation, simulation.

1 Introduction

Simulated Annealing (SA) is a stochastic heuristic search technique, which simulates a system of particles that suffers changes in temperature. The idea comes from metallurgy, where annealing is the process used to temper or harden metals and glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to coalesce into a low-energy crystalline state. Kirkpatrick et al. [1] used Simulated Annealing to attack a classical NP-hard optimization problem,

^{*} A preliminary version of this paper was published at the 18th International Conference on Control Systems and Computer Sciences (CSCS-18), held in Bucharest, Romania, during 24 – 27 May 2011.

the travelling salesman problem, and they also applied SA to NP-hard computer design problems like partitioning, component placement and wiring of electronic systems.

Simulated Annealing is easy to implement. It has the ability of giving reasonably good solutions. Another advantage is its applicability for many combinatorial optimization problems. However, the parameters of the algorithm must be carefully chosen, since SA can easily run for a very long time until it gives a good solution. Because Simulated Annealing is a very general algorithm, several generic and problem-specific choices have to be made in order to implement it for a particular problem [2].

Simulated Annealing is used for the Network-on-Chip (NoC) application mapping, too. The NoC application mapping problem is defined as the topological placement of the Intellectual Property (IP) cores onto the on-chip tiles [3]. This is an NP-hard problem because, theoretically, there is a factorial number of possible mappings of c IP cores onto a network with n nodes, $c \leq n$. Therefore, heuristic algorithms are required for addressing this problem.

We propose an Optimized Simulated Annealing (OSA) technique for the Network-on-Chip application mapping problem. We begin by presenting some related work. After OSA is described, our simulation methodology is presented. Next, we present OSA's performance in terms of speed, memory consumption and solution quality, by comparing it with a general Simulated Annealing and a Branch and Bound (BB) algorithm. Finally, our conclusions and directions for future work are presented.

2 Related Work

Simulated Annealing was one of the first algorithms proposed for the Network-on-Chip application mapping problem [3]. Using an analytical model that estimates the energy needed to communicate a bit of data, the authors proposed an energy-aware mapping for square 2D mesh NoCs, with XY routing and wormhole switching. Simulated Annealing was compared with a Branch and Bound technique. Both these heuristic algorithms are bandwidth constrained and try to find the best solution by minimizing the communication energy. Hu and Marculescu showed that SA is capable of finding better mappings than the ones found using Branch and Bound. However, SA has the big disadvantage of being very slow. Their simulation results show that BB is tens of times faster than SA (e.g.: for a 10x10 NoC, SA did not finish its execution in 40 hours!). Both SA and BB algorithms were further developed in [4], where the mapping problem is treated in conjunction with the routing problem.

Little research has been done to optimize Simulated Annealing for mapping cores onto NoC tiles, using domain-knowledge. Cluster-based Simulated Annealing (CSA) [5] tries to exploit the application's communication locality so that it can identify clusters of IP cores. Firstly, the NoC is clustered by grouping its nodes based on the distance between them and their connectivity. Secondly, cores are clustered based on communication. A core cluster is then associated with each NoC node cluster. At high temperatures, the annealing process occurs normally:

any cores may be moved. However, when the temperature decreases enough, the annealing process is limited to intra-cluster cores. CSA's clustering technique is static and it is driven first by the NoC topology and second by the application. Clustering is an example of a problem-specific choice for Simulated Annealing. It improves SA's speed because it uses more knowledge about the NoC application mapping problem than a general Simulated Annealing.

We propose next an Optimized Simulated Annealing algorithm for NoC application mapping that performs an implicit and dynamic clustering, during the entire annealing process. As it will be shown next, OSA is significantly faster than both SA and CSA. OSA is application and network dependent. As compared to CSA, OSA does not cluster the NoC nodes because we do not want to restrict the core clustering process. Only the IP cores are clustered, and not explicitly but, implicitly. We rather influence the moves during the annealing process, so that the IP cores that communicate with each other clump together.

3 Optimized Simulated Annealing

Optimized Simulated Annealing was evolutionary created by continuing the work of Hu and Marculescu. Their Simulated Annealing and Branch and Bound algorithms are available through the NoCmap project [6]. We ported their two algorithms into our developed unified framework for the evaluation and optimization of NoC application mapping algorithms (UniMap) [7]. UniMap's goal is to allow the evaluation and potential optimization of different application mapping algorithms for NoCs, under a common frame [8]. OSA is a Simulated Annealing approach that, using domain-knowledge, is optimized for the Network-on-Chip application mapping problem, by applying and adapting some of the best practices for Simulated Annealing used in task mapping problems [9].

The pseudocode for our Optimized Simulated Annealing is presented in Figure 1. It is derived from the general Simulated Annealing proposed in [9].

The mappings produced with OSA are evaluated in terms of energy consumption, using the bit energy analytical model from [3].

We have chosen for OSA the geometric annealing schedule because this is the most used and recommended one [2] and because the general SA implementations use it too.

OSA works by default with an initial temperature of $T_0 = 1$ but, this is considered an algorithm parameter. The final temperature is fixed to $T_f = 0.001$. It is correlated with the acceptance function. In contrast, Hu and Marculescu's SA starts from a temperature of 100 and the final temperature is not bounded (a different stop criteria is implemented).

The general Simulated Annealing sets L , the number of iterations per temperature level, to $100(N \times N)^2$, where $N \times N$ represents the NoC 2D mesh size. For example, for a 4x4 2D mesh, SA tries $L = 25600$ mappings, at each temperature level. These mappings are randomly generated, from the current mapping, by

```

Require:  $M_i \neq 0$ 
Ensure:  $T_0 \geq 1$ 
 $M \leftarrow M_i$ 
 $C \leftarrow \text{BitEnergyCost}(M_i)$ 
 $M_{best} = M$ 
 $C_{best} = C$ 
 $T_f = 0.001$ 
 $R = 0$ 
for  $i = 0$  to  $\infty$  do
  if  $i \% L = 0$  then
     $R = 0$ 
  end if
   $T = T_0 \cdot 0.9^{\lfloor \frac{i}{L} \rfloor}$ 
   $M_{new} = \text{PDFbasedSwapping}(M, T)$ 
   $C_{new} = \text{BitEnergyCost}(M_{new})$ 
   $\Delta C = C_{new} - C$ 
  if  $\Delta C < 0$  or  $\text{NormInvExpAccept}(\Delta C, T)$  then
    if  $C_{new} < C_{best}$  then
       $M_{best} = M_{new}$ 
       $C_{best} = C_{new}$ 
       $R = 0$ 
    else
       $R = R + 1$ 
    end if
     $M = M_{new}$ 
     $C = C_{new}$ 
  else
     $R = R + 1$ 
  end if
  if  $T \leq T_f$  and  $R = L$  then
    break
  end if
end for
return  $M_{best}$ 

```

Fig. 1. Optimized Simulated Annealing

interchanging two cores, or - if possible - by placing a core into an empty NoC node. If we consider that SA runs for 100 temperature levels, this leads to 2560000 generated mappings. An Intel Xeon processor from our HPC system [10] evaluates a mapping in 0.04 ms. Thus, in this case SA would run for about 102 seconds. On a 10x10 mesh, SA would require more than one hour running time. However, the general SA algorithm is not bounded by the number of temperature levels, and we observed that it easily runs for more than 150 levels of temperature for a 4x4 2D mesh. We argue SA's number of iterations per temperature level is very high and it has a deep impact on SA's speed. Also, we observe that this number is only NoC aware. It is by no means application aware. For example, mapping 15 or 16 cores on a 4x4 2D mesh uses the same L .

Considering the above simple observations, we use in OSA the following relation to compute the number of iterations for each temperature level:

$$L_{OSA} = {}_n C_2 - {}_{n-c} C_2 = \frac{c(2n - c - 1)}{2}, c, n \in \mathbb{N}^*, n \geq c. \tag{1}$$

We noted with n the number of NoC nodes and with c the number of cores to be mapped.

This number of iterations per temperature level represents the total number of mappings that can be obtained from a given mapping, by performing a single core swapping. When $c = n$, a core swapping is an interchange of two cores. When $c < n$, a core may also be moved on another empty NoC node. . Because we noted the number of NoC nodes with n , taking into account that that SA has $L = 100(N \times N)^2$, we can write that $L_{SA} = 100n^2$. It is obvious that $L_{OSA} < L_{SA}$.

Also, in terms of algorithm’s speed, we note that OSA speedup over SA is:

$$S = 1 - \frac{L_{OSA}}{L_{SA}} \Rightarrow \lim_{n \rightarrow \infty} S = 99.5\%. \tag{2}$$

We considered that $n = c$, which implies a maximum (worst case) L_{OSA} . This is in perfect concordance with our further quantitative results.

L_{OSA} counts all mappings that are obtained by making a single modification (core swapping) to the given mapping (otherwise, the total possible mappings are of course ${}_n P_c \gg L_{OSA}$). The set of mappings obtained through a single core swapping form what we call the mapping’s immediate neighborhood. By analogy with Markov chains, OSA’s number of iterations per temperature level can be associated with the number of possible single step transitions from a Markov chain, which describes the mapping state space exploration performed by our algorithm. Later on, we will show how OSA assigns probabilities for each single step transition, using the concept of Probability Distribution Function (PDF). One may also observe that OSA’s number of iterations per temperature level is both NoC and application aware: n is a NoC topology characteristic and c is an application characteristic.

