

Reneta P. Barneva
Valentin E. Brimkov
Jake K. Aggarwal (Eds.)

LNCS 7655

Combinatorial Image Analysis

15th International Workshop, IWCIAC 2012
Austin, TX, USA, November 2012
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Reneta P. Barneva Valentin E. Brimkov
Jake K. Aggarwal (Eds.)

Combinatorial Image Analysis

15th International Workshop, IWCIA 2012
Austin, TX, USA, November 28-30, 2012
Proceedings



Springer

Volume Editors

Reneta P. Barneva
State University of New York at Fredonia
Department of Computer and Information Sciences
280 Central Ave.
Fredonia, NY 14063, USA
E-mail: reneta.barneva@fredonia.edu

Valentin E. Brimkov
SUNY Buffalo State College
Mathematics Department
1300 Elmwood Ave.
Buffalo, NY 14222, USA
E-mail: brimkove@buffalostate.edu

Jake K. Aggarwal
University of Texas at Austin
Department of Electrical and Computer Engineering
1 University Station C0803
Austin, TX 78712-0240, USA
E-mail: aggarwaljk@mail.utexas.edu

ISSN 0302-9743
ISBN 978-3-642-34731-3
DOI 10.1007/978-3-642-34732-0
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349
e-ISBN 978-3-642-34732-0

Library of Congress Control Number: 2012951153

CR Subject Classification (1998): I.4.5-6, I.4.9-10, I.3.5, I.3.7, I.5.4, F.2.2, G.2.1, G.1.6, I.4

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, 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.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

This volume contains the articles presented at the 15th International Workshop on Combinatorial Image Analysis, IWCIA 2012, which was held in Austin (TX), November 28–30, 2012. The 14 previous meetings were held in Paris (France) 1991, Ube (Japan) 1992, Washington DC (USA) 1994, Lyon (France) 1995, Hiroshima (Japan) 1997, Madras (India) 1999, Caen (France) 2000, Philadelphia (USA) 2001, Palermo (Italy) 2003, Auckland (New Zealand) 2004, Berlin (Germany) 2006, Buffalo (USA) 2008, Playa del Carmen (Mexico) 2009, and Madrid (Spain) 2011.

Combinatorial image analysis provides theoretical foundations and methods for solving problems from various areas of human practice. In contrast to traditional approaches to image analysis which implement continuous models, float arithmetic and rounding, combinatorial image analysis features discrete models using integer arithmetic. The developed algorithms are based on studying combinatorial properties of classes of digital images, and often appear to be more efficient and accurate than those based on continuous models.

IWCIA is an exciting opportunity for scholars, graduate students, and educators across the world to meet and share information about their latest findings in the field of combinatorial image analysis, be enriched with new ideas, reflect on some open problems, learn about new applications, and reconnect with colleagues. All papers submitted to the conference were carefully reviewed as each manuscript was sent for a double-blind review to at least three highly qualified members of the international Program Committee. The submission and review process of the workshop was carried out through the professional OpenConf conference management system. After a rigorous review process, 23 papers authored by 51 researchers from 11 countries were accepted for presentation at the workshop and for inclusion in this volume.

IWCIA 2012 featured keynote talks delivered by three outstanding scholars, whose excellent presentations inspired the audience with new ideas.

An opening talk given by János Pach (EPFL, Lausanne and Alfréd Rényi Institute of Mathematics, Budapest) was devoted to geometric graph theory. The latter studies geometric (topological) graphs that can be drawn in the plane by straight-line or curvilinear edges satisfying certain conditions. In his talk, the speaker discussed fundamental extremal questions in geometric graph theory and surveyed various results and unsolved problems.

David A. Eppstein (Donald Bren School of Information and Computer Sciences, University of California, Irvine) presented an approach based on three-dimensional hyperbolic geometry to forming a novel type of Voronoi diagram for circles in the plane. The proposed method provides a discrete combinatorial representation for a class of objects which may be applicable to visualization

of broader classes of low-degree planar graphs via “Lombardi drawings” with circular-arc edges.

Gerhard X. Ritter (University of Florida, Gainesville) presented a lattice algebra approach to computational intelligence and image processing. He provided an overview of lattice theory-based models and techniques in the field of computational intelligence and discussed the specific applications to hyperspectral image segmentation and pattern recognition.

The contributed papers included in the volume are grouped into two parts. The first one includes 11 papers devoted to diverse problems of digital geometry and topology, in particular studies on geometry and topology of digital curves and surfaces, the design of space-efficient algorithms, and others. The second part includes papers presenting array grammars and languages for image analysis, research on picture transformations, morphological operations, image segmentation, discrete tomography, and applications.

We believe that all presented works were of high quality and the attendees benefited from the scientific program.

We would like to express our gratitude to everyone who contributed to the success of IWCIA 2012 – from the Steering to the Program and Organizing Committees. We are indebted to our sponsors SUNY Buffalo State College and SUNY Fredonia, and in particular to the Interim Provost Kevin P. Kearns of SUNY Fredonia, who endorsed the publication of this volume.

We wish to express our special thanks to the invited speakers David A. Epstein, János Pach, and Gerhard X. Ritter for their remarkable talks and overall contribution to the workshop program. We thank all authors for their valuable works and hope that the reader will find them interesting and useful. We wish to thank the participants and everyone who made this workshop an enjoyable and fruitful scientific event. We had a great time at the Joe C. Thompson Conference Center of the University of Texas at Austin thanks to Elisabel Bordallo, Conference Services Manager, and Bailey Anne Dermanci; we appreciate their work. Finally, we express our gratitude to Springer’s Computer Science Editorial team, and especially to Alfred Hofmann and Anna Kramer, for their efficient and kind cooperation in the timely production of this book.

November 2012

Reneta P. Barneva
Valentin E. Brimkov
Jake K. Aggarwal

Organization

IWCIA 2012 was held at the University of Texas at Austin, TX, USA, November 28-30, 2012

General Chairs

Jake K. Aggarwal	University of Texas at Austin, USA
Valentin E. Brimkov	SUNY Buffalo State College, USA

Program and Publication Chair

Reneta P. Barneva	SUNY Fredonia, USA
-------------------	--------------------

Steering Committee

Valentin E. Brimkov	SUNY Buffalo State College, USA
Gabor T. Herman	CUNY Graduate Center, USA
Kostadin Koroutchev	Universidad Autonoma de Madrid, Spain
Petra Wiederhold	CINVESTAV-IPN, Mexico

Invited Speakers

David A. Eppstein	University of California, Irvine, USA
János Pach	EPFL, Lausanne, Switzerland and Rényi Institute, Budapest, Hungary
Gerhard X. Ritter	University of Florida, USA

Program Committee

Til Aach	RWTH Aachen University, Germany
Lyuba Alboul	Sheffield Hallam University, UK
Eric Andres	University of Poitiers, France
Arrate Muñoz	Universidad de Navarra, Spain
Akira Asano	Hiroshima University, Japan
Jacky Baltes	University of Manitoba, Canada
Péter Balázs	University of Szeged, Hungary
George Bebis	University of Nevada at Reno, USA
Bedrich Benes	Purdue University, USA
Gilles Bertrand	ESIEE, France
Bhargab B. Bhattacharya	Indian Statistical Institute, India
Peter Brass	City College, City University of New York, USA

VIII Organization

Alfred M. Bruckstein	Technion, I.I.T, Israel
Jean-Marc Chassery	University of Grenoble, France
Li Chen	University of the District of Columbia, USA
Marco Cristani	University of Verona, Italy
Guillaume Damiand	LIRIS-CNRS, Université de Lyon, France
Leila De Floriani	University of Genova, Italy and University of Maryland, USA
Isabelle Debled-Rennesson	Nancy University, LORIA, France
Eduardo Destefanis	Universidad Tecnologica Nacional Córdoba, Argentina
Chiou-Shann Fuh	National Taiwan University, Taiwan
Jürgen Gall	ETH Zürich, Switzerland
Edgar Garduño	IIMAS-UNAM, Mexico
Rocío González Díaz	University of Seville, Spain
Jordi González i Sabaté	UAB, Spain
Concettina Guerra	Università di Padova, Italy
Edwin Hancock	University of York, UK
Atsushi Imiya	IMIT, Chiba University, Japan
Damien Jamet	University of Nancy, France
María José Jiménez	University of Seville, Spain
Ramakrishna Kakarala	NTU, Singapore
Andreas Koschan	University of Tennessee, USA
Walter G. Kropatsch	Vienna University of Technology, Austria
Longin Jan Latecki	Temple University, USA
Jerome Liang	SUNY Stony Brook, USA
Pascal Lienhardt	University of Poitiers, France
Shih-Schon Lin	University of Pennsylvania, USA
Joakim Lindblad	Swedish University of Agricultural Sciences, Sweden
Hongbing Lu	Fourth Military Medical University, China
Avner Magen	University of Toronto, Canada
Rémy Malgouyres	Université d'Auvergne, France
Ramon Mas Sansó	Universitat de les Illes Balears, Spain
Erik Melin	Uppsala University, Sweden
Christian Mercat	Université Montpellier, France
Vittorio Murino	University of Verona, Italy
Benedek Nagy	University of Debrecen, Hungary
Akira Nakamura	Hiroshima University, Japan
Renato M. Natal Jorge	University of Porto, Portugal
Gregory M. Nielson	Arizona State University, USA
János Pach	City College and Courant Institute, USA
Kalman Palagyi	University of Szeged, Hungary
Petra Perner	Institute of Computer Vision and Applied Computer Sciences, Germany
Hemerson Pistori	Dom Bosco Catholic University, Brazil

Ioannis Pitas	University of Thessaloniki, Greece
Konrad Polthier	Freie Universität Berlin, Germany
Hong Qin	SUNY Stony Brook, USA
Paolo Remagnino	Kingston University, UK
Ralf Reulke	Humboldt University, Germany
Gerhard Ritter	University of Florida, USA
Mariano Rivera	CIMAT, Mexico
Xavier Roca Marvà	UAB, Spain
Bodo Rosenhahn	MPI Informatik, Germany
Arun Ross	West Virginia University, USA
Angel Sappa	Computer Vision Center, Spain
Henrik Schulz	Forschungszentrum Dresden, Germany
Nikolay Sirakov	Texas A&M University, USA
Rani Siromoney	Madras Christian College, India
Isabelle Sivignon	LIRIS-CNRS, University of Lyon, France
Wladyslaw Skarbek	Warsaw University of Technology, Poland
Ali Shokoufandeh	Drexel University, USA
Alberto Soria	CINVESTAV, Mexico
K.G. Subramanian	Universiti Sains Malaysia, Malaysia
Akihiro Sugimoto	National Institute of Informatics, Japan
Mohamed Tajine	Louis Pasteur University, Strasbourg, France
Joao Manuel R.S. Tavares	University of Porto, Portugal
Antonio Turiel	ICM, CSIC, Spain
Peter Veelaert	Ghent University, Belgium
Young Woon Woo	Dong-Eui University Busan, Korea
Jinhui Xu	SUNY University at Buffalo, USA
Yasushi Yagi	Osaka University, Japan
Jason You	Cubic Imaging LLC, USA
Richard Zanibbi	Rochester Institute of Technology, USA
Larry Zeng	University of Utah, USA

Organizing Committee

Reneta P. Barneva	SUNY Fredonia, USA
Michael Szocki	SUNY Fredonia, USA
Boris Brimkov	SUNY Fredonia, USA

Sponsoring Institutions

SUNY Buffalo State College, Buffalo, NY

SUNY Fredonia, Fredonia, NY

The Interim Provost of SUNY Fredonia, Dr. Kevin P. Kearns, sponsored the conference proceedings

Table of Contents

Part I: Geometry and Topology

Digital Geometry, Combinatorics in Digital Spaces. Digital Curves and Surfaces

On Finding Shortest Isothetic Path Inside a Digital Object	1
<i>Mousumi Dutt, Arindam Biswas, Partha Bhowmick, and Bhargab B. Bhattacharya</i>	
Fast Slicing of Orthogonal Covers Using DCEL	16
<i>Nilanjana Karmakar, Arindam Biswas, and Partha Bhowmick</i>	
Fast Combinatorial Algorithm for Tightly Separating Hyperplanes	31
<i>Peter Veelaert</i>	
Digital Curvatures Applied to 3D Object Analysis and Recognition: A Case Study	45
<i>Li Chen and Soma Biswas</i>	
Discrete Polynomial Curve Fitting to Noisy Data	59
<i>Fumiki Sekiya and Akihiro Sugimoto</i>	
A Probabilistic Measure of Circularity	75
<i>Ana Marcela Herrera-Navarro, Hugo Jiménez-Hernández, and Iván Ramón Terol-Villalobos</i>	

Digital Topology

A New Framework for Connected Components Labeling of Binary Images	90
<i>Tetsuo Asano and Sergey Bereg</i>	
Small Work Space Algorithms for Some Basic Problems on Binary Images	103
<i>Tetsuo Asano, Sergey Bereg, and Lilian Buzer</i>	
Adjacencies for Structuring the Digital Plane	115
<i>Josef Šlapal</i>	
On Topology Preservation for Triangular Thinning Algorithms	128
<i>Péter Kardos and Kálmán Palágyi</i>	
Cellular Topology on the Triangular Grid	143
<i>Benedek Nagy</i>	

Part II: Grammars, Transformations, Applications

Grammars and Models in Image Analysis

A P System Model for Contextual Array Languages	154
<i>K.G. Subramanian, Ibrahim Venkat, and Petra Wiederhold</i>	
Rectangular Arrays and Petri Nets	166
<i>D. Lalitha, K. Rangarajan, and Durairaj Gnanaraj Thomas</i>	
Regional Hexagonal Tile Rewriting Grammars	181
<i>Thangasamy Kamaraj and Durairaj Gnanaraj Thomas</i>	
Partial Commutation on Array Languages	196
<i>Thangasamy Kamaraj, Durairaj Gnanaraj Thomas, H. Geetha, and T. Kalyani</i>	

Picture Transformations, Morphologic Operations, Image Segmentation

Incremental Learning of the Model for Watershed-Based Image Segmentation	209
<i>Anja Attig and Petra Perner</i>	
Fast Level-Wise Convolution	223
<i>Damien Gonzalez, Rémy Malgouyres, Henri-Alex Esbelin, and Chafik Samir</i>	
Combinatorial Properties of 2D Discrete Rigid Transformations under Pixel-Invariance Constraints	234
<i>Phuc Ngo, Yukiko Kenmochi, Nicolas Passat, and Hugues Talbot</i>	
Novel Morphological Algorithms for Dominating Sets on Graphs with Applications to Image Analysis	249
<i>Anupama Potluri and Chakravarthy Bhagvati</i>	

Discrete Tomography, Applications

Binary Image Reconstruction from Two Projections and Skeletal Information	263
<i>Norbert Hantos, Péter Balázs, and Kálmán Palágyi</i>	
Energy-Minimization Based Discrete Tomography Reconstruction Method for Images on Triangular Grid	274
<i>Tibor Lukić and Benedek Nagy</i>	

Skin Lesion Feature Vector Space with a Metric to Model Geometric Structures of Malignancy for Classification	285
<i>Mutlu Mete, Ye-Lin Ou, and Nikolay Metodiev Sirakov</i>	
Segmentation by a Local and Global Fuzzy Gaussian Distribution Energy Minimization of an Active Contour Model	298
<i>Quang Tung Thieu, Marie Luong, Jean-Marie Rocchisani, Nikolay Metodiev Sirakov, and Emmanuel Viennet</i>	
Author Index	313

On Finding Shortest Isothetic Path inside a Digital Object

Mousumi Dutt^{1,*}, Arindam Biswas¹,
Partha Bhowmick², and Bhargab B. Bhattacharya³

¹ Department of Information Technology,
Bengal Engineering and Science University, Shibpur, Howrah, India
{duttmousumi,barindam}@gmail.com

² Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur, India
bhowmick@gmail.com

³ Advanced Computing and Microelectronics Unit,
Indian Statistical Institute, Kolkata, India
bhargab@isical.ac.in

Abstract. Shortest path algorithms are finding interesting applications in recent times in various emerging areas of image analysis and computer vision. Such algorithms are designed to solve shortest path problems with variegated need-based constraints. We present here an efficient combinatorial algorithm to find a/the shortest isothetic path (SIP) between two grid points in a digital object such that the SIP lies entirely inside the object. The algorithm first obtains the inner isothetic cover (simple and without holes) of the object and then applies certain combinatorial rules to construct the SIP and its constituent monotone sub-paths. For a given grid size, the entire algorithm runs in $O(n \log n)$ time, n being the number of grid points on the border of the cover. Experimental results show the effectiveness of the algorithm and its further prospects in shape analysis.