Some criticism of this approach might be that, although huge speedup can be obtained with L_{OSA} , OSA might find worse solutions because it does less exploration of the search space. We argue that the general SA can easily repeat a lot of moves without finding better solutions. Additionally, OSA considers the initial temperature a parameter, which can be increased so that the algorithm does more exploration. We will sustain our qualitative assessments through simulations.

Both general SA and CSA algorithms use the Metropolis acceptance probability function. OSA however uses the normalized inverse exponential acceptance function because it is shown in [9] that it yields better results when it is used for task scheduling. The practical difference between the two functions is that while the exponential form always accepts a new mapping that has exactly the same cost like the current mapping, the (normalized) inverse exponential form accepts such a new mapping with a 50% probability. Therefore, OSA knows to select with equal probability between two equally good mappings.

OSA's acceptance function performs cost normalization by dividing the cost variance (ΔC) with the initial mapping cost (C_0). This allows the temperature T to be independent of the cost function: $T \in (0, 1]$. Remember that OSA usually sets the initial temperature to 1 and the final temperature to 0.001. However, OSA allows the initial temperature to be higher than one. Regarding the normalized inverse exponential function, this would mean the initial cost is artificially increased.

The SA algorithms presented in our related work use random core swapping. A core is selected uniform randomly and it is then swapped with another uniform randomly selected core (an empty node can also be used). OSA does not use a uniformly random probability when determining the core to be moved. Instead, it adapts the variable grain single move (based on probability densities and used for task mapping [9]) into a variable grain swapping move, which uses two Probability Density Functions (PDFs). Based on the amount of data communicated by each core, OSA creates a PDF for each one. Thus, a core that communicates more data, has better chances to be chosen than a core that communicates less data. As the annealing temperature decreases, the probabilities uniformly equalize. Through this approach, OSA uses domain-knowledge (dynamic characteristics) to explore the search space.

$$P[\text{SelectedCore} = i] = \frac{1}{c} + \frac{T}{T_0} \left(\frac{\text{coreToComm}_i}{\text{totalToComm}} - \frac{1}{c} \right). \quad (3)$$

In formula (3), c is the number of cores to be mapped, T and T_0 are the current and, respectively, the initial temperatures, coreToComm_i is the amount of data communicated by core i and totalToComm is the total data communicated by all cores.

The second core used for swapping is selected by accounting for the communication volumes between the core to be swapped and the rest of the cores. Each core gets such a PDF associated before the annealing process starts.

$$P[c_i \leftrightarrow c_j] = \frac{\text{comm}_{ij}}{\text{totalComm}}. \quad (4)$$

In formula (4), comm_{ij} is the communication volume between cores i and j (this value is positive if core i sends data to core j , or vice versa; otherwise, it is zero) and totalComm is the data amount communicated by the entire application.

According to the PDF described above, the second core is selected. Then, OSA searches, in a uniformly random way, for a direct neighbor of the second selected core. This one will be swapped with the first core. This kind of move tries to make communicating cores attract each other, to naturally cluster themselves.

Compared to CSA, our algorithm clusters the cores dynamically, during the annealing phase. OSA does not work with predetermined clusters, and it also does not cluster the NoC nodes. Network-on-Chip node clustering is unneeded because OSA looks in the NoC node's neighborhood.

This kind of move is what we call a *PDF-based swapping move*. At every temperature level, OSA performs exactly L_{OSA} PDF-based swappings.

As recommended in [9], OSA employs a coupled temperature and rejection threshold stopping condition. The annealing process stops when the temperature reaches or goes below the final temperature and the last L_{OSA} tried mappings did not lead to a solution better than the best solution found so far. Therefore, OSA runs for a determined number of annealing temperatures, which can be exceeded while better solutions are found.

We note OSA implements a form of elitism because, during the entire running process, it stores the current best solution. This is another difference from the general Simulated Annealing technique.

Because OSA's stopping condition determines a number of annealing levels independent of the problem size, the runtime of our algorithm is quasi-constant when the algorithm is run more than once, in exactly the same conditions. This property does not apply to Hu and Marculescu's Simulated Annealing.

Compared to the general Simulated Annealing, our developed algorithm determines the number of iterations per temperature level by computing the neighborhood size of the current mapping. Also, OSA moves from one mapping to another using an implicit and dynamic clustering technique, based on Probability Density Functions. In contrast, CSA's clustering approach is explicit and static.

Currently, OSA works only with 2D mesh topologies but, it can be adapted to work with other NoC topologies, as well. Like Hu and Marculescu's Simulated Annealing, OSA is also capable to generate the routing functions, using a deadlock- and livelock-free approach, and to check if the obtained mapping meets the bandwidth constraints.

4 Simulation Methodology

This research uses the E3S benchmark suite [11]. It includes Communication Task Graphs (CTGs) [12] for five classes of real applications: automotive, consumer, networking, office and telecommunications. Because each application has only a few cores, we combined all Application Characterization Graphs (APCGs) [12] from each application domain and considered them to form a single system of several similar applications that need to be mapped on the same Network-on-Chip architecture.

Additionally, we work with some of the most used APCGs found in literature: Multimedia System - MMS (CTG 0) [4], MMS (CTG 1) [3], Video Object Plane Decoder - VOPD (CTG 0) [13] and several core graphs from SUNMAP [14]: Picture in Picture (PIP), MPEG4, Multi-Window Display (MWD) and VOPD (CTG 1). By manually creating the APCG from the CTG, we also introduced in [15] the H.264 decoder, with the tasks obtained through data partitioning - H.264 (CTG 0) and through functional partitioning - H.264 (CTG 1). These benchmarks have a number of cores ranging from 5 up to 30 [15]. Further details are available in [16].

We work with one of the most common Network-on-Chip architectures: a 2D mesh with regular tiles, using wormhole switching and XY routing. The NoC topology size is a simulation parameter. 2D meshes up to 15x15 in size were used

[15]. The NoC link bandwidth was set high enough so that bandwidth constraints are always met. The NoC communication energy values were taken from NoCmap.

In order to increase the simulations' accuracy, each application was mapped 1000 times, with each algorithm (SA, OSA and BB). For each simulation, the initial mapping was randomly chosen. To make the comparisons fair, the seed of the random number generator was set so that all algorithms start from the same search space point, every simulation.

For each mapping, its energy cost (in pJoule), algorithm runtime (user CPU time) and average heap memory consumption were recorded. All simulations were performed on our High Performance Computing system [10]: 15 nodes, each with 8 Intel Xeon cores and 5 GB of DRAM memory, running Red Hat Linux.

5 Results

In this section, we evaluate our Optimized Simulated Annealing by comparing it with the two algorithms from NoCmap: Simulated Annealing and Branch and Bound. We are interested in runtime, memory consumption and solution quality.

A runtime comparison between OSA and SA and, respectively, OSA and BB, are presented next. For each benchmark, the average of the 1000 runtime speedups is shown.

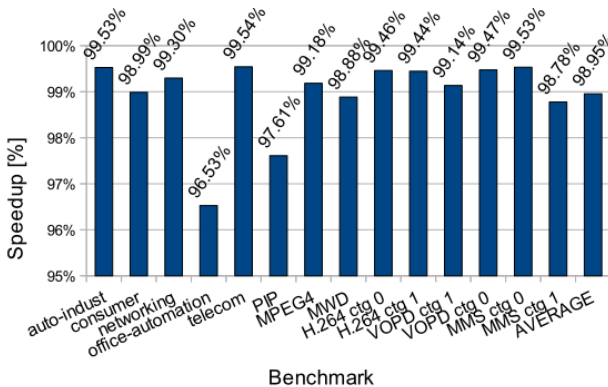


Fig. 2. OSA speedup over SA

Figure 2 shows that OSA is much faster than Hu and Marculescu's Simulated Annealing. An average speedup of 98.95% was obtained, which is in correlation with our theoretical expectations, derived from the comparison between L_{SA} and L_{OSA} – as it is shown in formula (2). The “lowest” speedups were recorded on the benchmarks with the smallest number of IP cores (office-automation and PIP). This significant speed gain is mainly justified by the way OSA computes the number of iterations per temperature level.

We showed in [15] that in comparison to Branch and Bound, OSA is slower on average by 24%. Compared to Branch and Bound, our algorithm obtained poorer runtimes on MPEG4 (more than twice slower), H.264 (1.5 times slower in both cases) and lower but comparable runtimes for PIP, office-automation, VOPD (CTG 1) and auto-indust. However, for half of the benchmarks, OSA is faster. We observed OSA being faster on the biggest benchmarks: 25% speedup for MMS (25 cores) and 41% speedup for telecom (30 cores).

We also showed in [15] how OSA’s memory consumption is, compared to the memory consumed by Simulated Annealing and Branch and Bound. Simulated Annealing consumes less memory than OSA, when mapping more than 16 cores. OSA manages to beat SA on several benchmarks with 16 cores but, on average, Simulated Annealing consumes with 13% less memory than our Optimized Simulated Annealing. However, compared to Branch and Bound, OSA takes a little bit less memory on average. Actually, we observed Branch and Bound’s tendency to grow its memory requirements as the problem size increases. For the telecommunication application OSA saves more than 33% heap memory than Branch and Bound.

Figure 3 compares the mappings found by SA and OSA. For each benchmark, we evaluate the 1000 mappings returned by the two algorithms and count how many times one algorithm returned better mappings than the other one (marked with “<” in the chart’s legend). Cases when both algorithms returned mappings with exactly the same cost are marked distinctively. We notice that both algorithms find the same “best solution”, after all 1000 runs, for benchmarks: networking, office-automation and PIP. For the last two of these three benchmarks, we confirm the solution is optimal because we applied an Exhaustive Search, too. Overall, OSA finds worse solutions than SA for 6 of the 14 benchmarks used in our simulations.

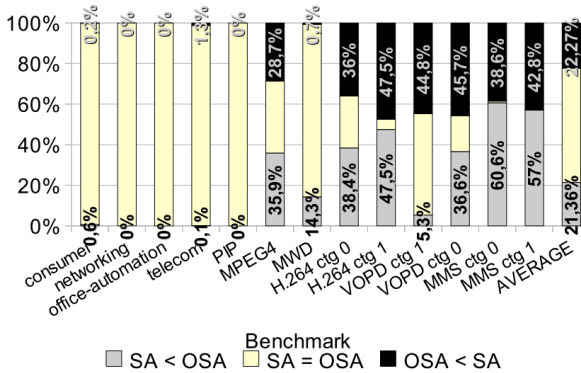


Fig. 3. OSA mappings cost, compared to SA mappings cost

We also compared only the best solutions found by each algorithm. We found out that SA and OSA always find the same best solution. However, Branch and

Bound fails to obtain the best mapping found by SA and OSA in two cases: for MMS (CTG 1), the energy lost with BB’s mapping is 0.1% and for auto-indust the energy loss is around 6%.

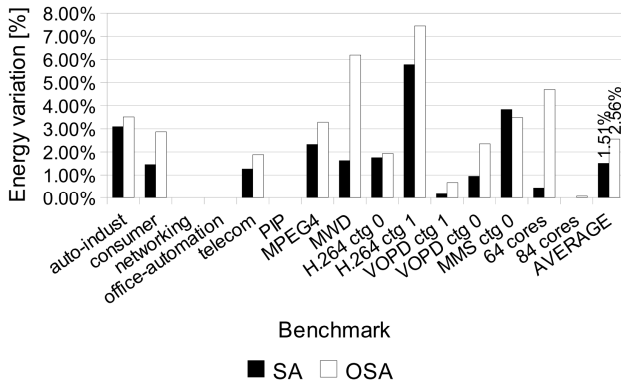


Fig. 4. SA and OSA energy variations (without MMS CTG 1)

Figure 4 shows a comparison between the energy variations generated by the worst and best mappings, found with SA and OSA, for all the benchmarks, except MMS (CTG 1). We excluded this benchmark because we obtained a 70% energy variation between SA’s best and worst mappings. With OSA the energy variation for this application was just 2%. For the rest of the benchmarks, OSA had, on average, an energy variation just one percent higher than SA’s average energy variation. Both algorithms have thus a small energy variation, which is less than 3%. However, if we also consider MMS (CTG 1), then SA’s average energy variation is 5.85%, while OSA’s average energy variation is just 2.53%.

We also showed in [15] how many times the best solution, given by all three algorithms, was found by each one of them. OSA finds the best solution more often than SA for several benchmarks: auto-indust, telecom, MPEG4, H.264 (CTG 0), VOPD (CTG 1). Branch and Bound outmatches OSA for the MMS benchmarks, VOPD (CTG 0), H.264 (CTG 1), MWD and consumer. Another observation is related to BB: it finds the best solution with probability 1 for all benchmarks, except auto-indust and MMS (CTG 1). We also observed that, on average, OSA finds the best solution a bit more frequently than Simulated Annealing.

We averaged the quality of the 1000 mappings per benchmark as well. Branch and Bound is the algorithm that, on average, gives the mapping with the smallest energy consumption. It fails just on auto-indust benchmark, where OSA provides the best average mapping cost. Optimized Simulated Annealing achieves for MMS (CTG 1) a far better average cost compared to Simulated Annealing: more than 34% energy gain is obtained. For the rest of the benchmarks, the differences between OSA and SA are less than one percent. Compared to BB, OSA provides solutions that are worse with no more than 2.5% on each benchmark, except auto-indust, where OSA is better with more than 6%.

Using 1000 simulations per benchmark, we have previously shown that the percentage of better solutions was lower for OSA than for SA on six benchmarks: MPEG-4, MWD, H.264 (CTG 0), MMS (both CTGs) and consumer. In order to get OSA’s percentage of better solutions over SA’s percentage, we reran OSA over the mentioned benchmarks, by increasing its initial temperature, until OSA gave a better percentage than SA. Raising the initial temperature makes OSA to run longer but, it also allows it to evaluate more mappings. Also, the higher the temperature is, the bigger is the probability to accept “bad” moves during the annealing process. By increasing the initial temperature, we managed to make OSA better than SA on all six benchmarks, except one [15]. For MMS (CTG 1), we were only able to reduce the difference between SA and OSA to half. Still, for MMS (CTG 1), OSA gives significantly better solutions on average. OSA’s speedup over SA remained very high even when it started with a very high initial temperature.

We were also interested to find out how our clustering technique affects OSA. We present next a comparison between OSA with and without clustering. The single thing that distinguishes OSA without clustering from OSA (with clustering) is that, in the first case, the simple random core swapping is used, without any restrictions.

Figure 5 shows how frequently the best solution is found. For all benchmarks, OSA with clustering finds the best solution more frequently than OSA without clustering. More than this, we observe significant differences for the benchmarks mapped onto the 4x4, 5x5 and 6x5 2D mesh NoCs. It is important to mention that the two OSA variants find the same best solution for all benchmarks, except MMS (CTG 1), where the best mapping found by OSA w/o clustering is 0.02% worse. On average, the best solution percentage for OSA with clustering is 18% higher than for OSA without clustering. Therefore, our implicit and dynamic clustering technique helps OSA to find the best mappings more often.

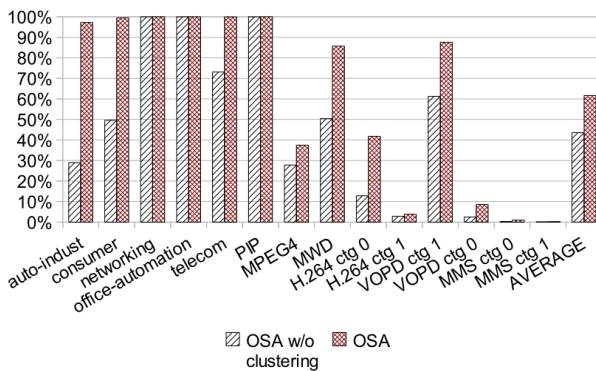


Fig. 5. The influence of OSA’s clustering on best solution percentage

We also measured in [15] how much energy is consumed, on average, by OSA without clustering (compared to OSA with clustering). It was noticed that, for all benchmarks, OSA without clustering generates mappings that determine additional energy consumption. Our clustering mechanism lowers the energy consumption with more than 1% in some cases. OSA with clustering always gives better average results (more than half a percent) than OSA without clustering.

Additionally, simulations on bigger benchmarks were also performed. Like in [5], we used four instances of the VOPD benchmark with 16 cores and we mapped them on an 8x8 2D mesh. SA ran ten times and OSA and BB ran 100 times. For Simulated Annealing we obtained an average running time of 12.65 hours (per simulation). Branch and Bound required just 114 seconds and OSA was 36% slower than BB. OSA consumes with approximately 39% less memory than Branch and Bound. The best mapping was found by Simulated Annealing. However, OSA's best mapping is only 0.7% worse.

Using all E3S benchmarks we obtained 84 cores that we mapped onto a 10x9 mesh. In this case SA required approximately 70 hours per simulation. Branch and Bound needed only 380 seconds (on average). OSA was 48% slower than BB. OSA consumed approximately the same amount of memory that Branch and Bound required (we noticed Branch and Bound prunes 85% to 93% of the explored search space). Simulated Annealing found the best solution but, it is better than OSA's best solution by only 0.09%.

Using the H.264 (CTG 1), MMS (CTG 0), MMS (CTG 1), MPEG4, MWD and VOPD (CTG 0) benchmarks, we obtained 97 cores (10x10 NoC). We used only OSA and BB (both were run ten times). Optimized Simulated Annealing ran, on average, approximately 15.9 minutes per simulation, being only 3% slower than Branch and Bound. The memory consumption was similar (40 MB).