Keywords: Shortest path, Shortest isothetic path, Manhattan path, Monotone path, Shape analysis.

1 Introduction

Shortest paths—constrained and customized according to need and application—are increasingly used today in various fields. Some of the worth-mentioning ones are networking, robotics, GIS, VLSI design, resource allocation and collection, TSP with neighborhoods, etc. [2,9,12,15,22,26,27]. Due to such a vast range of applications, the variation in constraint formulation and customization has also become quite engrossing, as evidenced in the related literature [5,3,4,10,14,18,19,21,24].

* Corresponding author.

Over the last few years, shortest-path algorithms have also been used in several interesting applications in the domain of image analysis and computer vision. Some typical ones are borehole image analysis, image patching, object boundary detection, road detection in satellite images, shape classification, panoramic stereo matching, fracture identification, analysis of orientation, stratigraphic and structural dip, etc. [8,25]. The work in [25] proposes the extraction of a *circular shortest path* in an image. Quite recently, shortest path has been used in [20] to measure the *inner-distance* between two points (lying possibly in two different ‘parts’ of a given object), which is subsequently used for shape analysis and shape classification. In the context of *active contours*, *regularized shortest path* has been used in [8], with a scheme of *pixel subdivision* to resolve discretization effects. The algorithm is modified from a simple and intuitive smoothness constraint, proposed earlier in [1].

Our work is focused on finding the *shortest isothetic path* (SIP) inside a digital object A laid on a grid \mathbb{G} (Sec. 2), such that the path lies entirely inside A and consists of moves along grid edges only. Although our problem has some similarity with certain problems of computational geometry, no efficient algorithmic solution is available for the digital-geometric problem stated here, till date. For a given grid size, the proposed algorithm runs in $O(n \log n)$ time, n being the number of grid points on the border of the inner isothetic cover, P (maximum-area isothetic polygon inscribing A [6,7]). Note that, if the shortest path is obtained directly from A (without using P), then the time complexity of an optimal algorithm would be no less than $O(n^2)$, since the number of constituent pixels of A can be $O(n^2)$.

One of the related geometric problems is finding a/the 2D rectilinear shortest path from one point to another in a polygonal region. The best-known algorithm has been proposed most recently in [16], which takes $O(n + m(\log n)^{3/2})$ time, m being the number of polygonal obstacles with n vertices in total; and for $m = O(n)$, it takes $O(n(\log n)^{3/2})$ time. A similar but simpler problem is finding a shortest path amidst axis-parallel rectangular barriers, and its algorithm executes in $O(t + \log n)$ time, with preprocessing in $O(n \log n)$ time [23], n being the number of disjoint rectangles and t the number of turns on the path. For a given monotone direction, the monotone path between two points in a polygon is solvable in $O(n \log n + rR)$ time, where n is the number of polygon vertices, r the number of scan reflex vertices, and R is the total number of reflex vertices of the polygon [28]; clearly, for $rR = O(n^2)$, it is inefficient for its quadratic time complexity. (Given some inclination, a reflex vertex v of a polygon is called a ‘scan reflex vertex’ if both the vertices adjacent to v lie on one side of the line parallel to the inclination and passing through v [13].) A review of algorithms on similar problems may be seen in [16].

A shortest path or a monotone path is usually not unique, and our algorithm reports one of the shortest paths. The inner isothetic cover, P , of the digital object, A , is first constructed, from which SIP is derived between two given points, p and q , by applying certain combinatorial rules (Sec. 4). As SIP lies inside P and P is (maximally) contained in A (Sec. 2), we get the solution by

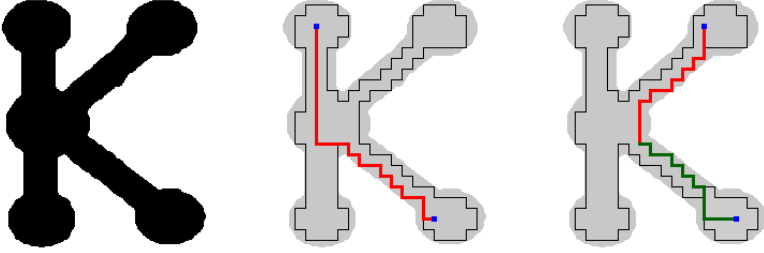


Fig. 1. Left: Digital object A . **Middle:** SIP with one (isothetic) monotone sub-path. **Right:** SIP with two monotone sub-paths (red and green). See electronic version for the original color.

processing P in an efficient way (Sec. 5). Figure 1 shows a digital object A and the solution of SIP for two pairs of points lying inside A . Further results with necessary explanation are given in Sec. 6.

2 Definitions and Preliminaries

A digital object A is a 8-connected component [17]. The *background grid* is given by $\mathbb{G} = (\mathbb{H}, \mathbb{V})$, where \mathbb{H} and \mathbb{V} represent the respective sets of (equi-spaced) horizontal grid lines and vertical grid lines. The *grid size* g is defined as the distance between two consecutive horizontal/vertical grid lines. A *grid point* is the point of intersection of a horizontal and a vertical grid line. A *unit grid block* (UGB) is the smallest square having its four vertices as four grid points and edges as grid edges. The *inner isothetic cover* of A is a polygon P whose vertices are grid points and edges lie on grid lines. An (simple) *isothetic path* from a grid point $p \in P$ to a grid point $q \in P$ is a sequence of 4-connected grid points such that all the constituent points of π are distinct and lies on or inside P ; and a *shortest isothetic path* (SIP) is an/the isothetic path of minimum length. A SIP is said to be *monotone* if it consists of moves only in one or two (orthogonal) directions. If a SIP is not monotone, then it is decomposed into (minimum) monotone sub-paths by our algorithm (Fig. 1).

The 90° and 270° vertices of P are referred to as type **1** and type **3** vertices. The sequence of vertices of P is such that P always lies left of each edge during its traversal. In this vertex sequence, a pair of consecutive vertices of type **1** gives rise to a *convexity*, whereas a pair of consecutive vertices of type **3** gives rise to a *concavity*. The straight line passing through two consecutive vertices of type **1** (type **3**) is called the *convexity line* (*concavity line*). A *convex region* corresponding to a horizontal (vertical) convexity line is the maximum-area rectangle such that the horizontal (vertical) line segment whose endpoints lie on two vertical (horizontal) sides of the rectangle, is either the convexity line or lies strictly inside P . For example, in Fig. 2(left), the convex region is closed on three sides

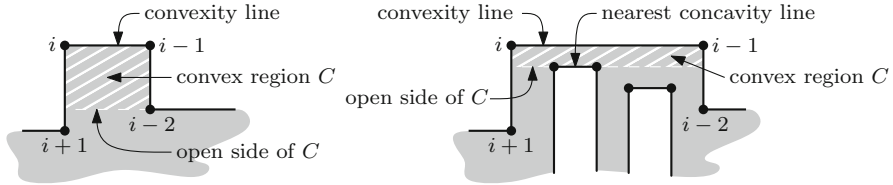


Fig. 2. Instances of convex region (hatched in white) defined by convexity line $v_i v_{i-1}$ on one side and nearest vertex v_{i-2} or v_{i+1} (left) or nearest concavity line (right) on the opposite side. A vertex v_i is denoted by i for simplicity.

by the edges (v_{i-2}, v_{i-1}) , (v_{i-1}, v_i) , and (v_i, v_{i+1}) , and open on the side defined by the horizontal line passing through v_{i-2} . In Fig. 2(right), the convex region is closed similarly on three sides by the edges as in Fig. 2(left), and it is open on the side defined by the nearest concavity line. The side of the convex region opposite to the convexity line is referred to as its *open side*. A SIP has certain interesting properties in terms of the lines of convexity and concavity, which are explained in Sec. 3.

Using the algorithm in [6,7], we obtain (the ordered set of vertices of) P for A , which is, therefore, the maximum-area isothetic polygon inscribing A . During the construction of P , the vertices of P are dynamically inserted in a circular doubly-linked list, L ; the vertices and the grid points lying on the edges of P are simultaneously stored in two lexicographically sorted (in increasing order) lists, L_x and L_y , with respective primary and secondary keys as x - and y -coordinates for L_x , and opposite for L_y . Each vertex in L is assigned an *index* in order of its occurrence during traversal of P .

3 Properties of a SIP

We explain here some important properties of a SIP, which are subsequently used in our algorithm. The notation $|\pi(p, q)|$ is used to denote the length of isothetic path, $\pi(p, q)$, between two grid points $p(x_p, y_p)$ and $q(x_q, y_q)$ lying inside or on the border of P . The path $\pi(p, q)$ is monotone if and only if it comprises either one move only (when $x_p = x_q$ or $y_p = y_q$) or two orthogonal moves taken alternately (when $x_p \neq x_q$ and $y_p \neq y_q$). Let $|\pi(p, q)|_x$ and $|\pi(p, q)|_y$ be the sum of lengths of all moves parallel to x -axis and that of all moves parallel to y -axis, respectively. Then it is easy to see that $|\pi(p, q)|_x = \text{abs}(x_p - x_q)$ and $|\pi(p, q)|_y = \text{abs}(y_p - y_q)$, where $\text{abs}(a - b) = \max(a, b) - \min(a, b)$; hence we have the following observation.

Observation 1. *An isothetic path $\pi(p, q)$ between $p(x_p, y_p)$ and $q(x_q, y_q)$ is monotone if and only if $|\pi(p, q)| = |\pi(p, q)|_x + |\pi(p, q)|_y$, where $|\pi(p, q)|_x = \text{abs}(x_p - x_q)$ and $|\pi(p, q)|_y = \text{abs}(y_p - y_q)$.*

A monotone path has, therefore, always the smallest length over all possible isothetic paths. In particular, we have the following lemma.

Lemma 1. *Let $\pi_1(p, q)$ be a monotone path and $\pi_2(p, q)$ be an isothetic path between p and q . Then $|\pi_1(p, q)| < |\pi_2(p, q)|$ if and only if $\pi_2(p, q)$ is non-monotone.*

Proof. W.l.o.g., let $x_p < x_q$ and $y_p > y_q$. If π_2 is a non-monotone path, then it contains more than two types of moves including rightward and downward moves. The monotone path π_1 , on the contrary, consists only of rightward and downward moves. Hence, from Observation [1](#), $|\pi_1| < |\pi_2|$.

Conversely, let $|\pi_1| < |\pi_2|$, which implies $|\pi_2| > |\pi(p, q)|_x + |\pi(p, q)|_y$, i.e., π_2 is not monotone (from Observation [1](#)). \square

Lemma 2. *If $\pi(p, q)$ is a SIP between p and q , and u and v are two grid points lying on $\pi(p, q)$, then the portion of $\pi(p, q)$ between u and v is also a SIP between u and v .*

Proof. Let $\pi(u, v)$ be a SIP between u and v , and $\pi_{pq}(u, v)$ be the portion of $\pi(p, q)$ between u and v . If $|\pi(u, v)| < |\pi_{pq}(u, v)|$, then $\pi_{pq}(u, v)$ could be replaced by $\pi(u, v)$ in $\pi(p, q)$, thereby yielding a shorter SIP between p and q —a contradiction. \square

Theorem 1 (convex region). *A SIP between two grid points p and q does not pass through any convex region other than only the one(s) in which p and q might lie.*

Proof. Let $\pi(p, q)$ be a SIP between p and q . If p or q lies in a convex region, then $\pi(p, q)$ has to enter that convex region. Otherwise, if $\pi(p, q)$ passes through a convex region C in which neither p nor q lies, then $\pi(p, q)$ must enter and leave C through two distinct grid points, namely u and v , both lying on the ‘open side’ of C . The portion of $\pi(p, q)$ between u and v , namely $\pi_{pq}(u, v)$, is longer than the SIP given by the line segment uv lying on the open side of C , which contradicts Lemma [2](#). Hence the proof. \square

Theorem [1](#) provides an important characterization of SIP. This is used in our work to frame a set of rules for contracting an isothetic path to get the final solution (Sec. [4](#)). Another important characterization of a SIP is in terms of the concavity lines that it passes through. The following theorem explicates the idea.

Theorem 2 (concavity line). *A non-monotone SIP always passes through some concavity line(s), whence its monotonicity is broken. However, passing through a concavity line is not the sufficient condition for a SIP to be non-monotone.*

Proof. Observe that a SIP is non-monotone if and only if it takes a ‘U-turn’—defined as three consecutive moves in which the first and the third are in opposite directions. If the points surrounded by such a U-turn lie inside P , then it would result in a convex region through which passes the SIP—a contradiction of Theorem [1](#). Hence, the points surrounded by such a U-turn lie outside P , which implies that the second move of the U-turn is definitely along a concavity line.

The converse is not always true. It is evident from a typical example when a SIP is monotone and just touches one or more concavity lines without taking any U-turn (e.g., Fig. [3](#)(right)). \square

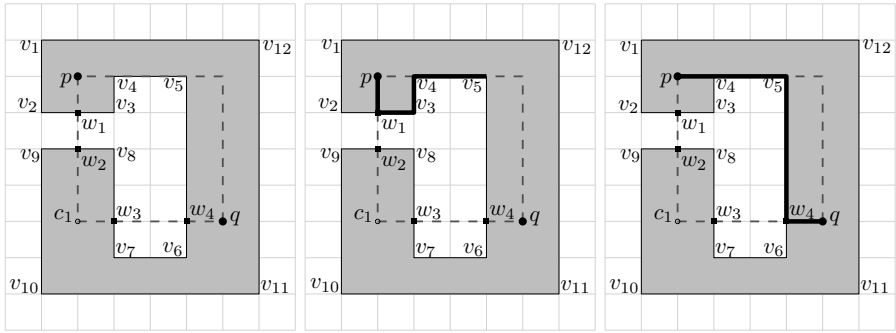


Fig. 3. **Left:** An isothetic polygon P and the bounding rectangle R having p and q as two opposite corners. **Middle:** Intermediate steps. **Right:** Resulting SIP.