By combining all non E3S benchmarks, we get a benchmark with 131 cores (12x11 NoC). OSA required, on average, approximately 51 minutes for mapping this application. Branch and Bound was 15% faster. In this case, OSA consumed less memory, 36 MB, while BB memory requirements were 14% higher. Optimized Simulated Annealing found a better mapping than BB, on each of the ten simulations.

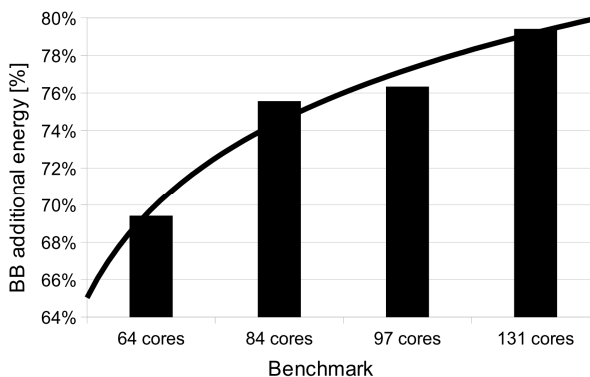


Fig. 6. BB mappings' additional energy, relative to OSA

Although Branch and Bound is faster than OSA, we obtained significantly worse solutions with BB. Figure 6 shows that the mappings found with BB are with more than 69% worse than the mappings found with OSA. For the 64 cores benchmark (four VOPD instances) we obtained a BB average mapping cost 69% worse than OSA's. For the 84 cores and 97 cores benchmarks, Branch and Bound gave, on average, a mapping cost around 76% worse than OSA's. For the 131 cores application OSA's solutions are 79.4% better than BB's solutions. We observe that, as the problem size increases, Branch and Bound finds mappings that require more and more energy (as compared to the OSA mappings).

Finally, we combined all of our benchmarks and obtained 215 cores (15x15 NoC). OSA ran for 8.4 hours and consumed (on average) 265 MB of memory, for each simulation. BB's runtime was more than half OSA's runtime. Memory consumption was also significantly lower: only 158 MB. However, all mapping attempts with Branch and Bound failed. Each time, BB did not finish mapping because it pruned more than 98.7% of the search space. We notice BB's memory consumption does not grow exponentially but, the quality of solution is heavily affected, up to the point where no solution is found.

6 Conclusions and Further Work

We presented in this paper an Optimized Simulated Annealing (OSA) algorithm for Network-on-Chip application mapping. Like Hu and Marculescu's Simulated Annealing, OSA is energy- and performance-aware. In contrast with their work, our approach uses application knowledge. Like Clustered-based Simulated Annealing, OSA also performs clustering but, implicitly and dynamically, not explicitly and statically, with certain performance benefits. In accordance with our theoretical expectations, OSA proved to be much faster than Hu and Marculescu's Simulated Annealing. We obtained an average runtime speedup of 98.95% while the quality of the mapping solution was not lost. While SA is not feasible for NoCs with more than 100 nodes, we showed OSA is feasible for NoC meshes with size higher than 10x10. OSA is also comparable to Branch and Bound in terms of memory consumption and speed. However, Branch and Bound failed to find better solutions on auto-indust and MMS (CTG 1) benchmarks. More than this, the mapping solution given by BB is more than 69% worse than the one found by OSA, when mapping cores onto a 2D mesh with size greater than 8x8. We also found that, due to an aggressive pruning, Branch and Bound is unable to map an application with 215 cores, onto a 15x15 Network-on-Chip.

On the 64 cores benchmark, we found OSA is 99.97% faster than CSA. However, the comparison between OSA and CSA runtimes is likely to be unfair. This can be due to several reasons: implementation language (OSA is written in Java but, we do not know how CSA is implemented), cost function (OSA is energy aware, while CSA is bandwidth and latency constrained). Also, CSA does not specify the number of iterations per temperature level. Still, we expect OSA to be faster than CSA because of the significantly high difference in runtime.

As further work, we intend to compare OSA's performance with that of different Evolutionary Algorithms (like, for example, Particle Swarm Optimization

[17]), which we will be adapted for the Network-on-Chip application mapping problem. All these algorithms will be integrated in UniMap and will be used for different NoC designs, through the Framework of Automatic Design Space Exploration [18].

Acknowledgments. This research was supported by CNCSIS no. 485/2008 research grant offered by the Romanian National Council for Academic Research and by POSDRU PhD financing contract POSDRU 7706.

References

1. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by Simulated Annealing. *Science* 220, 671–680 (1983)
2. Fouskakis, D., Draper, D.: Stochastic Optimization: a Review. *International Statistical Review* 70, 315–349 (2007)
3. Hu, J., Marculescu, R.: Energy-aware mapping for tile-based NoC architectures under performance constraints. In: *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, pp. 233–239. ACM, Kitakyushu (2003)
4. Hu, J., Marculescu, R.: Energy- and performance-aware mapping for regular NoC architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 551–562 (2005)
5. Lu, Z., Xia, L., Jantsch, A.: Cluster-based Simulated Annealing for Mapping Cores onto 2D Mesh Networks on Chip. In: *Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pp. 1–6. IEEE Computer Society, Washington, DC (2008)
6. SLD: System Level Design Group @ CMU: NoCmap: an energy- and performance-aware mapping tool for Networks-on-Chip, <http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:nocmap>
7. Radu, C., Vințan, L.: UniMap: Unified Framework For Network-On-Chip Application Mapping Research. *Acta Universitatis Cibiniensis Technical Series* (2011)
8. Radu, C., Vințan, L.: Optimizing Application Mapping Algorithms for NoCs through a Unified Framework. In: *2010 9th Roedunet International Conference (RoEduNet)*, pp. 259–264. IEEE Computer Society, Sibiu (2010)
9. Orsila, H., Salminen, E., Hämäläinen, T.D.: Best Practices for Simulated Annealing in Multiprocessor Task Distribution Problems. *Simulated Annealing*, pp. 321–342. I-Tech Education and Publishing KG (2008)
10. ULBS HPC cluster, <http://zamolxe.hpc.ulbsibiu.ro/>
11. The Embedded System Synthesis Benchmarks Suite (E3S) website, <http://ziyang.eecs.umich.edu/~dickrp/e3s/>
12. Marculescu, R., Ogras, U.Y., Peh, L.-S., Jerger, N.E., Hoskote, Y.: Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *Trans. Comp.-Aided Des. Integ. Cir. Sys.* 28, 3–21 (2009)
13. Murali, S., Micheli, G.D.: Bandwidth-Constrained Mapping of Cores onto NoC Architectures. In: *Proceedings of the Conference on Design, Automation and Test in Europe*, vol. 2, pp. 896–903. IEEE Computer Society, Paris (2004)

14. Murali, S., Micheli, G.D.: SUNMAP: a tool for automatic topology selection and generation for NoCs. In: Proceedings of the 41st Annual Design Automation Conference, pp. 914–919. ACM, San Diego (2004)
15. Radu, C., Vințan, L.: Optimized Simulated Annealing for Network-on-Chip Application Mapping. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS-18), pp. 452–459. Politehnica Press, Bucharest (2011)
16. Radu, C.: Optimized Algorithms for Network-on-Chip Application Mapping. “Lucian Blaga” University of Sibiu, Romania (2011)
17. Lungu, V., Sofron, A.: Using Particle Swarm Optimization to Create Particle Systems. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS-18), pp. 750–754. Politehnica Press, Bucharest (2011)
18. Calborean, H., Jahr, R., Ungerer, T., Vintan, L.: Optimizing a Superscalar System using Multi-objective Design Space Exploration. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, pp. 339–346 (2011)

A Comparison of Multi-objective Algorithms for the Automatic Design Space Exploration of a Superscalar System

Horia Calborean¹, Ralf Jahr², Theo Ungerer², and Lucian Vintan¹

¹ "Lucian Blaga" University of Sibiu, Advanced Computer Architecture & Processing Systems Research Lab, E. Cioran Str., No. 4, Sibiu - 550025, Romania

² University of Augsburg, Institute of Computer Science, Universitaetsstr. 6a, 86159 Augsburg, Germany

{Horia.Calborean, Lucian.Vintan}@ulbsibiu.ro,

{Jahr, Ungerer}@informatik.uni-augsburg.de

Abstract. In today's computer architectures the design spaces are huge, thus making it very difficult to find optimal configurations. One way to cope with this problem is to use Automatic Design Space Exploration (ADSE) techniques. We developed the Framework for Automatic Design Space Exploration (FADSE) which is focused on microarchitectural optimizations. This framework includes several state-of-the-art heuristic algorithms.

In this paper we selected three of them, NSGA-II and SPEA2 as genetic algorithms as well as SMPSO as a particle swarm optimization, and compared their performance. As test case we optimize the parameters of the Grid ALU Processor (GAP) microarchitecture and then GAP together with the post-link code optimizer GAPtimize. An analysis of the simulation results shows a very good performance of all the three algorithms. SMPSO reveals the fastest convergence speed. A clear winner between NSGA-II and SPEA2 cannot be determined.

Keywords: Automatic design space exploration, evolutionary and bio-inspired algorithms, particle swarm optimization, superscalar microarchitecture, simulation.

1 Introduction

During the last years we have witnessed a spectacular growth in the computer systems complexity. With it, the number of architectural parameters has also increased. This means that a huge number of possible configurations can be considered while designing a computer system. As an example, if there are 50 parameters, each being able to have one of eight possible values, it involves 8^{50} possible configurations. Simulating all these configurations to find the best possible solutions would take extremely long time for representative benchmarks. The time-to-market requirements will not allow this.

The current design methodology implies a designer who, based on his/her experience, manually chooses parameters for the architecture. As an alternative

solution for this hard job, automated search tools have been built to help the user find good configurations. Since optimizing a system is usually a multi-objective problem (most of the times with conflicting objectives) the task of finding a good solution becomes even harder.

In order to do this automatically, we have developed the Framework for Automatic Design Space Explorations (FADSE) introduced by Calborean et al. [1][2][3][4] and Jahr et al. [5]. FADSE is able to perform automatic design space explorations using state-of-the-art evolutionary and bio-inspired multi-objective algorithms.

In this article we compare three well known heuristic algorithms: two genetic algorithms (NSGA-II and SPEA2) and one particle swarm optimization (SMPSO). As an optimization problem for the algorithms we first use the Grid ALU Processor (GAP). Second, we test the algorithms on a design space exploration of both GAP and the code optimization tool called GAPtimize.

The article is structured as follows: Section 2 provides an overview of the related work. In Section 3, some basic concepts about design space exploration are presented as well as the metrics used in our experiments. The tools (FADSE, GAP and GAPtimize) are presented in Section 4 along with the objectives used and the parameters for the DSE algorithms. The results of our evaluation are presented in Section 5 and finally Section 6 concludes the paper and presents some further work.

2 Related Work

There is not too much work on comparing different algorithms on computer architecture simulators. Such a comparison is made in [6] with the M3Explorer. The authors compare some algorithms including NSGA-II and a particle swarm optimization algorithm [7][8]. NSGA-II is clearly the best performing algorithm from the selected ones and outperforms the particle swarm optimization algorithm. The design space of their exploration consists of only 9126 possible configurations which allowed them to perform an exhaustive evaluation (by simulating all possible configurations). In [9] a set of single objective algorithms (genetic algorithms, ant colony optimization, hill climbing, random search, etc.) is compared against an exhaustive simulation. The search space is greatly reduced by fixing many parameters to allow the authors an exhaustive search. The conclusion is that the genetic algorithm is able to reach a solution 0.1% worse than the best possible solution but 23000 times faster. Still, the work is not extended to multi-objective optimization.

The authors of NASA [10] perform a DSE using single objective genetic algorithms for each objective. Multiple algorithms are used at the same time and a comparison between sets of these algorithms is made.

There are also many articles comparing algorithms not on real world problems but on synthetic ones. For example, in [11] SMPSO is compared with NSGA-II and SPEA2. The conclusion which the authors reach is that SMPSO clearly outperforms NSGA-II and SPEA2. In [12] a comparison between SPEA2 and NSGA-II is performed but no clear winner is found.

From the perspective of DSE on optimizing computer architectures there are some other existing tools: Archexplorer [13] is a web-tool which allows users to (up)load implementations of their computer architecture components (e.g. caches) to be optimized. The modules have to respect a special interface so they can be integrated in the DSE framework. The type of components that can be integrated is limited to the available interfaces. The algorithm used for DSE is statistical exploration with genetic operators. The DSE algorithm cannot be changed and also no comparison between different types of algorithms is performed by the authors.

3 DSE Basic Concepts

In many of the real world optimization problems there are multiple (often) conflicting objectives. Hence, an order between individuals (solutions to the problem) cannot be established easily. One individual can be better than another individual on one objective but worse on another one. Therefore the notions from the concept of Pareto efficiency are used [5].

To be able to perform an unbiased comparison of the three algorithms metrics are needed.

Coverage [14] can be used, for example, to compare two multi-objective algorithms, or two runs of the same algorithm (with the same or different parameters). For each analyzed generation it returns the fraction of individuals produced by one algorithm that dominates individuals produced by the other algorithm.

Hypervolume [14], also known as hyperarea, computes the volume enclosed by the current Pareto front approximation and the axes, for a maximization problem. For a minimization problem a point has to be established, called hypervolume reference point, which will replace the origin in the hypervolume computation. The hypervolume reference point (HV) is the point set at the coordinates which are the maximum values of the objectives. The values returned by the hypervolume computation are normalized to the interval [0,1] using this constant area. If it is computed at each generation, the evolution of this volume shows if the algorithm makes progress (if it converges). Also, if multiple algorithms are compared, it can show the quality of the results of one algorithm over the other, but in this situation the maximum values have to be searched from all the different runs so that the area enclosed between the axes and the HV remains fixed.

Two set difference hypervolume (TSDH) [15] is defined as:

$$TSDH(X', X'') = H(X'+X'') - H(X''), \quad (1)$$

where $X', X'' \subseteq X$ are two sets of individuals (populations), $H(X)$ is the hypervolume of the space covered by population X , and $X'+X''$ is the population of nondominated individuals obtained after the union of X' and X'' . $TSDH(X', X'')$ computes the hypervolume of the space that it is dominated by X' but not by X'' .

4 Methodology and Tools

This section introduces the components of our setup.

FADSE (Framework for Automatic Design Space Exploration) allows distributed design space exploration of computer systems using multi-objective state-of-the-art design space algorithms. It was already presented in [1] and in [3]. Since the structure of the application has changed and many improvements have been made the current state is presented below.

FADSE was designed so that almost any existing computer system simulator can be connected to it by writing a specific connector. We have implemented connectors to computer architecture simulators like: GAP and GAPtimize (used in this article), M-SIM3, M-SIM2 [16], M5 [17], UniMap [18] and Multi2Sim v2 [19].

The framework has been designed as a client-server application. The server runs the DSE algorithm and the clients perform the simulations.

We have changed some of the algorithms implemented in jMetal by Durillo et al. [20] to work in a distributed manner.

FADSE has been tested successfully on a LAN of Windows machines, on a Windows based computer with 32 processor cores and on a Linux based HPC system (30 Intel Xeon quad-cores).

A database (currently MySQL) can be used to store the results of the simulations. The evolutionary algorithms tend to produce some of the individuals again during an exploration process. When the same individual is generated the results can be extracted from the database, instead of redoing the simulations, leading to a much shorter exploration time. In our experiments, on 100 generations with a population of 100 individuals per generation in a design space of around 1.1 million configurations with the NSGA-II algorithm, a reuse factor of 67% was observed.

A list of application-specific rules can be described. These rules are used to constrain the design space. Different rules can be designed: from simple conditions (e.g. L2 cache must be larger than L1 cache) to more complex ones. More details about FADSE can be found in [21] [22].

The Grid ALU Processor (GAP) introduced by [23] is a novel processor architecture designed to speed up the execution of sequential instructions streams. Its basis is a superscalar processor with in-order execution. The main change is replacing the execution units by an array of functional units (FUs). Instructions are mapped onto this structure dynamically and at runtime by an additional stage of the frontend, the configuration unit. Therefore, it is not necessary to re-compile a program to be able to execute it on the GAP; a common instruction set as for a superscalar processor can be used.

To buffer configurations, i.e. instruction sequences placed onto the array, a structure called configuration layers with large similarities to a trace cache has been introduced. With it the latencies for issuing instructions can be reduced effectively and hence the total system performance increases.

The GAP is very scalable because one can choose for the size of the array of FUs any size between 4x4 and 31x32. In the design space exploration we want to

find good sizes for the array (height, width) in connection with the number of configuration layers (1, 2, 4, ..., 64) as well as the size and organization of the instruction cache.

Two objectives were defined for the GAP so far. The first is the total system performance gained from a cycle accurate simulator. The number of clock cycles per instruction (CPI) is used as measure. As second objective we introduced a model (see [5]) to make the complexity of the hardware of two different configurations of the GAP comparable. This is referred to as complexity in the following. To it contribute mainly the instruction cache, whose size is calculated with CACTI, the number of FUs and the number of configuration layers. Both objectives are to be minimized.

With GAP's ability to execute legacy code, the best point to apply code-optimizations, which make use of platform-specific features, is a post-link optimizer. **GAPtimize** has been developed for this. It supports at the moment e.g. branch predication, an optimization called static speculation [24] and inline expansion of functions. In this paper, inlining is used as a prototype for a code optimization. Our heuristic has four parameters which form the parameter space to be explored by FADSE.

As objective function the number of clock cycles per reference instruction is used (CPRI). The number of clock cycles for the program is divided by the number of instructions in a reference run without code optimizations. This is because the optimizations can also vary the number of instructions which should not influence the objective. The optimization goal is the same as for CPI; smaller is better.

For the DSE process we selected **three algorithms**: two genetic algorithms NSGA-II [25] and SPEA2 [12] and a particle swarm optimization algorithm called SMPSO [11].

For all the DSE algorithms we used the same mutation and crossover (if required) operators, i.e. bit flip mutation and single point crossover. The selection operator for the genetic algorithms was binary tournament selection as described in [25]. The mutation probability was set for all the algorithms to $1/(\text{number of decision variables})$. The crossover probability was set to 0.9. These values are commonly used for the evolutionary algorithms. The population size (swarm size for particle swarm optimization) was set to 100 for all the evaluations and the archive size for SPEA2 and SMPSO was also set to 100, as recommended in [25].