The usefulness of Theorem 2 is that a SIP loses its monotonicity due to U-turns. Each U-turn is characterized by a pair of consecutive vertices of type 3. During traversal of P , the types of its vertices are used to determine such U-turns after applying necessary path contraction rules, as explained in Sec. 4.

4 Finding a SIP

To find a SIP between p and q , first a (isothetic) *bounding rectangle*, R , is considered with p and q as opposite corners (Fig. 3). Let, w.l.o.g., p be the top-left corner of R . Let the bottom-left corner be denoted as c_1 . Then the *left semi-border* of R , formed by pc_1 and c_1q , may intersect P at different grid points. Let $M = \langle w_1, w_2, \dots, w_k \rangle$ be the sequence of intersection points between the left semi-border of R and P . If w_i lies on the edge v_jv_{j+1} of P , then we assign $\text{index}[w_i] = j + 0.5$ as the index of w_i ($i = 1, 2, \dots, k$), since $\text{index}[v_j] = j$ for each vertex $v_j \in P$. The ordering of these intersection points has certain interesting properties related with SIP, as stated in the following theorem.

Theorem 3 (ordering). *If the (open) line segment w_iw_{i+1} lies outside P , w_i lies on a SIP $\pi(p, q)$, and $\pi(w_i, w_{i+1})$ is a SIP between w_i and w_{i+1} , then for each vertex $u \in P$ lying on $\pi(w_i, w_{i+1})$, one of the following four conditions is true.*

1. if $\text{index}[w_1] \leq \text{index}[w_i] < \text{index}[w_{i+1}] \leq \text{index}[w_k]$, then $\text{index}[w_i] \leq \text{index}[u] \leq \text{index}[w_{i+1}]$.
2. if $\text{index}[w_1] < \text{index}[w_k] \leq \text{index}[w_{i+1}] < \text{index}[w_i]$, then $\text{index}[w_{i+1}] \leq \text{index}[u] \leq \text{index}[w_i]$.
3. if $\text{index}[w_k] \leq \text{index}[w_{i+1}] < \text{index}[w_i] \leq \text{index}[w_1]$, then $\text{index}[w_{i+1}] \leq \text{index}[u] \leq \text{index}[w_i]$.
4. if $\text{index}[w_i] < \text{index}[w_{i+1}] \leq \text{index}[w_k] < \text{index}[w_1]$, then $\text{index}[w_i] \leq \text{index}[u] \leq \text{index}[w_{i+1}]$.

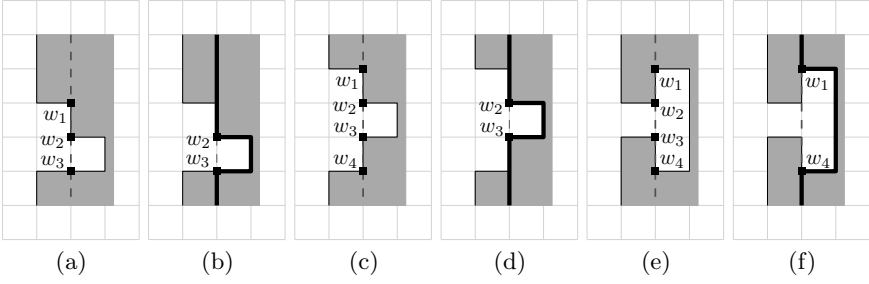


Fig. 4. Illustration of rules for selecting intersection points. The vertical dashed line is the left border of R .

Proof. We first prove Condition 1. Observe that, since w_i is a point on the semi-border of R , we have two vertices of P that are adjacent to w_i . Let these two vertices be u' and u'' . Clearly, $index[w_i]$ would lie between $index[u']$ and $index[u'']$. W.l.o.g., let $index[u'] < index[w_i] < index[u'']$. If $\pi(w_i, w_{i+1})$ contains u' after w_i , then all other vertices following u' in $\pi(w_i, w_{i+1})$ will be in decreasing order of their index values and w_{i+1} cannot be reached unless the points in $\pi_{pq}(p, w_i)$ are visited. This is not possible, as each point in $\pi(p, q)$ occurs exactly once. Hence, $\pi(w_i, w_{i+1})$ should contain u'' after w_i . As the vertices of P next to u'' have increasing index values, the proof follows.

In Condition 2, as w_{i+1} has a smaller index, the indices of vertices along $\pi(w_i, w_{i+1})$ should be in decreasing order so that w_{i+1} can be reached without meeting any vertex already included in $\pi_{pq}(p, w_i)$. Conditions 3 and 4 are simply the reverse of Conditions 1 and 2, respectively. Hence, their proofs are similar to those of Conditions 1 and 2.

To complete the proof of the theorem as a whole, observe that the four indices, i.e., $index[w_1], index[w_i], index[w_{i+1}], index[w_k]$, can be arranged on the number line in $4! = 16$ ways. Out of these, 12 cases are not possible, since the interior of P always lies left during traversal. For example, $index[w_1] < index[w_k] \leq index[w_i] < index[w_{i+1}]$ is one of such impossible cases; for, w_{i+1} always comes after w_k and w_i always after w_{i+1} while traversing P , and hence w_i cannot have an index lower than w_{i+1} . \square

4.1 Intersection Points

The intersection points are detected using L_x and L_y , and stored in M_x and M_y respectively (Sec. 2). Some of them are discarded based on combinatorial rules explained next. During the construction of SIP, reduction rules (Sec. 4.2) are used to shorten the length of the path wherever applicable. Also note that on considering the top-right corner of R , we get intersection points on the top and right borders of R , which may be used to find SIP. The paths, although different, will ensure the same length by our algorithm.

```

Procedure REMOVE-POINTS( $M_y, M_x, p, q$ )
1. CONSTRUCTPAIR( $M_y, M_x$ )
2. if ( $|M_y| = \text{EVEN}$ )
3.    $M \leftarrow \text{CONCAT}(p, M_y, c_1, M_x, q), k' \leftarrow 1$ 
4. else
5.    $M \leftarrow \text{CONCAT}(p, M_y, M_x, q), k' \leftarrow 0$ 
6. Initialize  $i \leftarrow 1$ 
7. while ( $M[i] \neq q$ )
8.   if ( $M[i] = c_1$ )
9.      $i \leftarrow i + 1$ 
10.  continue
11. else
12.  while not ( $(\text{index}[M[i]] < \text{index}[M[i+1]] \leq \text{index}[M[k+k']]) \vee (\text{index}[M[i]] >$ 
     $\text{index}[M[i+1]] \geq \text{index}[M[k+k']])$ )
13.    if ( $M[i+2] = c_1$ )
14.      REMOVE( $M[i+1], M[i+2], M[i+3]$ )
15.    else
16.      REMOVE( $M[i+1], M[i+2]$ )
17.    if ( $\text{type}[M[i]] = 3 \wedge \text{type}[M[i+1]] = 3$ )
18.      REMOVE( $M[i], M[i+1]$ )
19.    continue
20.   $i \leftarrow i + 2$ 

```

Fig. 5. Procedure REMOVE-POINTS

In the sequence of points of intersection between R and P , if there are two consecutive vertices of types **13** (**31**), then the vertex with greater (lower) index is discarded so that the line segment joining two intersection points lies entirely outside P . In Fig. 4(a,b), $\text{type}(w_1, w_2) = \mathbf{31}$, so w_1 is discarded (since w_1w_2 lies on P but w_2w_3 lies outside P). In Fig. 4(c,d), $\text{type}(w_1, w_2) = \mathbf{31}$, so w_1 is discarded, and $\text{type}(w_3, w_4) = \mathbf{13}$, so w_4 is discarded. Similar explanation holds for the case in Fig. 4(e,f). This operation on vertices are performed in CONSTRUCTPAIR (Step 1 in REMOVE-POINTS).

Figure 5 shows the procedure REMOVE-POINTS that removes points from M_y and M_x , which initially contain all the intersection points along the left semi-border of R . The list of selected intersection points, M , is formed by concatenating p , M_y , c_1 (if it is inside P), M_x , and q . If the number of intersection points in M_y is even, then c_1 is inside P (Step 2). In Steps 12–19, the indices of pairs of intersection points are checked to find whether they are in increasing or in decreasing order and whether the index falls within the indices of extreme two points in M , namely $M[1]$ and $M[k+k']$, where k is the total number of intersection points. The value of k' indicates whether c_1 has been included in M ($k' = 1$ (Step 3)) or not ($k' = 0$ (Step 5)).

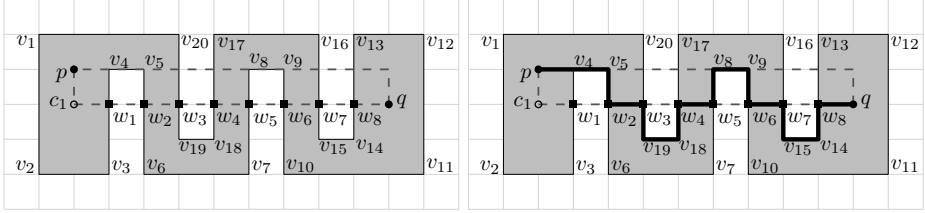


Fig. 6. **Left:** Polygon P , with rectangle R defined by p and q . **Right:** Resulting SIP (thick line).

If the test in Step 12 succeeds, then three consecutive intersection points in M are removed when c_1 is the second next point from $M[i]$ (Step 14); else next two consecutive points are removed (Step 16). If $M[i]$ and $M[i+1]$ are of Type 3, then both are discarded, and the while loop at Step 7 continues (Steps 17–19). The index i is increased by 2 (Step 20) in order to check the next pair of points. The **while** loop (Steps 7–20) continues till the last point in M , i.e., until q is reached. See Fig. 3: REMOVE-POINTS outputs $M = \langle p, w_1, w_4, q \rangle$. For the polygon in Fig. 6 the indices of (w_1, w_2) , (w_3, w_4) , (w_5, w_6) , and (w_7, w_8) are $(3.5, 5.5)$, $(19.5, 17.5)$, $(7.5, 9.5)$, and $(15.5, 13.5)$, respectively. Thus, $index[w_1] = 3.5$ and $index[w_8] = 13.5$ (increasing order). For (w_1, w_2) , Condition 1 of Theorem 3 is true, and so the SIP from w_1 to w_2 passes through v_4 and v_5 (increasing order). For (w_3, w_4) , Condition 2 of Theorem 3 is true, and so the SIP from w_3 to w_4 is through v_{19} and v_{18} (decreasing order). Other pairs may be explained in a similar manner.

4.2 Reduction Rules

To incorporate Theorem 1, reduction rules are applied during the traversal along (border of) P when two consecutive *convex vertices* (type 11) appear, so that the length of the path is minimized. Let v_0, v_1, v_2, v_3 , and v_4 be five consecutive vertices for which the rules have to be applied to reduce the path-length, and the most recently traversed point is v_4 . Let l_i be the distance from v_i to v_{i+1} . Two sets of rules are designed: Rule R_1 (R_{11}, R_{12}, R_{13}) for simple convex regions (type 11) and Rule R_2 when there are one or more *concavities* (a pair of type 33) intruding inside the convex region. The rules are applied on the vertex set $\{v_0, v_1, \dots, v_4\}$ if $type(v_2v_3) = 11$.

Rule R11: Applied when $l_1 = l_3$ (Fig. 7a). Vertices v_1, v_2, v_3, v_4 are removed, l_0 modified as $l_0 + l_2 + l_4$, path-length reduced by $(l_1 + l_3)$.

Rule R12: Applied when $l_1 < l_3$ (Fig. 7b). Vertices v_1, v_2 are removed, v'_3 is introduced instead of v_3 , l_0 updated to $l_0 + l_2$, l'_3 set to $l_3 - l_1$, path-length reduced by $2l_1$.

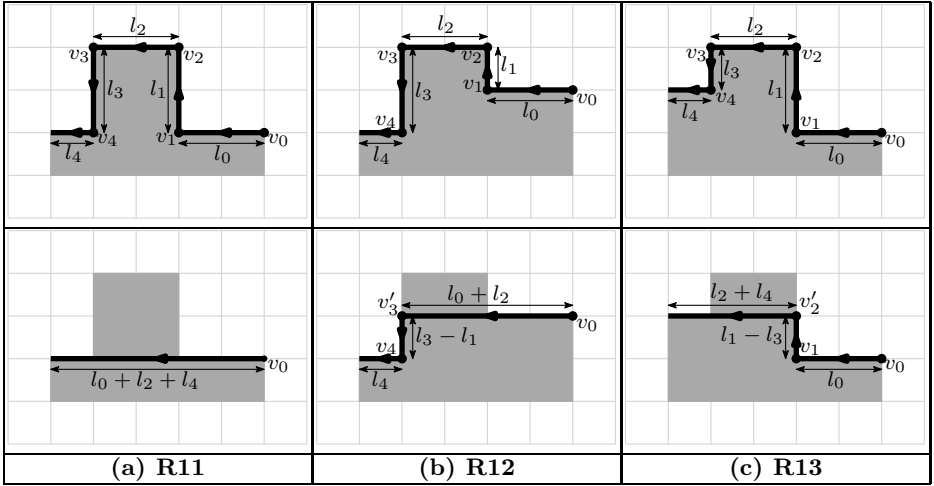


Fig. 7. Rules of reduction for two consecutive convex vertices (type 11)

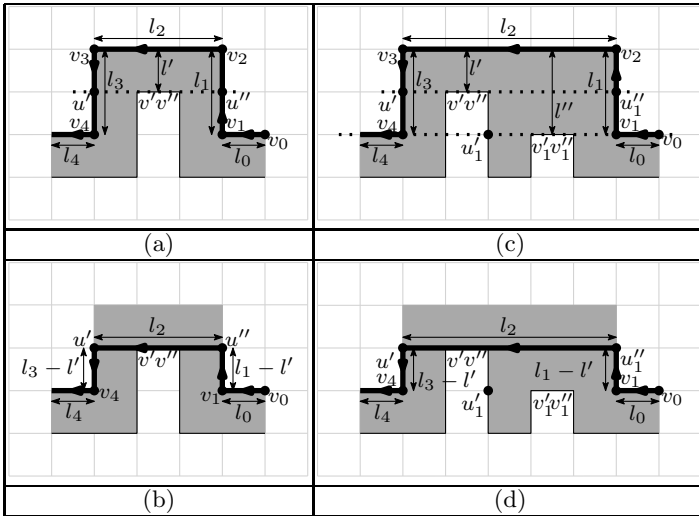


Fig. 8. Reduction rule **R2**

Rule R13: Applied when $l_1 > l_3$ (Fig. 7c). Vertices v_3, v_4 are removed, v_2 is modified as v'_2 , l_1 updated to $l_1 - l_3$, l'_2 set to $l_2 + l_4$, path-length reduced by $2l_3$.

Rule R2: Applied when one ($v'v''$ in Fig. 8a) or more ($v'v''$ and $v'_1v''_1$ in Fig. 8c) concavities are inside the convex region described by v_2v_3 . Let $l' (> \min\{l_1, l_3\})$ denote the minimum of the distances between the line v_2v_3 and the line(s) of concavities inside the convex region (in this case, $v'v''$ and $v'_1v''_1$ extended in

<p>Algorithm FIND-SIP(L_x, L_y, p, q)</p> <ol style="list-style-type: none"> 1. $c_1 \leftarrow \text{RECT}(p, q)$ 2. $M_y \leftarrow \text{SEARCHVERT}(p, c_1, L_y)$ 3. $M_x \leftarrow \text{SEARCHHORZ}(c_1, q, L_x)$ 4. $M \leftarrow \text{REMOVE-POINTS}(M_y, M_x, p, q)$ 5. $i \leftarrow 1, m \leftarrow 0$ 6. $\text{APPEND}(\pi[m], p)$ 7. while ($M[i] \neq q$) 8. if ($M[i] = c_1$) 9. $\text{APPEND}(\pi[m], c_1)$ 10. $i \leftarrow i + 1$ 11. continue 12. $\text{APPEND}(\pi[m], M[i])$ 13. $\text{TRAVERSE}(M[i], M[i + 1], \pi, m)$ 14. $\text{APPEND}(\pi[m], M[i + 1])$ 15. $i \leftarrow i + 2$ 16. $\text{APPEND}(\pi[m], q)$ 17. $\text{DECOMPOSE}(\pi)$ 	<p>Procedure TRAVERSE ($L, M[i], M[i + 1], \pi, m$)</p> <ol style="list-style-type: none"> 1. if ($\text{index}[M[i]] <$ $\text{index}[M[i + 1]]$) 2. $l' \leftarrow \lfloor (\text{index}[M[i]] + 1) \rfloor$ 3. $l'' \leftarrow \lceil (\text{index}[M[i + 1]] - 1) \rceil$ 4. $\text{APPEND}(\pi[m], L[l'])$ 5. $j \leftarrow l' + 1$ 6. while ($j \leq l''$) 7. $\text{APPEND}(\pi[m], L[j])$ 8. if ($(\text{type}[\pi[m - 2]] = 1 \wedge$ $\text{type}[\pi[m - 1]] = 1)$) 9. $\text{APPLY-RULE}(\pi[m - 2],$ $\pi[m - 1])$ 10. $j \leftarrow j + 1$ 11. else 12. $l' \leftarrow \lceil (\text{index}[M[i]] - 1) \rceil$ 13. $l'' \leftarrow \lfloor (\text{index}[M[i + 1]] + 1) \rfloor$ 14. $\text{APPEND}(\pi[m], L[l'])$ 15. $j \leftarrow l' - 1$ 16. while ($j \geq l''$) 17. $\text{APPEND}(\pi[m], L[j])$ 18. if ($(\text{type}[\pi[m - 2]] = 1) \wedge$ $\text{type}[\pi[m - 1]] = 1)$) 19. $\text{APPLY-RULE}(\pi[m - 2],$ $\pi[m - 1])$ 20. $j \leftarrow j - 1$
<p>Procedure APPLY-RULE($\pi[m - 2], \pi[m - 1]$)</p> <ol style="list-style-type: none"> 1. if ($\text{CHKVTX}(\pi[m - 3], \dots, \pi[m])$) then $\text{APPLY-R2}(\pi[m - 2], \pi[m - 1])$ 2. else if ($l_1 = l_3$) 3. $\text{APPLY-R11}(\pi[m - 2], \pi[m - 1])$ 4. else if ($l_1 < l_3$) 5. $\text{APPLY-R12}(\pi[m - 2], \pi[m - 1])$ 6. else if ($l_1 > l_3$) 7. $\text{APPLY-R13}(\pi[m - 2], \pi[m - 1])$ 	

Fig. 9. The algorithm and related procedures to find SIP and its monotone path(s)

either directions). Let the line of concavity at the minimum distance intersects P at u' and u'' . Then, v_2 and v_3 are removed, and the modified path becomes $v_0v_1u''u'v_4$ (Fig. 8(b,d)), path-length reduced by $2l'$. As stated in Theorem 2, since there is a concave portion of P inside the convex region, the path becomes non-monotone at this point.

5 Proposed Algorithm

The algorithm FIND-SIP (Fig. 9) takes L, L_x, L_y , and the points p and q as input. In Step 1, the procedure RECT computes the corner point, c_1 , of R . The list of intersection points, M_y (M_x), of pc_1 (c_1q) with P , is found by searching L_y (L_x) (Steps 2-3). The list M of all intersection points of P with left semi-border¹ of R is obtained by REMOVE-POINTS in Step 4. At first, p is included in

¹ It can be done with *right semi-border* consisting of top and right borders of R , as well; in either case, we get a SIP.

the path π (Step 6). Let $M = \langle p, w_1, w_2, \dots, w_k, q \rangle$. When the left semi-border of R lies inside P , (semi-border of) R is traversed; otherwise, P . So pw_1 is traversed along R , then w_1w_2 along P , next w_2w_3 along R , again w_3w_4 along P , and so on (Steps 7-15). When c_1 is encountered, it is added to the path π , and the index i is incremented by 1 (Steps 8-9). After adding the first point $M[i]$ of the pair $(M[i], M[i+1])$ to π , P is traversed from $M[i]$ to $M[i+1]$ (Step 13), and points are added in π by TRAVERSE, until $M[i+1]$ is reached and added to π (Step 14). In Step 15, the index is incremented by 2 in order to consider the next pair for traversal. Lastly, q is added to π . In Step 17, π is decomposed into monotone sub-paths (Sec. 4.2:R2), if it is not monotone.

In the procedure TRAVERSE, if the index of $M[i]$ is less than that of $M[i+1]$, then P is traversed in an anticlockwise manner (Steps 1-10) using L (Sec. 2); otherwise, P is traversed clockwise (Steps 11-20). In Steps 2-3, l' and l'' indicate the appropriate pointers to the neighbor vertices of $M[i]$ and $M[i+1]$ in L , taken appropriately. After adding $L[l']$ to π (Step 4 or 14), each vertex on the path is appended to π in the **while** loop (Steps 6-10 or 10-20), until the vertex $L[l'']$ is reached. Proper rules (Sec. 4.2) are applied by calling APPLY-RULE as and when necessary.

The proof of correctness of the algorithm is based on the argument similar to finding *the* shortest path between two points in a simple polygon by applying a rubberband algorithm [19]. Since the internal region containing p and q is connected, there exists a sequence of connected rectangles through which a path can be traced from p to q . Imagine that we lay out a loose rubberband through this region connecting p and q . Since obstacles, if any (when R intersects the border of P), are not holes but the vertices forming concavity lines, the position and hence the length of the rubberband, once tightened, will be unique. This minimum-length rubberband will be a sequence of straight line segments, L .

The rules used in our algorithm basically mutate the initial path segments to follow the rubberband path, with the exception that each line segment $l \in L$ is replaced by half-perimeter monotone sub-path of the empty rectangle R_l whose diagonal is l . Therefore, our algorithm finally produces a SIP between p and q . Note that, in the presence of holes, our algorithm may not yield a SIP.

5.1 Time Complexity

The inner isothetic cover P , which tightly inscribes the object A , is obtained as an ordered sequence of n grid points (vertices and edge points) and stored in L . This needs O/ng time, since the intersection of each grid edge with A is checked in $O(g)$ time, and the number of grid points visited is bounded by $\Theta(n)$. Note that, ng is linear in the number of pixels constituting the contour of A . Subsequently, the lists L_x and L_y are constructed from L in $O(n \log n)$ time.

In the algorithm FIND-SIP, $O(\log n)$ time is needed to search p and q (in L_y and L_x) and $O(n)$ time for rest of the intersection points (Steps 2 and 3). Step 4 requires $O(n)$ time, as number of points in L_x and L_y together is $O(n)$. The **while** loop calls TRAVERSE to apply the required rules through APPLY-RULE. Each of the two **while** loops in TRAVERSE executes for $|l'' - l'|$ (grid) points, and once processed, none of these points is processed again (by APPLY-RULE).

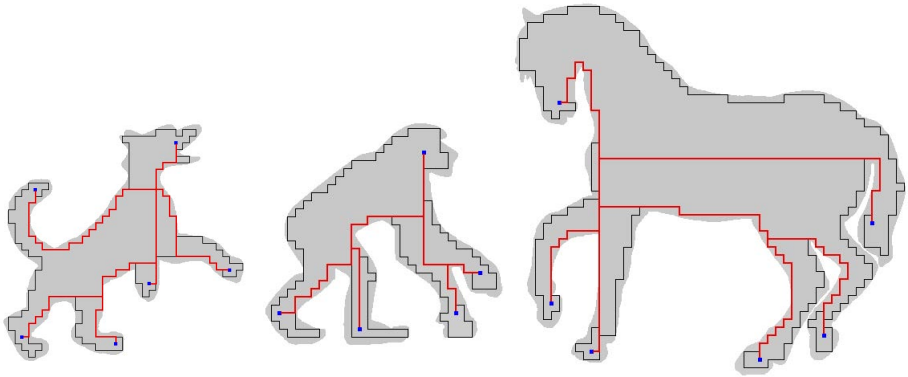


Fig. 10. Multiple SIPs (all in thick red lines) from a point (blue) near the mouth of dog, pre-man, horse images to other points (blue) in different parts of the body. See electronic version for the original color.

Logo 1293	$m = 2$	$m = 2$	$m = 1$
Logo 401	$m = 3$	$m = 2$	$m = 1$
Logo 421	$m = 1$	$m = 1$	$m = 1$

Fig. 11. Experimental results of SIPs with one or more monotone paths. See electronic version for the original color.

Hence, processing of each point in TRVERSE—and of each point in M , thereof—is done at most once. Applying a rule from R1 (R11, R12, R13) needs constant time. For R2, the pair(s) of concave vertices (type **33**) is searched in L_x or L_y . For the first pair, $O(\log n)$ time is needed. For all subsequent pairs, $O(n)$ time is needed in total, since the points are sorted lexicographically in L_x and L_y , and a linear search is done iteratively from the concave pair dealt in the last step of the iteration. This explains $O(n)$ time for applying necessary rules on all concerned points. Thus, total time complexity to find SIP from P is $O(n)$. On including the time complexity of constructing P from A , the overall time complexity becomes $O(n \log n)$.

6 Experimental Results and Conclusion

The algorithm is implemented in C in Ubuntu 10.4, Kernel version 2.6.32-21-generic, Dual Intel Xeon Processor 2.8 GHz, 800 MHz FSB. It is tested on several datasets containing various digital images of different shapes and forms. In Fig. [11](#), results for several typical objects are given for few different point pairs. The monotone sub-paths are shown in red and green lines alternately. The number of monotone sub-paths, m , shows how SIP might be useful to provide an idea about the complexity of the object in terms of m .

The algorithm can be used to find SIPs for all pairs of grid points lying on the border of P . Whether all-pair SIPs can be obtained with a more efficient algorithm remains an open problem. For an object with one or more holes, computation of these SIPs demands further improvements. All-pair SIPs may be used to capture some shape-related signature [20](#). As an example, see Fig. [10](#) that shows how the SIPs are formed from a particular point inside a part of a shape to points lying in other parts. Observe the overlaps among these SIPs—they get diverted when they enter two different parts. Theoretical analysis, in fact, is required to find whether there exists any relation on ortho-convex covering [11](#) of P with these SIPs. For, as observed in the context of our work, if a SIP between p and q consists of more than one monotone sub-path, then no ortho-convex sub-polygon of P would contain both p and q . With these interesting facts and findings, we conclude that SIP can be used for both theoretical developments and practical applications related to image analysis.

Acknowledgment. This research work is funded by INSPIRE Program under Department of Science and Technology, Government of India.

References

1. Amini, A.A., Weymouth, T.E., Jain, R.C.: Using dynamic programming for solving variational problems in vision. *IEEE Trans. PAMI* 12(9), 855–867 (1990)
2. Arkin, E., Mitchell, J., Piatko, C.: Minimum-link watchman tours. *IPL* 86, 203–207 (2003)
3. de Berg, M.: On rectilinear link distance. *CGTA* 1(1), 13–34 (1991)
4. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: *Computational Geometry Algorithms and Applications*. Springer, Heidelberg (2008)

5. de Berg, M., van Kreveld, M., Nilsson, B.J., Overmars, M.H.: Finding Shortest Paths in the Presence of Orthogonal Obstacles Using a Combined L_1 and Link Metric. In: Gilbert, J.R., Karlsson, R. (eds.) SWAT 1990. LNCS, vol. 447, pp. 213–224. Springer, Heidelberg (1990)
6. Biswas, A., Bhowmick, P., Bhattacharya, B.B.: TIPS: On Finding a Tight Isothetic Polygonal Shape Covering a 2D Object. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 930–939. Springer, Heidelberg (2005)
7. Biswas, A., Bhowmick, P., Bhattacharya, B.B.: Construction of isothetic covers of a digital object: A combinatorial approach. JVCIR 21(4), 295–310 (2010)
8. Buckley, M., Yang, J.: Regularised shortest-path extraction. PRL 18(7), 621–629 (1997)
9. Chin, W.P., Ntafos, S.: The zookeeper route problem. Information Sc. 63(3), 245–259 (1992)
10. Clarkson, K.L., Kapoor, S., Vaidya, P.: Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In: Proc. SCG, pp. 251–257 (1987)
11. Culberson, J.C., Reckhow, R.A.: Orthogonally convex coverings of orthogonal polygons without holes. J. Computer and Sys. Sc. 39(2), 166–204 (1989)
12. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. J. Algorithms 48(1), 135–159 (2003)
13. Farin, G., Hoschek, J., Kim, M.S.: Handbook of Computer Aided Geometric Design. Elsevier (2002)
14. Ghosh, S.K.: Visibility Algorithms in the Plane. Cambridge University Press (2007)
15. Gudmundsson, J., Levkopoulos, C.: A Fast Approximation Algorithm for TSP with Neighborhoods and Red-Blue Separation. In: Asano, T., Imai, H., Lee, D.T., Nakano, S.-i., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627, pp. 473–482. Springer, Heidelberg (1999)
16. Inkula, R., Kapoor, S.: Planar rectilinear shortest path computation using corridors. CGTA 42, 873–884 (2009)
17. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Picture Analysis. Morgan Kaufmann (2004)
18. Larson, R.C., Li, V.O.: Finding minimum rectilinear distance paths in the presence of barriers. Networks 11, 285–304 (1981)
19. Li, F., Klette, R.: Finding the Shortest Path Between Two Points in a Simple Polygon by Applying a Rubberband Algorithm. In: Chang, L.-W., Lie, W.-N. (eds.) PSIVT 2006. LNCS, vol. 4319, pp. 280–291. Springer, Heidelberg (2006)
20. Ling, H., Jacobs, D.W.: Shape classification using the inner-distance. IEEE Trans. PAMI 29, 286–299 (2007)
21. Lozano, P.T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Magazine Comm. ACM 22(10), 560–570 (1979)
22. Ntafos, S.: Watchman routes under limited visibility. CGTA 1, 149–170 (1992)
23. de Resende, P.J., Lee, D.T., Wu, Y.F.: Rectilinear shortest paths with rectangular barriers. In: Proc. SCG, pp. 204–213 (1985)
24. Sharir, M., Schorr, A.: On shortest paths in polyhedral spaces. In: Proc. STOC, pp. 193–215 (1986)
25. Sun, C., Pallottino, S.: Circular shortest path in images. PR 36(3), 709–719 (2003)
26. Tan, X.: Approximation algorithms for the watchman route and zookeeper’s problems. Discrete App. Maths. 136, 363–376 (2004)
27. Tan, X., Hirata, T.: Finding shortest safari routes in simple polygons. IPL 87(4), 179–186 (2003)
28. Wei, X.: Monotone path queries and monotone subdivision problems in polygonal domains. Ph.D. thesis, Hong Kong Univ. Sc. & Tech. (2010)

Fast Slicing of Orthogonal Covers Using DCEL

Nilanjana Karmakar¹, Arindam Biswas¹, and Partha Bhowmick²

¹ Department of Information Technology,
Bengal Engineering and Science University, Shibpur, Howrah, India
{nilanjana.nk,barindam}@gmail.com

² Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur, India
bhowmick@gmail.com

Abstract. A combinatorial algorithm to find the intersection of an axis-parallel slicing plane with the orthogonal cover of a digital object is presented in this paper. The orthogonal cover is the smallest-volume 3D orthogonal polyhedron containing the object and stored in a doubly connected edge list (DCEL). Its intersection with the slicing plane consists of one or more isothetic polygons that provide topological information about the cover. The algorithm traverses the vertices on the polyhedron boundary and lying on the slicing plane, and the direction of next move from a vertex is determined by a set of linear equations in the integer domain. Results for few objects with successive slicing planes are presented to demonstrate the effectiveness and elegance of the algorithm.