For the velocity computation in SMPSO the following values were used (as recommended in [11]): $C1$ and $C2$ are randomly chosen in the interval $[1.5, 2.5]$, $r1$ and $r2$ are randomly chosen in the interval $[0, 1]$ and inertial weight W is 0.1.

We have generated a random initial population and all the algorithms start from this initial population. This provides a fair comparison of the algorithms. Without this, multiple runs would be needed and an average result should be presented to provide unbiased results. Multiple runs were infeasible due to time constraints.

5 Evaluation

In this section we present the evaluation results for all the algorithms on the GAP processor and respectively on GAP with GAPtimize.

5.1 FADSE with GAP

The DSE process on GAP was done with six microarchitectural parameters on ten benchmarks selected from the MiBench suite. The benchmarks were compiled with GCC with optimizations turned on (-O3). The design space has a size of over $1.1 \cdot 10^6$ individuals. The DSE was stopped after around 50 generations.

We first compute the coverage and compare the two genetic algorithms (see Fig. 1). It can be observed that NSGA-II performs better than SPEA2. The evolution is close for the first 18 generations but then NSGA-II clearly performs better.

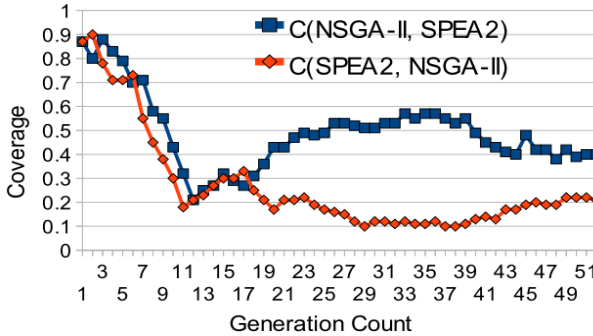


Fig. 1. Coverage comparison between NSGA-II and SPEA2

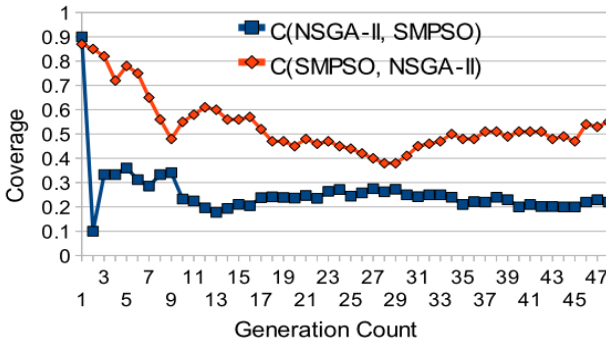


Fig. 2. Coverage comparison between NSGA-II and SMPSO

A coverage comparison between NSGA-II and SMPSO was made (see Fig. 2) and SMPSO obtains better results than NSGA-II for all generations (since NSGA-II is better than SPEA2 the coverage between SMPSO and SPEA2 is not presented). It can be observed in Fig. 2 that the coverage difference between SMPSO and NSGA-II is extremely high at the second generation. This happens because SMPSO converges faster and obtains very good results quickly while NSGA-II requires some time to reach the same quality of configurations.

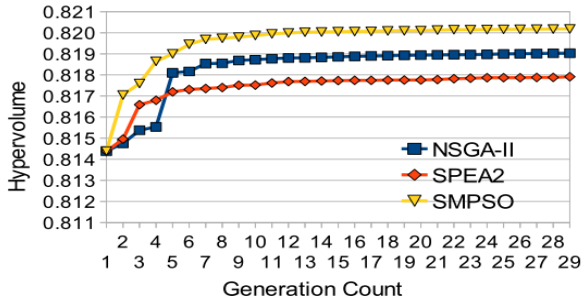


Fig. 3. Hypervolume comparison between NSGA-II, SPEA2, and SMPSO (Hypervolume axis has been trimmed for visibility)

Next we compare the algorithms using the hypervolume metric. The hypervolume shows the speed of the convergence of the algorithms. If the same reference point is used, when multiple runs are compared, it can also give a hint about the quality of results. From Fig. 3 we can conclude that SMPSO is still the best performing algorithm from the selected three: it finds the best solutions (highest value of the hypervolume) and also SMPSO converges the fastest to a better Pareto front approximation (hypervolume values rise faster than for the other algorithms).

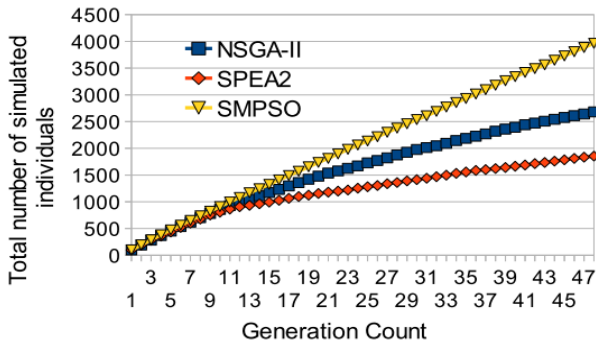


Fig. 4. Number of simulated individuals required to reach a certain generation

Since in design space exploration of computer architectures we are dealing with long evaluation times, it is necessary to compare the number of distinct evaluations required by each algorithm to reach a certain generation. In Fig. 4 it can be observed that SPEA 2 requires fewer evaluations to reach generation 50. SPEA2 tends to retain many duplicates in its archive during the environmental selection process (see [12]) partially because of the density computation method.

In SPEA2 identical individuals will have the same fitness assigned (only one neighbour is used to compute the density), while in NSGA-II these individuals can have a different fitness assigned. This happens because of the crowding

assignment [25]: two individuals are used to compute the crowding, one on each side of the current individual (sorted according to an objective), which means that twins will have one individual identical but the other one is a different one with a great probability. In SPEA2 there is a big chance that both individuals get selected and passed to the next generation while in NSGA-II one of them might be discarded.

During the selection process all the individuals have the same chance of becoming parents, but since in SPEA's archive there are more duplicates than in NSGA-II, they have a greater chance of being selected and produce the same individuals.

Another issue that leads to duplicates is the archive used in SPEA2. SPEA2 stores in the archive only the nondominated individuals. Parents are selected only from the archive but in our explorations we noticed that there are usually fewer nondominated individuals than the maximum size of the archive/population. So the archive will be quite empty (hence a greater chance to select the same parents again) while in NSGA-II the population is filled with a little worse individuals until it is full.

The advantage of SPEA2 is that these duplicated individuals do not have to be simulated again due to the integrated database. The disadvantage is that SPEA2 does not have such a good spread of solutions (even if at the end there are 100 individuals many of them are duplicates). SMPSO has a different approach since it does not create new individuals/particles and it only "flies" them through space. When a particle is moved all its parameters are changed. This behaviour of PSO algorithms prevents generating the same positions for a certain individual so often, but also increases the number of required simulations.

With this in mind, it is interesting to compare the hypervolume against the number of simulated individuals. In Fig. 5 we can see that the conclusion remains the same. SMPSO has the best performance from this point of view, too. At the same number of simulated individuals the hypervolume value of the Pareto front approximation, discovered by the particle swarm optimization algorithm, is superior to those obtained by the other two genetic algorithms.

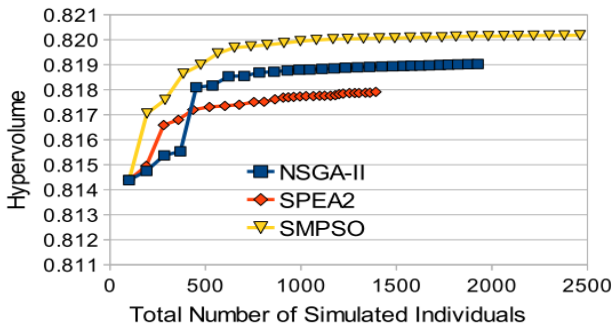


Fig. 5. Hypervolume comparison of the three selected algorithms against the total number of evaluated designs (Hypervolume axis has been trimmed for visibility)

The final results (after 50 generations) are shown in Fig. 6. The figure presents a portion of the Pareto front approximation. We have selected only this area because on the rest of the obtained Pareto front approximation there are no visible differences between the algorithms. It can be observed that SMPSO indeed finds better configurations but only on a small area, between a complexity of 100-180, while the complexities of all the solutions found by the algorithms range from 30 to 2000 (not shown in the figure).