Keywords: 3D orthogonal cover, 3D orthogonal polyhedron, 3D orthogonal polyhedron, 3D image analysis, DCEL.

1 Introduction

Extraction and analysis of features, pertaining to shape and peripheral structure of 3D digital objects, is an important field in the domain of 3D image analysis. Given a 3D digital object A imposed on a 3D grid \mathbb{G} (Sec. 2), a technique on finding its *orthogonal cover* $\overline{P}_{\mathbb{G}}(A)$ has recently been proposed in [11]. As a further step towards acquiring a rough-and-fast estimate on structural details of the object, this paper presents an efficient algorithm to find the intersection of the 3D orthogonal cover with a given *slicing plane* parallel to one of the coordinate planes.

A set of slice polygons is found to be effective in various CAD applications in the field of Mechanical and Architectural drafting, in order to convey information about cross-sections of machine parts and engineering assemblies [13,2]. Determination of cross-sections or slice polygons in an algorithmic manner is in demand in several state-of-the-art tools, such as 3D image editing, in order to visualize the plan and the section in the layout document of a building model [9,10,15]. For exporting a 3D model as bitmap images in computer-enabled applications,

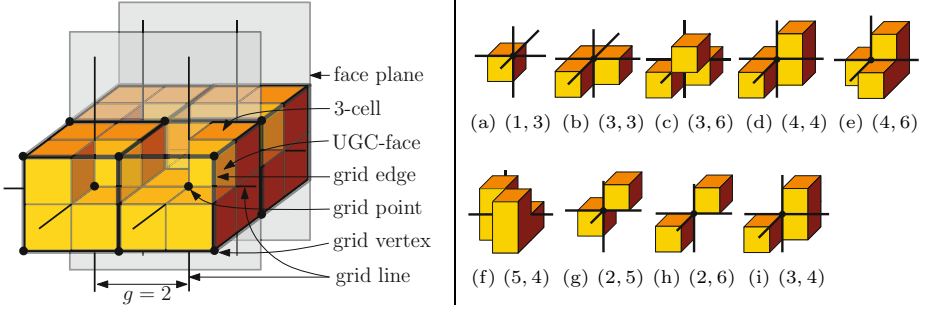


Fig. 1. Left: α -adjacent 3-cells for $g = 2$. Right: (a)–(f): Types of vertex; (g)–(i): Types of pseudo-vertex. See electronic version for the original color.

slice polygons are also used [5]. The differences in the structure of consecutive slices can be used to study or analyze object topology. The proposed technique can also be useful in the field of digital tomography to distinctly identify certain parts of an object lying on the focal plane while other redundant portions are blurred [7,17]. Techniques that highlight single layers are used in medical imaging, and some related works may be seen in [7,8,16].

The paper is organized as follows. Preliminary ideas on theoretical foundation are given in Sec. 2. The algorithm to extract successive slices from the orthogonal cover is explained and demonstrated in Sec. 3. Experimental results and concluding notes are given in Sec. 4 and Sec. 5 respectively.

2 Definitions and Preliminaries

We start with the theoretical foundation in the context of this work. A *digital object* A is defined as a finite subset of \mathbb{Z}^3 , with all its constituent points (i.e., voxels) having integer coordinates and connected in 26-neighborhood. Each voxel is equivalent to a *3-cell* [12] centered at the concerned integer point (Fig. 1(Left)).

2.1 Digital Grid

The orthogonal cover of A is obtained w.r.t. a digital grid in 3D digital space. A *digital grid* \mathbb{G} consists of three orthogonal sets of equi-spaced grid lines, \mathbb{G}_{yz} , \mathbb{G}_{zx} , and \mathbb{G}_{xy} , where $\mathbb{G}_{yz} = \{l_x(j \pm ag, k \pm bg) \mid a \in \mathbb{Z}, b \in \mathbb{Z}\}$. Similarly, \mathbb{G}_{zx} and \mathbb{G}_{xy} can be represented in terms of l_y and l_z for a grid size $g \in \mathbb{Z}^+$. Here, $l_x(j, k) = \{(x, j, k) : x \in \mathbb{R}\}$, $l_y(i, k) = \{(i, y, k) : y \in \mathbb{R}\}$, and $l_z(i, j) = \{(i, j, z) : z \in \mathbb{R}\}$ denote the *grid lines* (Fig. 1(Left)) along x -, y -, and z -axes respectively, where i , j , and k are integer multiples of g . The three orthogonal lines $l_x(j, k)$, $l_y(i, k)$, and $l_z(i, j)$ intersect at the point $(i, j, k) \in \mathbb{Z}^3$, which is called a *grid point*; a shift of $(\pm 0.5g, \pm 0.5g, \pm 0.5g)$ with respect to a grid point designates a *grid vertex*, and a pair of adjacent grid vertices defines a *grid edge* [12] (Fig. 1(Left)).

A grid, as defined above, is characterized by several elements (Fig. 1(Left)). A *unit grid cube* (UGC) is a (closed) cube of length g whose vertices are *grid*

vertices, edges constituted by *grid edges*, and faces constituted by *grid faces*. Each face of a UGC lies on a *face plane* (henceforth referred as a UGC-face), which is parallel to one of three coordinate planes. Clearly, each *face plane*, containing coplanar UGC-faces, is at a distance of integer multiple of g from its parallel coordinate plane. A UGC-face, f_k , has two adjacent UGCs, U_1 and U_2 , such that $f_k = U_1 \cap U_2$. The *interior* of a UGC is the open cubical region lying strictly inside the UGC. A smaller (larger) value of g implies a denser (sparser) grid. For $g = 1$, the grid \mathbb{G} essentially corresponds to \mathbb{Z}^3 . As each grid point p is equivalent to a 3-cell c_p centered at p for $g = 1$, each face of c_p is a *grid face* lying on a *face plane*, which is parallel to a coordinate plane. A UGC consists of $g \times g \times g$ voxels and each UGC-face consists of $g \times g$ voxels.

Two 3-cells c_1 and c_2 having centers at $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ respectively are α -adjacent if and only if $p_1 \neq p_2$ and $c_1 \cap c_2$ contains an α -cell ($\alpha \in \{0,1,2\}$) [12]. The grid points p_1 and p_2 are in k -neighborhood where $k \in \{6, 18, 26\}$; $k = 6$ denotes *2-adjacency*, $k = 18$ denotes *1-adjacency*, and $k = 26$ denotes *0-adjacency* in the cell model. For a grid point $p(x, y, z)$, its respective k -neighborhoods for $k = 6, 18, 26$ are given by

$$\begin{aligned} N_6(p) &:= \{p' : p' \in \mathbb{Z}^3 \wedge L_1(p, p') = 1\}, \\ N_{18}(p) &:= \{p' : p' \in \mathbb{Z}^3 \wedge L_1(p, p') \in \{1, 2\} \wedge L_\infty(p, p') = 1\}, \\ N_{26}(p) &:= \{p' : p' \in \mathbb{Z}^3 \wedge L_1(p, p') \in \{1, 2, 3\} \wedge L_\infty(p, p') = 1\}, \end{aligned}$$

where, $p' = (x', y', z')$, $L_\infty(p, p') = \max\{|x - x'|, |y - y'|, |z - z'|\}$, and $L_1(p, p') = |x - x'| + |y - y'| + |z - z'|$. Each point in $N_k(p)$ is said to be a k -neighbor of p . Two points p and q are k -connected in a digital set $A \subseteq \mathbb{Z}^3$ if and only if there exists a sequence $\langle p := p_0, p_1, \dots, p_n := q \rangle \subseteq A$ such that $p_i \in N_k(p_{i-1})$ for $1 \leq i \leq n$. For any point $p \in A$, the set of points that are k -connected to $p \in A$ is called a k -connected component of A . In other words, a k -connected component of a nonempty set $A \subseteq \mathbb{Z}^3$ is a maximal k -connected set of A . If A has only one connected component, it is called a k -connected set.

2.2 Orthogonal Cover

An *orthogonal polyhedron* is a 3D polytope with all its vertices as grid vertices, all its edges made of grid edges, and all its faces lying on face planes. Each face of an orthogonal polyhedron is an *isothetic polygon* whose alternate edges are orthogonal and constituted by grid edges of \mathbb{G} . The *orthogonal cover* of an object A is then defined as an orthogonal polyhedron $\overline{P}_{\mathbb{G}}(A)$ that tightly circumscribes A , i.e., the minimum-volume polyhedron containing A .

Problem Statement. Let A be a 3D digital object, Π , the *slicing plane* parallel to yz -, zx -, or xy -plane, and $\overline{P}_{\mathbb{G}}(A)$, the corresponding orthogonal cover for grid size g . For simplicity, Π is treated as the finite rectangular plane touching the bounding cuboid of A with its four sides. The objective is to find the intersection of Π with $\overline{P}_{\mathbb{G}}(A)$, and report the resultant *slice* as a set of 2D isothetic polygons. Figure 2 shows the orthogonal cover (obtained by the algorithm in [11]) of Stanford Bunny, and the results by slicing planes parallel to yz -plane.

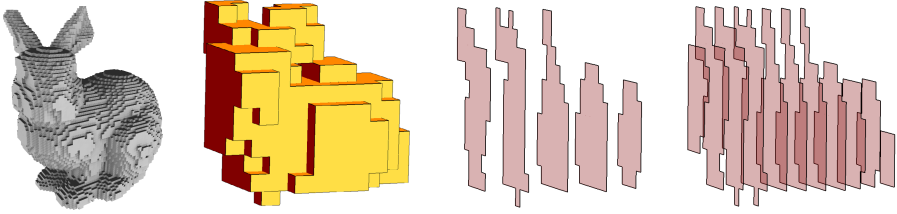


Fig. 2. Slices from the orthogonal cover of *Stanford Bunny* with face planes parallel to yz -plane. From left to right: Voxelated object A ; orthogonal cover Π for $g = 10$; Slicing by some planes; Slicing by all face planes. See electronic version for the original color.

3 Slicing the Cover

The cover $\overline{P}_{\mathbb{G}}(A)$ is stored in a Doubly Connected Edge List (DCEL), which is a geometric data structure that records topological information about a 2D subdivision (embedded in 3D space) as a collection of vertex list V , edge list E , and face list F [14,3]. The face list F contains an entry corresponding to an edge of each face of the cover and the whole face can be traversed from the edges listed in E . A brief outline of the algorithm is given in Fig. 3, the details being explained from Sec. 3.1 onwards.

```

01. for each polytope face  $f_i$  lying on  $\Pi$ 
02.   classify the grid vertices on boundary of  $f_i$ 
       as vertex, pseudo-vertex, or edge point (Sec. 3.2)
03. for each polytope face  $f_j$  perpendicular to  $\Pi$ 
04.   classify the grid vertices on boundary of  $f_j$ 
       and lying on  $\Pi$  as isolated point (Sec. 3.2)
05. for each grid vertex  $q$  on  $\Pi$ 
06.   if  $q = v_s$  (start point) (Sec. 3.3)
07.     set grid vertex  $v \leftarrow next[v_s]$ 
08.     set  $visited[v] \leftarrow \text{TRUE}$ 
09.     do
10.       find direction of traversal through  $v$  (Sec. 3.4)
11.       set  $v \leftarrow next[v]$ 
12.       set  $visited[v] \leftarrow \text{TRUE}$ 
13.     while ( $v \neq v_s$ )
14.   set  $visited[v_s] \leftarrow \text{TRUE}$ 

```

Fig. 3. Brief outline of the proposed algorithm

3.1 Object Occupancy

Let $\mathcal{U}_q = \{U_q^{(i)} : i = 1, 2, \dots, 8\}$ be the set of eight UGCs incident at q , where $U_q^{(i)}$ denotes the i th UGC incident at q . If $U_q^{(i)}$ contains an object voxel, then $U_q^{(i)}$ is said to have *object occupancy*, defined as follows.

$$\Phi(U_q^{(i)}) = \begin{cases} 0 & \text{if } U_q^{(i)} \cap A = \emptyset \\ 1 & \text{if } U_q^{(i)} \cap A \neq \emptyset \end{cases} \quad (1)$$

The UGC-face $f = U_q^{(i)} \cap U_q^{(j)}$ is said to have *object occupancy* if $(A \cap U_q^{(i)}) \vee (A \cap U_q^{(j)}) = 1$. From above equation, the number of UGCs incident at q having object occupancy is given by

$$m_q = \sum_{U_q^{(i)} \in \mathcal{U}_q} \Phi(U_q^{(i)}). \quad (2)$$

Depending on the value of m_q and the arrangement of the object-containing UGCs, q is classified as a polyhedron vertex, a pseudo-vertex, or an edge point, as explained next.

3.2 Grid Vertex Classification

A vertex q of $\overline{P}_{\mathbb{G}}(A)$ essentially coincides with some grid vertex of \mathbb{G} and is assigned a tuple (m_q, n_q) , where n_q is number of edges of $\overline{P}_{\mathbb{G}}(A)$ incident at q . Six arrangements of object-occupied neighboring UGCs, namely (1, 3), (3, 3), (3, 6), (4, 4), (4, 6), and (5, 4) \square , as illustrated in Fig. \square (a-f), classify a grid vertex as a *vertex* of $\overline{P}_{\mathbb{G}}(A)$. Three other arrangements, namely (2, 5), (2, 6), and (3, 4) (Fig. \square (g-i)) do not satisfy the criteria of being a vertex; however, these grid vertices (always) will be the vertices of the slice (intersection) polygon,

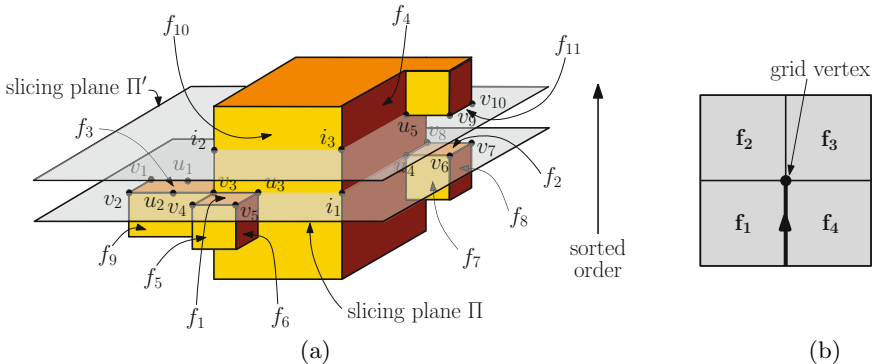


Fig. 4. (a) Process of classification of grid vertices. (b) UGC-faces neighboring a grid vertex w.r.t. an incoming direction.

called *pseudo-vertices*. For all other possible configurations of the grid vertex q of $\overline{P_G}(A)$, the grid vertex is an *edge point*. An edge point has all the four neighboring UGC-faces, lying on the slicing plane, as occupied. Hence, such a grid vertex can never qualify as the start point of a slice polygon (face or hole).