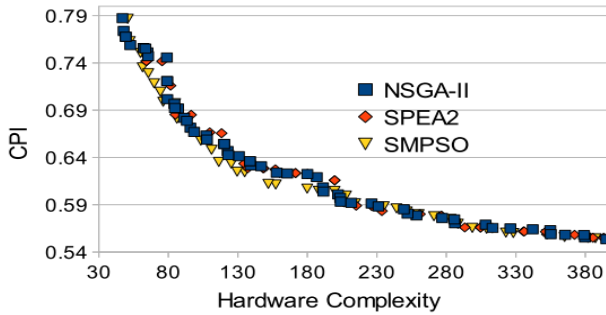


Fig. 6. Section of the Pareto front approximations obtained after 50 generations

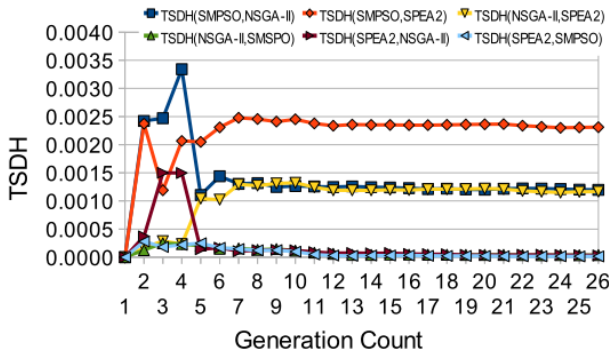


Fig. 7. Comparison between algorithms using the two set difference hypervolume

We decided to use the Two Set Difference Hypervolume (TSDH) metric to see how much of the space is really dominated by a single algorithm (Fig. 7). The highest value is obtained when comparing SMPSO with SPEA2. The next two (in terms of value) comparisons are between SMPSO and NSGA-II and between NSGA-II and SPEA2, so this metric keeps the ranking showed by the other metrics. It also shows us that in fact only 0.001-0.002% of the entire space is covered only by the winning algorithms.

5.2 FADSE with GAPtimize

In this section we present the results obtained after FADSE was run on GAP and GAPtimize with NSGA-II, SPEA2 and SMPSO.

We used the same ten benchmarks as in the previous section, but this time they were compiled without function inlining since GAPtimize is used for function inlining. The cache and the number of configuration layers for GAP were fixed and only the size of the array was varied. For GAPtimize four parameters were changed. The design space with these six parameters has a size of around $1.6 \cdot 10^{13}$.

We have done a hypervolume comparison between NSGA-II, SPEA2 and SMPSO on this problem, too (see Fig. 8). From the hypervolume we conclude that SMPSO still has the best convergence speed and also the best quality of solutions. SPEA2 performs better than NSGA-II. Also SPEA2 continues to perform fewer simulations than NSGA-II.

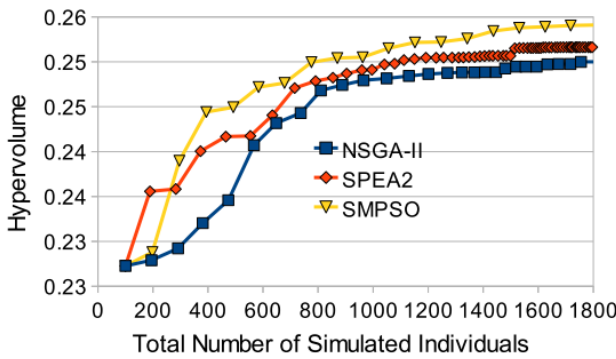


Fig. 8. Hypervolume comparison between NSGA-II, SPEA2, and SMPSO (Hypervolume axis has been trimmed for visibility)

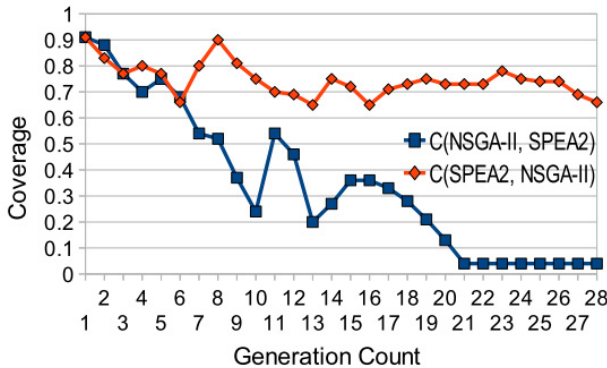


Fig. 9. Coverage comparison between DSE runs with NSGA-II and SPEA2

The coverage computation (see Fig. 9) between the two genetic algorithms also shows that SPEA2 performs much better than NSGA-II when the design space exploration process is performed on both GAP and GAPTimize. We selected SPEA2 as the best algorithm from the previous comparison and put it side by side with SMPSO using the coverage metric (see Fig. 10). SMPSO has the best results, but the difference is not that big, as between SPEA2 and NSGA-II.

NSGA-II was stopped after 30 generations. To reach generation 30 NSGA-II had to simulate around 1800 individuals. We continued the run of SPEA2 until the same number of simulations were performed (10 benchmarks * 1800 individuals). SPEA2 reaches generation 68 before evaluating 1800 individuals. SMPSO simulates the most individuals: it passed 1800 simulations after only 19 generations.

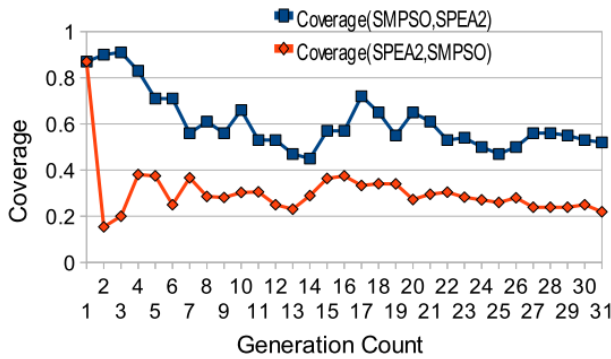


Fig. 10. Coverage comparison between DSE runs with SPEA2 and SMPSO

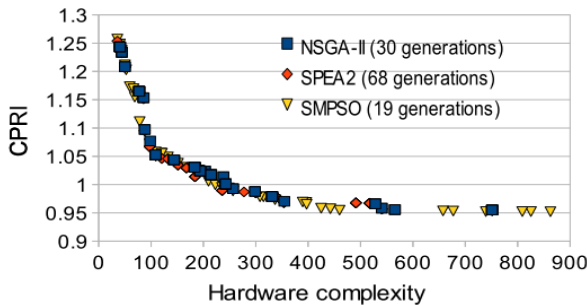


Fig. 11. The final Pareto front approximations obtained by NSGA-II, SPEA2, and SMPSO after 1800 simulations

A comparison of the Pareto front approximations obtained after 1800 evaluations is shown in Fig. 11. SMPSO has a better spread of solutions, while SPEA2 seems to have the worst spread. In terms of quality SMPSO and SPEA2 find better solutions in the area around the hardware complexity of 200.

Finally we compare the three algorithms (two by two) using the two set difference hypervolume metric – TSDH – (see Fig. 12). The results confirmed conclusions. The difference between the results obtained by the algorithms is very small. For the first generations (until generation 7) SMPSO has better results revealed by a higher TSDH value, especially against NSGA-II. SPEA2 also obtains better results compared to NSGA-II for the first generations.

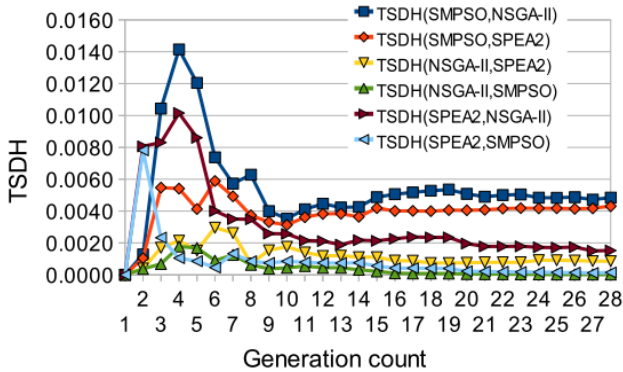


Fig. 12. Comparison between algorithms using the two set difference hypervolume

6 Conclusions and Further Work

We have already proven in [5] that evolutionary algorithms are able to find good results, better than the ones found by a human expert (NSGA-II was used). If we look at the Pareto front approximations obtained by all the evaluated algorithms (see Fig. 6 and Fig. 11) the difference between the best found individuals in terms of performance is not greater than 1% (and usually around 0.1-0.01%) at the same complexity. All the algorithms are able to find good results, so the factor that needs to decide which one proves to be best is convergence speed.

The results obtained show the best convergence for the PSO algorithm, thus confirming the results obtained on synthetic problems by Nebro et al. [11] in their comparison of SMPSO with NSGA-II and SPEA2. Also the spread of the PSO algorithms was better for both explorations (especially compared with SPEA2).

A clear recommendation between NSGA-II and SPEA2 cannot be made. NSGA-II performed better than SPEA2 on GAP but worse on GAP with GAPtimize. The spread, observed after looking at the obtained Pareto front approximations, of the solutions found by SPEA2 is worse than the one of NSGA-II.

Two set difference hypervolume metric proved once more that the difference between all the algorithms is quite small.

We can also conclude that the coverage metric can be misleading by showing a big difference between the algorithms while in fact the results are only slightly better. So even if the quality of solutions seems much better, in reality there is not a great difference.

In a future article metrics based on e-dominance as presented by Coello in [13] could be used, which should avoid these pitfalls. Hypervolume does provide a fair image and it helps discovering which of the algorithms converges faster and when does the algorithm stop discovering better solutions.