Observe that a set F_Π of polygonal faces of $\overline{P_G}(A)$ lies on Π , while other faces (set F'_Π) whose edges are either orthogonally incident on Π or lie on Π are perpendicular to Π . In Fig. 4(a), $F_\Pi = \{f_1, f_2, f_3\}$ and $F'_\Pi = \{f_4, f_5, \dots, f_{10}\}$. The set V_Π of grid vertices (v_1, v_2, u_2 , etc.) on the boundaries of faces in F_Π are classified as above. Let V'_Π be the set of grid vertices on the boundaries of faces in F'_Π . So, vertices in V'_Π , which also belong to V_Π , get classified. The remaining vertices in V'_Π are marked as *isolated points* (e.g., i_1). Each grid vertex of Π is initialized as unvisited. Note that, while tracing a slice polygon, each vertex or edge point or isolated point is visited once, but each pseudo-vertex is visited twice.

The grid vertex classification facilitates the determination of start point of traversal of the grid vertices in order to trace the slice polygon. The start vertex of a face polygon is always considered such that only one of its neighboring UGC-faces lying on the slicing plane is occupied, and for a hole polygon three UGC-faces are occupied. This concept is analogous to the derivation of start vertex in 4. A grid vertex of type (1, 3) denotes a start vertex of face polygon, and that of type (3, 3), (3, 6), (4, 4), (4, 6), or (5, 4) denotes a start vertex of hole polygon. The concept of cross-vertex in 2D (as proposed in 4) is not extended to 3D. As a result, a pseudo-vertex (type (2, 5), (2, 6), or (3, 4)) may also act as the start point of a face polygon, as explained next in Sec. 3.3.

3.3 Start Point Identification

$\overline{P_G}(A)$ consists of polygonal faces parallel to yz -, zx -, and xy -planes. Some of these faces lie on the intersecting region Π . Thus, initially we have a set of disconnected faces lying on Π . An unvisited vertex, a pseudo-vertex visited once, or an unvisited isolated point belonging to such a face may qualify as a start point. A pseudo-vertex acts as the start point if and only if there are more than one slice polygons. A pseudo-vertex has object occupancy in exactly one pair of diagonally opposite UGC-faces, i.e., $(\mathbf{f}_1, \mathbf{f}_3)$ or $(\mathbf{f}_2, \mathbf{f}_4)$, shown in Fig. 4(b)). Starting from a pseudo-vertex, the boundary of the UGC-face \mathbf{f}_3 (\mathbf{f}_2) will be traversed while tracing a slice polygon, but UGC-face \mathbf{f}_1 (\mathbf{f}_4) will be left out. This implies that another slice polygon must have been traced previously that includes the UGC-face \mathbf{f}_1 (\mathbf{f}_4). To find the start point of the slice polygon, Π is scanned from bottom-back if Π is parallel to yz -plane, from left-back if parallel to zx -plane, and bottom-left if parallel to xy -plane, until a grid vertex is found to be qualifying as a start point, v_s ; the respective start directions (Sec. 3.4) are set to (0, 1, 0), (0, 0, 1), and (1, 0, 0). The slice polygon is traced from v_s and concludes when it returns to v_s . The scan resumes in order to find any other unvisited grid vertex that qualifies as a new start point for tracing another slice polygon.

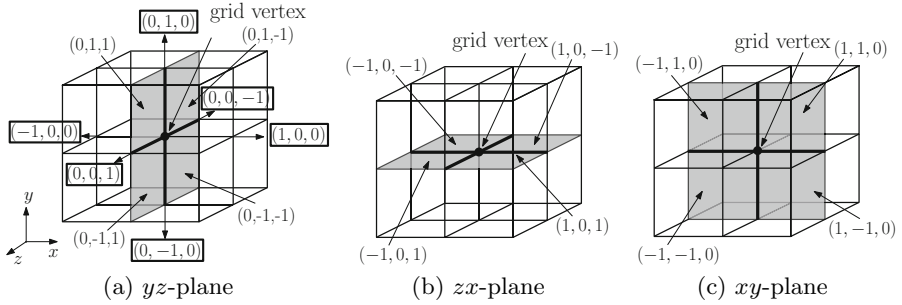


Fig. 5. Neighboring UGC-faces of a grid vertex along different planes

3.4 Direction of Traversal

Traversal through a grid vertex is possible in 6 directions: $\{(\pm 1, \pm 1, \pm 1)\}$ (Fig. 5(a)). Out of these, along a specified plane Π , traversal is possible in 4 directions; e.g., for $\Pi : y = \beta$, $\{(\pm 1, 0, \pm 1)\}$ (Fig. 5(b), bold black lines through the grid vertex). Any grid vertex q lying on the path of traversal would have an incoming direction and an outgoing direction depending on the topology at q . In particular, corresponding to an incoming direction, there can be at most 3 outgoing directions. Thus, $4 \times 3 = 12$ incoming-outgoing combinations are possible along a given plane, Π .

Since a grid vertex q is common to 4 UGC-faces along Π , the direction of tracing depends on the object occupancy of these 4 UGC-faces named clockwise as \mathbf{f}_1 , \mathbf{f}_2 , \mathbf{f}_3 , and \mathbf{f}_4 with respect to the incoming direction, as shown in Fig. 4(b). Fig. 5 shows 3 such sets, each with 4 UGC-faces, along three coordinate planes. Since a face is always traversed anti-clockwise, the direction of traversal at q ensures that the object always lies left. During the traversal in the direction \mathbf{d} from the current grid vertex $\mathbf{q}_i(x_i, y_i, z_i)$, the next grid vertex $\mathbf{q}_o(x_o, y_o, z_o)$ is given by

$$\mathbf{q}_o = \mathbf{q}_i + \mathbf{d} \cdot g \quad (3)$$

The outgoing direction \mathbf{d}_o at \mathbf{q}_o , given the incoming direction \mathbf{d}_i , is

$$\mathbf{d}_o = \begin{cases} \mathbf{d}_i + \mathbf{f}_1 & \text{if one UGC-face is occupied (i)} \\ -\mathbf{d}_i - \mathbf{f}_1 + \mathbf{f}_2 & \text{if two adjacent UGC-faces are occupied (ii)} \\ \mathbf{d}_i + \mathbf{f}_1 - \mathbf{f}_2 + \mathbf{f}_3 & \text{if 3 UGC-faces are occupied (iii)} \end{cases} \quad (4)$$

Here \mathbf{f} denotes the UGC-face occupied, expressed as a 3-element vector (Fig. 5). Evidently, Eqn. 4(i) denotes a left turn, Eqn. 4(ii) denotes no change in direction, and Eqn. 4(iii) denotes a right turn (Fig. 6). For instance, in Fig. 6(g), $\mathbf{f}_1 = (-1, -1, 0)$, $\mathbf{f}_2 = (-1, 1, 0)$, and $\mathbf{f}_3 = (1, 1, 0)$ are occupied; incoming direction $\mathbf{d}_i = (0, 1, 0)$. So, by Eqn. 4(iii), outgoing direction is $\mathbf{d}_o = \mathbf{d}_i + \mathbf{f}_1 - \mathbf{f}_2 + \mathbf{f}_3 = (0, 1, 0) + (-1, -1, 0) - (-1, 1, 0) + (1, 1, 0) = (1, 0, 0)$. Note that when two UGC-faces are diagonally occupied (pseudo-vertex), e.g., $(\mathbf{f}_1, \mathbf{f}_3)$, it is treated as if only UGC-face \mathbf{f}_1 is occupied, as stated in Eqn. 4(i).

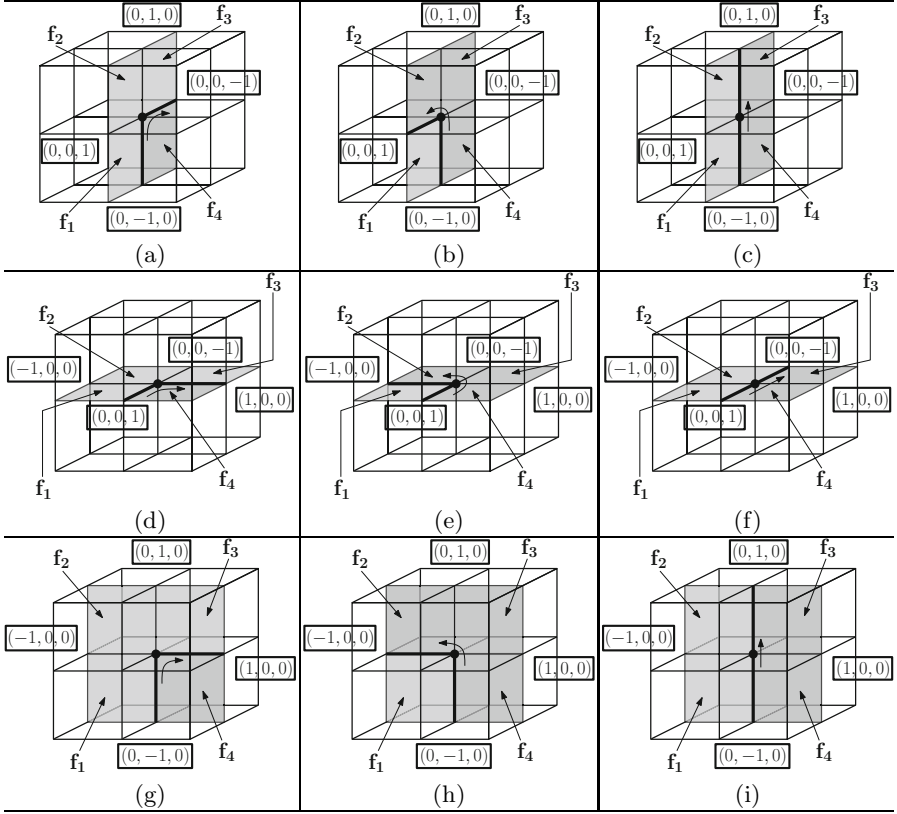


Fig. 6. Direction of traversal (along the bold black lines) through a grid vertex along yz -plane (a,b,c), zx -plane (d,e,f), and xy -plane (g,h,i). The occupied UGC-faces are marked light gray. The directions are labeled in boxes.

3.5 Algorithm SLICE

Given $\overline{P_G}(A)$ and Π , the faces of $\overline{P_G}(A)$ lying on Π are detected from F . If f_i lies on Π , then all the vertices of f_i are classified by CLASSIFY-VERTEX procedure as vertices, pseudo vertices, or edge points having values 1, 2, or 3, respectively (Algorithm SLICE: Steps 1-4 in Fig. 7). Next, the faces which are not lying on Π but having vertices or edge points on Π , are classified, and the concerned vertices or edge points are added to *Array* (Steps 6-8). Some of the grid vertices appearing in *Array* are already classified when the grid vertices belonging to faces on Π are classified in Steps 3-4. Rest of the grid vertices in *Array* are classified as isolated points with value 4 (Step 11).

Once all grid vertices on Π are classified, the slice polygon is determined starting from a grid vertex v_s and tracing a closed polygon. All grid vertices traversed in search of v_s are marked as *visited* (Step 26). For v_s , \mathbf{d}_o is initialized to a value depending on the coordinate plane with which Π is parallel $((0, 1, 0))$

<p>Algorithm SLICE($A, \mathbb{G}, \Pi, \overline{P}_{\mathbb{G}}(A)$)</p> <ol style="list-style-type: none"> 1. for each face $f_i \in F$ 2. if $plane[f_i] = \Pi$ 3. for each edge $e_{ij} \in f_i$ 4. $vertex[src[e_{ij}]]$ \leftarrow CLASSIFY-VERTEX($src[e_{ij}]$) 5. else 6. for each $e_{ij} \in f_i$ 7. if $src[e_{ij}]$ lies on Π 8. $Array[\Pi]$ \leftarrow $Array[\Pi] \cup src[e_{ij}]$ 9. for each point $t \in Array[\Pi]$ 10. if $vertex[t] = 0 \triangleright$ init value 11. $vertex[t] \leftarrow 4 \triangleright$ isolated point 12. for each grid vertex $q \in \Pi$ 13. if ($vertex[q] \in \{1, 4\}$ and $visited[q] = 0$) or ($vertex[q] = 2$ and $visited[q] = 1$) 14. $v_s \leftarrow q$ 15. $v \leftarrow v_s + \mathbf{d}_o \cdot g$ 16. $\mathbf{d}_i \leftarrow \mathbf{d}_o$ 17. do 18. $\mathbf{d}_o \leftarrow$ MOVE-DIRECTION(v, \mathbf{d}_i) 19. if $vertex[v] \in \{1, 3, 4\}$ 20. $visited[v] \leftarrow 1$ 21. else if $vertex[v] = 2$ 22. $visited[v] \leftarrow visited[v] + 1$ 23. $v \leftarrow v + \mathbf{d}_o \cdot g$ 24. $\mathbf{d}_i \leftarrow \mathbf{d}_o$ 25. while ($v \neq v_s$) 26. $visited[q] \leftarrow 1$ 	<p>Procedure CLASSIFY-VERTEX($src[e_{ij}]$)</p> <ol style="list-style-type: none"> 1. $q \leftarrow src[e_{ij}]$, $count \leftarrow 0$ 2. for each $U \in \mathcal{U}_q$ 3. if $U \cap A \neq \emptyset$ 4. $count \leftarrow count + 1$ 5. if ($count = 1$) or ($count = 3$ and UGCs are 1-adjacent or 0-adjacent) or ($count = 4$ and three UGCs are 1-adjacent and 4th UGC 2-adjacent to any one) or ($count = 5$ and three UGCs are 1-adjacent and two of these are 2-adjacent to two other UGCs) 6. return 1 \triangleright vertex 7. else if ($count = 2$ and UGCs are 0-adjacent or 1-adjacent) or ($count = 3$ and UGCs are 0-adjacent and any one pair is 2-adjacent) 8. return 2 \triangleright pseudo-vertex 9. return 3 \triangleright edge point <p>Procedure MOVE-DIRECTION(v, \mathbf{d}_i)</p> <ol style="list-style-type: none"> 1. for each UGC-face \mathbf{f}_k incident at v $\triangleright k \in \{1, 2, 3, 4\}$ 2. if OCCUPANCY(\mathbf{f}_k) 3. $K \leftarrow K \cup \{\mathbf{f}_k\}$ 4. if $K = \{\mathbf{f}_1\}$ 5. $\mathbf{d}_o \leftarrow \mathbf{d}_i + \mathbf{f}_1$ 6. else if $K = \{\mathbf{f}_1, \mathbf{f}_2\}$ 7. $\mathbf{d}_o \leftarrow -\mathbf{d}_i - \mathbf{f}_1 + \mathbf{f}_2$ 8. else if $K = \{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3\}$ 9. $\mathbf{d}_o \leftarrow \mathbf{d}_i + \mathbf{f}_1 - \mathbf{f}_2 + \mathbf{f}_3$ 10. return \mathbf{d}_o
---	--

Fig. 7. The slicing algorithm and related procedures

for yz -, $(0, 0, 1)$ for zx -, and $(1, 0, 0)$ for xy -plane), and the outgoing direction at each subsequent grid vertex serves as the incoming direction of the next grid vertex (Steps 16 and 24). The grid vertices are visited one by one in the **do while** loop (Steps 17-25) until the traversal reaches v_s (and new polygons are searched for). Consequently, the vertices, edge points, and isolated points are marked as *visited* (Steps 19-20), and the number of visits to a pseudo-vertex is updated (Steps 21-22).

The algorithm reports hole polygons also formed on the slicing plane. This is achieved by keeping the object always to the left during traversal. The algorithm is not affected by a protrusion on the surface of $\overline{P}_{\mathbb{G}}(A)$, since a grid vertex at the base of a protrusion never qualifies as the start point of a slice polygon and the slice polygon containing the former has already been generated.

The proof of completion of tracing each slice polygon, and the proof of correctness of the algorithm thereof, is as follows. The outgoing direction \mathbf{d}_o from a grid vertex v depends on the incoming direction \mathbf{d}_i and the set of occupied UGC-faces incident at v and lying on Π . Accordingly, the traversal advances to the next grid vertex in the direction of \mathbf{d}_o . The object always lies to the left during traversal. This is in analogy with the traversal of an outer isothetic cover of a 2D digital object [4], and hence finally v_s is reached. While scanning Π , every grid vertex v is considered at least once. Either v qualifies as a start point or it has already been traced in a previously traced slice polygon. If v qualifies as a start point, tracing a new polygon starts from there.

Preprocessing: We sort the vertices, edges, and faces (isothetic polygons) of $\overline{P}_{\mathbb{G}}(A)$ according to the direction of motion of the slicing plane. For example, in Fig. 4(a), all the vertices and pseudo-vertices belonging to f_1, f_2, \dots are sorted in the order of non-decreasing y -coordinate and stored in a list V' . Thus, $v_1, v_2, v_3, \dots, v_8$ belonging to faces f_1, f_2 , and f_3 on the slicing plane Π , are followed by v_9, v_{10} , etc. belonging to face f_{11} lying on the next slicing plane Π' . The face list F and the edge list E are also sorted accordingly in F' and E' . Now consider the intermediate step when the vertices on Π are obtained. Starting from v_1 , we check whether the grid vertex immediately above it, i.e., lying on the next slicing plane Π' , belongs to $\overline{P}_{\mathbb{G}}(A)$. This is checked from V' since vertices on Π' occur in V' immediately after the vertices on Π . For v_1 and v_2 , it reports none. For v_3 , the grid vertex i_2 lying on Π' belongs to $\overline{P}_{\mathbb{G}}(A)$. This follows from the fact that i_2 does not belong to V' but lies on an edge orthogonal to Π , as found from E' . So i_2 is an unvisited isolated point and taken as a start point while slicing by Π' . The start points for the slice at Π' are obtained in this manner. For the reverse case, when a grid vertex at Π is an isolated point, e.g., i_1 , then also the same procedure is applied to obtain the vertex (i_3) on Π' . For all other cases, similar arguments hold.

3.6 Time Complexity

Let n be the number of voxels constituting the object surface connected in 26-neighborhood. Since a UGC is a cube of length g , the number of UGCs that contains voxels from the object surface lies between $n_{u,\min} = O(n/g^3)$ and $n_{u,\max} = O(n/g)$. A UGC has six faces and contributes a maximum of five to the cover. Hence, the number of UGC-faces on the object surface is at least $n_{u,\min}$ and at most $n_{u,\max}$.

Best case: A UGC has six faces and if a face intersects the object A , then the UGC has object occupancy. As the face area is g^2 , object occupancy in a UGC is

verified in $O(g^2)$ time. Hence, time complexity for CLASSIFY-VERTEX is $O(g^2)$, since eight neighboring UGCs have to be checked for object occupancy; similarly, MOVE-DIRECTION also takes $O(g^2)$ time.

Outer **for** loop in Steps 1-8 of SLICE executes in $n_{u,\min} \cdot O(g^2) = O(n/g)$ time, since the time complexity of CLASSIFY-VERTEX is $O(g^2)$. The next dominant part is the **for** loop from Line 12 to Line 26. It traces all (isothetic) slice polygons for an input slicing plane, Π . *Without any preprocessing*, all $n_{u,\min}$ UGC-faces will be verified for their participation in formation of slice polygons, thus yielding a time complexity of $n_{u,\min} \times O(g^2) = O(n/g)$. In effect, the algorithm needs $O(n/g)$ time per slicing plane. However, *with preprocessing*—targeted to efficient slicing for successive slicing planes, we can have an improvement, as explained next.

Time needed to prepare V', E', F' is $O(|V| \log |V|)$, since $|E| = O(|V|)$ and $|F| = O(|V|)$. Note that, usually $|V|$ will be quite small compared to n , especially when Π has surface planarity; and in worst case, $|V| = O(n_{u,\min})$. Let L_y be the length of Π (measured in grid unit) along y -axis; then we have L_y slicing planes along y -axis. Over all these successive slicing planes, each UGC will be processed in $O(g^2)$ time, since a UGC will be processed for exactly two successive slicing planes. Hence, total time over L_y planes is $O(n/g)$. Now, if the three dimensions (L_x, L_y, L_z) of Π have comparable values or are sufficiently large in terms of grid unit g , then $L_y^2 = O(n_{u,\min})$, as the number of UGCs (on surface of Π) is $n_{u,\min}$. This yields an improved time complexity of $O(\sqrt{n}/g)$ per slice.

Worst case: Steps 1-8 of SLICE takes $n_{u,\max} \cdot O(g^2) = O(n/g)$ time. Hence, for the **for** loop from Line 12 to Line 26 (with preprocessing), $|V| = O(n_{u,\max})$ and total time over L_y planes is $O(n/g)$. With $L_y^2 = O(n_{u,\max})$, we get time complexity of $O(\sqrt{ng})$ per slice.

4 Experimental Results

The proposed algorithm has been implemented in C in Linux Fedora Release 7, Kernel version 2.6.21.1.3194.fc7, Dual Intel Xeon Processor 2.8 GHz, 800 MHz FSB. The algorithm has been tested on 3D orthogonal covers of several objects for slicing planes moving along three coordinate axes. A few are displayed below in Fig. 8 and Fig. 9, with results summarized in Table 1. Note that the average CPU time of slicing along a particular direction decreases with a decrease in the average size of the slices parallel to that plane. For instance, the average CPU time per slicing plane along x -axis for Pickup Van at $g = 2$ is 0.02 secs., and those along y - and z -axes are around 0.04 secs.; this is due to the difference in sizes of the corresponding slices (Fig. 9). For Stanford Bunny at $g = 10$, the respective slicing times are 0.6, 0.6, and 0.7 secs., which vary less as the average slice sizes along three directions have lesser variation. It is observed that the number of vertices, the number of edges, and the total area of the slices parallel to a coordinate plane increases with decrease in grid size for both Stanford Bunny and Pickup Van.

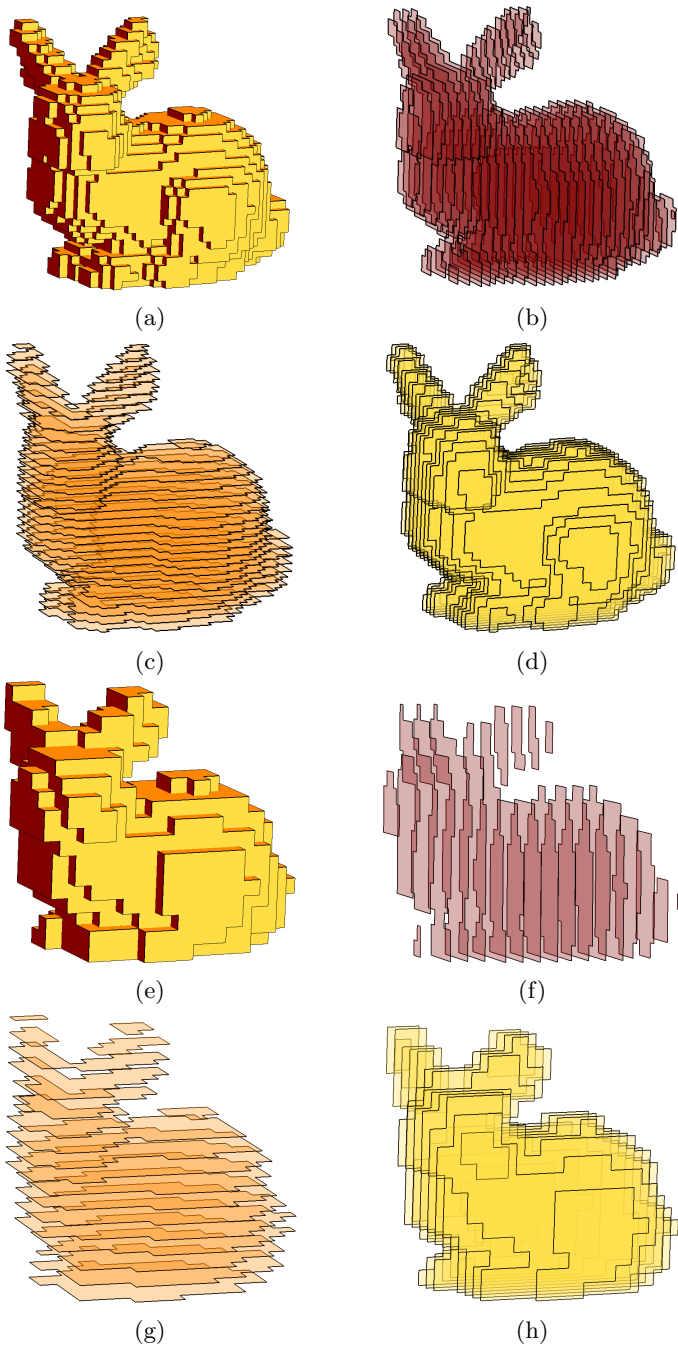


Fig. 8. Results of slicing by our algorithm on **Stanford Bunny**. (a, e) 3D orthogonal covers for $g = 5$ and $g = 10$; (b, f) Slices parallel to yz -plane; (c, g) Slices parallel to zx -plane; (d, h) Slices parallel to xy -plane. See electronic version for the original color.

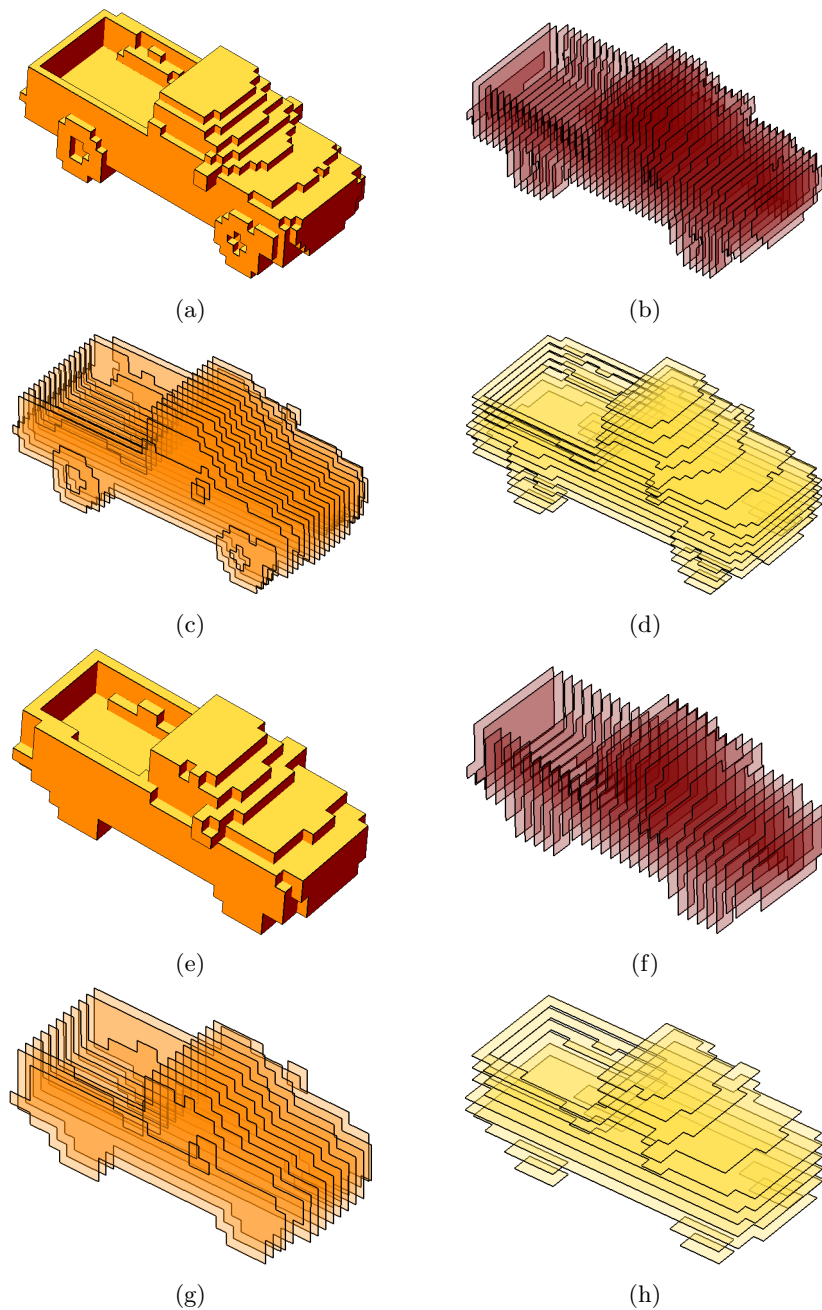


Fig. 9. Results of slicing by our algorithm on Pickup Van. (a, e) 3D orthogonal covers for $g = 2$ and $g = 3$; (b, f) Slices parallel to yz -plane; (c, g) Slices parallel to zx -plane; (d, h) Slices parallel to xy -plane. See electronic version for the original color.

Table 1. Statistical information and CPU time of slicing

Object	Statistical information										Average CPU time per slice (seconds)		
	Orthogonal cover			Slices									
				yz -plane			zx -plane			xy -plane			
$ V $	$ E $	$ F $	$ V $	$ E $	Σa	$ V $	$ E $	Σa	$ V $	$ E $	Σa		
Stanford Bunny ($g = 5$)	917	7358	895	1310	1277	881800	1391	1358	1355300	1104	1078	380125	1.1
Stanford Bunny ($g = 10$)	277	2016	243	378	360	826100	380	363	554200	332	318	307500	0.6
Pickup Van ($g = 2$)	287	3004	250	558	514	322996	602	582	117736	278	262	70472	0.03
Pickup Van ($g = 3$)	112	1436	112	324	294	76446	288	274	103905	126	115	66204	0.02

Σa = area over all slices parallel to a coordinate plane

5 Concluding Remarks

As found from our experimentation, the topology of an orthogonal cover—and of the original digital object, thereof—gets revealed from slice polygons obtained by intersection with successive slicing planes. As explained in Sec. II, these slices can be used for state-of-the-art applications, including shape analysis. For varying grid size (i.e., multiple resolution), significant differences in slice polygons along a particular slicing plane would carry important shape-related information, such as planarity, concavity, convexity, etc. Hence, shape analysis in 3D domain with multi-resolution slicing would be an engrossing work, which may be pursued for further applications.