In our evaluations, all algorithms were started from the same population (generated randomly). If some known good individuals would have been included not at random but because of domain knowledge it would potentially help the algorithm to find good solutions faster; we introduced this approach in [26].

Fuzzy control logic will be used to input domain knowledge information into the algorithms. The user can describe different relations between parameters from within FADSE [22]. These relations together with the rules can be used to specify domain knowledge easily and they help the design space exploration algorithms perform better.

References

1. Calborean, H., Vintan, L.: An Automatic Design Space Exploration Framework for Multicore Architecture Optimizations. In: Proceedings of the 9th IEEE RoEduNet International Conference, pp. 202–207. IEEE Xplore Digital Library, Sibiu (2010)
2. Calborean, H., Jahr, R., Ungerer, T., Vintan, L.: Optimizing a Superscalar System using Multi-objective Design Space Exploration. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, pp. 339–346 (2011)
3. Calborean, H., Vintan, L.: Toward an efficient automatic design space exploration frame for multicore optimization. ACACES 2010 poster Abstracts, Terassa, Spain, pp. 135–138 (2010)
4. Calborean, H., Vințan, L.: Framework for Automatic Design Space Exploration of Computer Systems. Acta Universitatis Cibiniensis Technical Series (2011)
5. Jahr, R., Ungerer, T., Calborean, H., Vintan, L.: Automatic Multi-Objective Optimization of Parameters for Hardware and Code Optimizations. In: Waleed, W., Smari, J.P.M. (eds.) Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011), pp. 308–316. IEEE (2011)
6. Silvano, C., Fornaciari, W., Palermo, G., Zaccaria, V., Castro, F., Martinez, M., Bocchio, S., Zafalon, R., Avasare, P., Vanmeerbeeck, G., et al.: MULTICUBE: Multi-Objective Design Space Exploration of Multi-Core Architectures. In: Proceedings of the 2010 IEEE Annual Symposium on VLSI, pp. 488–493 (2010)
7. Palermo, G., Silvano, C., Zaccaria, V.: Discrete Particle Swarm Optimization for Multi-objective Design Space Exploration. Digital System Design Architectures, Methods and Tools. In: 11th EUROMICRO Conference on DSD 2008, pp. 641–644 (2008)
8. Lungu, V., Sofron, A.: Using Particle Swarm Optimization to Create Particle Systems. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS-18), pp. 750–754. Politehnica Press, Bucharest (2011)
9. Kang, S., Kumar, R.: Magellan: a search and machine learning-based framework for fast multi-core design space exploration and optimization. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1432–1437. ACM, Munich (2008)

10. Jia, Z.J., Pimentel, A.D., Thompson, M., Bautista, T., Núñez, A.: Nasa: A generic infrastructure for system-level MP-SoC design space exploration. In: 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia), pp. 41–50 (2010)
11. Nebro, A., Durillo, J., Garcia-Nieto, J., Coello, C.A., Luna, F., Alba, E.: Smpso: A new pso-based metaheuristic for multi-objective optimization. In: Proceedings of the IEEE Symposium Series on Computational Intelligence, pp. 66–73 (2009)
12. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm. *Eurogen*, pp. 95–100 (2001)
13. Desmet, V., Girbal, S., Temam, O., France, B.F.: Archexplorer. org: Joint compiler/hardware exploration for fair comparison of architectures. In: INTERACT workshop at HPCA 2009 (2009)
14. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer (2002)
15. Zitzler, E.: *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Swiss Federal Institute of Technology, ETH (1999)
16. Sharkey, J.J., Ponomarev, D., Ghose, K.: M-SIM: A Flexible, Multithreaded Architectural Simulation Environment - Technical Report CS-TR-05-DP01 (2005)
17. Binkert, N.L., Dreslinski, R.G., Hsu, L.R., Lim, K.T., Saidi, A.G., Reinhardt, S.K.: The M5 Simulator: Modeling Networked Systems. *IEEE Micro*, 26, 52–60 (2006)
18. Radu, C., Vințan, L.: Optimized Simulated Annealing for Network-on-Chip Application Mapping. In: Proceedings of the 18th International Conference on Control Systems and Computer Science (CSCS 2018), pp. 452–459. Politehnica Press, Bucharest (2011)
19. Ubal, R., Sahuquillo, J., Petit, S., López, P.: Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors. In: Proc. of the 19th Int’l Symposium on Computer Architecture and High Performance Computing (2007)
20. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Departamento de Lenguajes y Ciencias de la Computación. University of Malaga, E.T.S.I. Informática (2006)
21. Jahr, R., Calborean, H., Vintan, L., Ungerer, T.: Boosting Design Space Explorations with Existing or Automatically Learned Knowledge. In: Schmitt, J. (ed.) *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pp. 221–235. Springer, Heidelberg (2012)
22. Calborean, H.: *Multi-Objective Optimization of Advanced Computer Architectures using Domain-Knowledge*. PhD Thesis, “Lucian Blaga” University of Sibiu, Romania, 2011 (PhD Supervisor: Prof. Lucian Vintan, PhD) (2011)
23. Uhrig, S., Shehan, B., Jahr, R., Ungerer, T.: The Two-dimensional Superscalar GAP Processor Architecture. *International Journal on Advances in Systems and Measurements* 3, 71–81 (2010)
24. Jahr, R., Shehan, B., Uhrig, S., Ungerer, T.: Static Speculation as Post-Link Optimization for the Grid Alu Processor. In: Proceedings of the 4th Workshop on Highly Parallel Processing on a Chip, HPPC 2010 (2010)
25. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 182–197 (2002)
26. Gellert, A., Calborean, H., Vintan, L., Florea, A.: Multi-Objective Optimizations for a Superscalar Architecture with Selective Value Prediction. *IET Computers & Digital Techniques* (accepted, manuscript ID CDT-2011-0116)

Author Index

- Abbaspour, Karim 279
Angelescu, Nicoleta 31
Arghira, Nicoleta 251
Asavei, Victor 379, 453
Aysan, Huseyin 393
- Bacu, Victor 279
Bogdan Moldoveanu, Alin Dragos 379
Bucur, Laurentiu 323
Burlacu, Adrian 115
- Calborean, Horia 489
Caramihai, Simona Iuliana 355
Chera, Catalin 323
Christov, Nicolai 265
Copot, Cosmin 115
Cristea, Valentin 423
Culiță, Janetta 223, 339
- Ditu, Bogdan 407
Dobra, Petru 129, 367
Dobre, Ciprian 423
Dobrescu, Radu 15, 31
Dobrin, Radu 393
Drobot, Radu 295
Dulf, Eva-Henrietta 235
Duma, Radu 367
Dumitrache, Ioan 59, 355
Dumitrașcu, Alexandru 223, 339
Dumitrescu, C.M. 59
Dumitru, Iulia 307
- Egner, Alexandru 379, 453
- Făgărășan, Ioana 251, 307
Forsell, Vili 1
- Giurcăneanu, Ciprian Doru 1
Gorgan, Dorian 279
- Hukkanen, Jenni 77
- Iliescu, Sergiu Stelian 251, 307
- Jahr, Ralf 489
Jelinek, Herbert F. 91, 177
- Kalchev, Boyko 265
Karperien, Audrey 177
Krstonošić, Bojana 177
- Lazar, Corneliu 115
Lazea, Gheorghe 129
Lungu, Valentin 207
Lungu, Vasile 295
- Manisor, Mihaela 191
Mănoiu-Olaru, Sorin 143
Marcu, Cosmin 191
Miclea, Liviu 191
Mihon, Danut 279
Milošević, Nebojša T. 91, 177
Mocanu, Mariana 295
Moldoveanu, Alin 453
Moldoveanu, Florica 379, 453
Morar, Anca 379, 453
Muste, Marian 295
- Neamtu, Andrei-Sorin 157
Negru, Catalin 423
Nițulescu, Mircea 143

- Odubășteanu, Carmen 439
Oltean, Virginia Ecaterina 15
Olujić, Maja 91
Oros, Ana 91
- Petrescu, Serban 323
Ploix, Stéphane 251, 307
Pop, Florin 423
Popescu, Dan 15, 31
Punnekkat, Sasikumar 393
- Radu, Ciprian 473
Radu, Serban 439
Radulescu, Florin 463
Robotin, Radu 129
Rodila, Denisa 279
Rouholahnejad, Elham 279
- Sabo, Edmond 77
Simeonov, Ivan 265
Sita, Ioan Valentin 367
- Ștefănoiu, Dan 223, 339
Stefanut, Teodor 279
Stoica, Adrian-Mihail 157
Szelitzky, Tibor 235
- Tabus, Ioan 77
Tapus, Nicolae 407
Tian, Yang 265
Toader, Adrian 45
Tomoaia, Gheorghe 191
Trusca, Mirela 367
Tudose, Cătălin 439
Turcitu, Cristian 463
- Ungerer, Theo 489
Ursu, Ioan 45
- Vințan, Lucian 473, 489
- Waliszewski, Przemyslaw 103
Wang, Haoping 265