Acknowledgement. A part of this research is funded by Council of Scientific and Industrial Research (CSIR), Government of India.

References

1. Aguilera, A.: Orthogonal Polyhedra: Study and Application. PhD Thesis, Universitat Politècnica de Catalunya (April 1998)
2. AutoCAD, <http://docs.autodesk.com>
3. Berg, M.D., Cheong, O., Kreveld, M.V., Overmars, M.: Computational Geometry—Algorithms and Applications. Springer (1997)
4. Biswas, A., Bhowmick, P., Bhattacharya, B.B.: Construction of Isothetic Covers of a Digital Object: A Combinatorial Approach. Journal of Visual Communication and Image Representation 21, 295–310 (2010)
5. CarveWright, <http://www.carvewright.com>
6. Colak, S.B., Papaioannou, D.G., Hooft, G.W., van der Mark, M.B., Schomberg, H., Paasschens, J.C.J., Melissen, J.B.M., van Asten, N.A.A.J.: Tomographic image reconstruction from optical projections in light-diffusing media. Appl. Opt. 36, 180–213 (1997)
7. Costa, J.M., Venetsanopoulos, A.N., Treffer, M.: Digital Tomographic Filtering of Radiographs. IEEE Trans. Medical Imaging 2(2), 76–88 (1983)
8. Defrise, M.: A short reader’s guide to 3D tomographic reconstruction. Comput. Med. Imaging Graph. 25(2), 113–116 (2001)

9. <http://3.bp.blogspot.com>
10. <http://translate.google.co.in>
11. Karmakar, N., Biswas, A., Bhowmick, P., Bhattacharya, B.B.: Construction of 3D Orthogonal Cover of a Digital Object. In: Aggarwal, J.K., Barneva, R.P., Brimkov, V.E., Koroutchev, K.N., Korutcheva, E.R. (eds.) IWICIA 2011. LNCS, vol. 6636, pp. 70–83. Springer, Heidelberg (2011)
12. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Morgan Kaufmann, San Francisco (2004)
13. My CAD Site, <http://www.we-r-here.com>
14. Preparata, F.P., Shamos, M.I.: Computational Geometry—An Introduction. Springer, New York (1985)
15. The Official Google Sketchup Blog, <http://sketchupdate.blogspot.in>
16. Trussell, H.J.: Processing of X-ray images. Proc. IEEE 69(5), 615–627 (1981)
17. Ziedses des Plantes, B.G.: Body-section radiography: History, image information, various techniques and results. Australasian Radiology XV(1), 57–64 (1971)

Fast Combinatorial Algorithm for Tightly Separating Hyperplanes

Peter Veelaert

Ghent University,
Valentin Vaerwyckweg 1, B9000 Ghent, Belgium
peter.veelaert@ugent.be

Abstract. We propose a new algorithm for finding separating hyperplanes between two data sets with respect to the L_∞ norm. The algorithm is an adaptation of a previous result on enclosing hyperplanes. Our main result is that the existing algorithm for finding enclosures can also be applied to find separations provided the two data sets cannot be separated in a space of lower dimension.

Keywords: Separating hyperplanes, Enclosure of point sets, Arithmetical thickness.

1 Introduction

A recurring problem in discrete geometry is to determine the two hyperplanes that enclose a finite point set as tightly as possible. This problem is relevant both from a topological as an algebraic viewpoint. For example, on a regular grid of discrete points a discrete analytical hyperplane consists of all the grid points between two parallel hyperplanes. The distance between the two parallel hyperplanes is also called the arithmetical thickness. When a discrete hyperplane S has the right arithmetical thickness, then S will be tunnel free (it separates space into distinct parts, and one cannot go from one part to another part without crossing the plane) and minimal (no points can be discarded without disturbing the topology of the plane and the surrounding space) [1,3,7].

In this paper we consider the related question of separating two finite point sets by two parallel hyperplanes which are as far away as possible from each other. In this sense the separation is also tight. We propose a fast combinatorial algorithm for determining these separating hyperplanes and we prove its correctness.

A fast algorithm for finding separating hyperplanes is also relevant in machine learning where one has to separate data classes by decision surfaces. In fact, the plane that lies in the middle between the tightly separating hyperplanes is the same as a support vector machine would find, if it uses the L_∞ norm, and hard bounds to separate two classes [4].

In Section 2 we examine the relation between enclosure and separation problems. Section 3 introduces a fast algorithm for finding tight enclosures. Section 4 contains the main results of the paper, and it shows how the enclosure algorithm can be adapted to find tight separations. Section 5 briefly addresses time complexity issues.

2 Separation and Enclosure

Let $f_a(p) : \mathbb{R}^n \rightarrow \mathbb{R}$ denote a real function of the form

$$f_a(p) = x_n - (a_0 + a_1x_1 + \cdots + a_{n-1}x_{n-1}),$$

where the a_i represent coefficients and $p = (x_1, \dots, x_n)$ is a point of \mathbb{R}^n . The equation $f_a(p) = 0$ defines a hyperplane in \mathbb{R}^n with coefficients $a = (a_0, \dots, a_{n-1})$.

In digital geometry one often considers the following enclosure problem. Let S be a finite subset of points p_i in \mathbb{R}^n . Let $\epsilon(S)$ denote the minimal value for ϵ for which the system

$$\epsilon \geq x_n - (a_0 + \cdots + a_{n-1}x_{n-1}) \geq -\epsilon, p \in S \quad (1)$$

is still feasible. Clearly, this is a linear programming problem in which we minimize ϵ , while the a_i and ϵ have to satisfy linear inequalities. If $a_0, \dots, a_{n-1}, \epsilon(S)$ represents a solution of (1), then the points of S are tightly enclosed by the hyperplanes $f_a(p) = \epsilon(S)$ and $f_a(p) = -\epsilon(S)$.

The problem that we want to consider, however, is a separation problem, stated for two sets S^+ and S^- . We define $\delta(S^+, S^-)$ as the maximal value of δ for which the system

$$\begin{aligned} x_n - (a_0 + \cdots + a_{n-1}x_{n-1}) &\geq \delta, p \in S^+ \\ x_n - (a_0 + \cdots + a_{n-1}x_{n-1}) &\leq -\delta, p \in S^- \end{aligned} \quad (2)$$

is still feasible. We want to find $\delta(S^+, S^-)$ as well as the parameters a_i . Clearly, this is again a linear programming problem, which we can easily reformulate as an adapted enclosure problem. To do so rewrite (2) as

$$\begin{aligned} (x_n - \tau) - (a_0 + \cdots + a_{n-1}x_{n-1}) &\geq \delta - \tau, p \in S^+ \\ (x_n + \tau) - (a_0 + \cdots + a_{n-1}x_{n-1}) &\leq -\delta + \tau, p \in S^- \end{aligned} \quad (3)$$

where τ is some real number. Now we let $\epsilon = -\delta + \tau$, and we derive from S^+ and S^- two new sets:

$$\begin{aligned} T_\tau^+ &= \{q : q = p + (-\tau, 0, \dots, 0), p \in S^+\} \\ T_\tau^- &= \{q : q = p + (\tau, 0, \dots, 0), p \in S^-\}, \end{aligned} \quad (4)$$

If we denote the coordinates of q as $q = (y_1, \dots, y_n)$, then finding the maximal value for δ in (3), is the same as finding the minimal value for ϵ for which

$$\begin{aligned} y_n - (a_0 + \cdots + a_{n-1}y_{n-1}) &\geq -\epsilon, q \in T_\tau^+ \\ y_n - (a_0 + \cdots + a_{n-1}y_{n-1}) &\leq \epsilon, q \in T_\tau^- \end{aligned} \quad (5)$$

is still feasible. We shall denote this maximal value as $\epsilon(T_\tau^+, T_\tau^-)$. Clearly,

$$\epsilon(T_\tau^+, T_\tau^-) = \tau - \delta(S^+, S^-). \quad (6)$$

Thus (5) takes the form of an enclosure problem in which we raised the points of S^- over a distance τ , while we lowered the points of S^+ over the same distance. The main difference between (1) and (5) is that (5) is no longer symmetrical. In (5) there are two different sets, T_τ^+ and T_τ^- that define the upper and lower bounds for ϵ . We will call this

problem a signed enclosure problem in which we have to compute $\epsilon(T_\tau^+, T_\tau^-)$, while (II) is called an unsigned enclosure problem where we compute $\epsilon(S)$, or $\epsilon(S^+ \cup S^-)$ for that matter.

Figure I illustrates the basic principle. The set S^+ consisting of light points is lowered, and the set S^- of dark points is raised. The tightly enclosing lines in Fig. I(b) yield tightly separating lines in Fig. I(c).

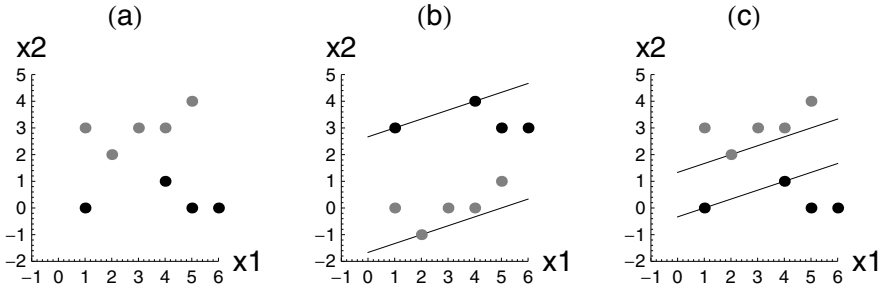


Fig. 1. A separation problem (a) is transformed into an enclosure problem (b) by raising the dark points, and lowering the light points. When the points return to their original positions, shown in (c), the tightly enclosing lines become tightly separating lines. For the moment the distance τ over which the points are displaced is not relevant, but later on, the distance will matter when we replace a signed enclosure problem by an unsigned enclosure problem.

For the moment the value of τ is not important, since (6) holds for any τ . Later we will add the condition that τ must be sufficiently large to make sure that the enclosure algorithm, which will be proposed later, computes the correct value for $\epsilon(T_\tau^+, T_\tau^-)$.

3 A Fast Algorithm for Finding Tight Enclosures

There is a fast algorithm for finding tight unsigned enclosures, which is based on elemental subsets [12]. Elemental is a term borrowed from robust regression [6, 10]. An elemental set is a subset of the data containing the minimum number of points needed to identify the parameters of the model. Here we shall add the constraint that the points are in general position so that we can uniquely determine these parameters. For our purposes, a set of n points p_i is in general position if there is a unique hyperplane $f_a(p) = 0$ passing through the points. For example, for planes in \mathbb{R}^3 , there is a unique plane passing through 3 distinct points, provided the points are not collinear.

Definition 1. Let S be a finite set of points in \mathbb{R}^n . An elemental subset E is a subset of S with $n + 1$ points, which has at least one n -point subset with its points in general position.

The primary importance of elemental subsets stems from the fact that when S is equal to an elemental subset E , the minimal value for $\epsilon(S)$, subject to the inequalities (II), can

be found in an analytical way. For $n + 1$ points $p_i = (x_{i1}, \dots, x_{in})$, $1 \leq i \leq (n + 1)$ we define the $(n + 1) \times (n + 1)$ matrix

$$M := \begin{pmatrix} 1 & x_{11} & \dots & x_{1n} \\ \dots & \dots & \dots & \dots \\ 1 & x_{(n+1)1} & \dots & x_{(n+1)n} \end{pmatrix}. \quad (7)$$

Let C_i , $1 \leq i \leq n + 1$ denote the cofactors of the last column of M . These cofactors play an important role with respect to the relative positions of the points in E and the enclosing planes $f_a(p) = \pm\epsilon$.

We start with a simple observation, which follows immediately from linear algebra.

Lemma 1. *Let E be a subset of $n + 1$ points p_i . Then E is an elemental subset if and only if at least one of the cofactors C_i is non vanishing.*

Proof. It suffices to show that if some cofactor $C_i \neq 0$, then there is always a unique plane passing through an n point subset of E . Without loss of generality we may assume $C_1 \neq 0$. Consider the system of linear equations

$$x_{in} = a_0 + a_1x_{i1} + \dots + a_{n-1}x_{i(n-1)}, \quad i = 2, \dots, n + 1.$$

If there is at least one $x_{in} \neq 0$ for $i = 2, \dots, n + 1$, this is a non-homogeneous system, which has a unique solution since the determinant of its coefficient matrix is non-zero. If all $x_{in} = 0$ for $i = 2, \dots, n + 1$, this is a homogeneous system with the unique solution $(a_0, \dots, a_{n-1}) = (0, \dots, 0)$. In both cases there is a unique hyperplane $x_n = a_0 + a_1x_1 + \dots + a_{n-1}x_{n-1}$ passing through the points p_i , with $i = 2, \dots, n + 1$. \square

For elemental subsets we introduce a residual $\epsilon(E)$ that weighs the absolute value of the determinant of M against the sum of the absolute values of the cofactors,

$$\begin{aligned} \epsilon(E) &:= |\det(M)| / (|C_1| + \dots + |C_{n+1}|) \\ &= |C_1p_{1n} + \dots + C_{n+1}p_{(n+1)n}| / (|C_1| + \dots + |C_{n+1}|). \end{aligned} \quad (8)$$

Since, by Lemma 1, for an elemental subset at least one of the cofactors is non-zero, the denominator in (8) is always non-zero, and $\epsilon(E)$ is therefore always defined.

To be able to describe a fast enclosure algorithm, we summarize three results proven in [11, 12].

Theorem 1. *Let E be an elemental subset for which all the n -point subsets are in general position. Then there is a unique pair of hyperplanes $f_a(p) = \epsilon(E)$, $f_a(p) = -\epsilon(E)$ such that each point p_i in E lies on the hyperplane*

$$f_a(p_i) = \text{sign}(C_i)\text{sign}(M)\epsilon(E).$$

Furthermore, there is no hyperplane $f_a(p) = 0$ for which $|f_a(p)| < \epsilon$ for all $p \in E$.

Hence, when S coincides with an elemental subset E , the residual $\epsilon(E)$ as defined in (8) coincides with the minimal value of ϵ in (refencleps), which we denoted as $\epsilon(S)$.

As noted in Lemma 1 the requirement that all n -point subsets are in general position, is equivalent to the condition that all cofactors C_i are non zero. Theorem 1 is no longer valid, however, when one or more of the cofactors are zero. To illustrate what happens, Figure 2 (a) and (b) shows a configuration where all points $p_i = (x_i, y_i)$ have the same x coordinate, except for p_5 . The elemental subset $E_{145} = \{p_1, p_4, p_5\}$ has matrix

$$M_{145} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_4 & y_4 \\ 1 & x_5 & y_5 \end{pmatrix}.$$

If we let C_1, C_4, C_5 denote the cofactors of the last column of M_{145} , then, since $x_1 = x_4 \neq x_5$, we have $C_1 \neq 0, C_4 \neq 0$, but $C_5 = 0$. As a result the enclosing hyperplanes $f_a(p) = \pm\epsilon(E)$ are not unique, although $\epsilon(E_{145})$ is well defined. Figures 2(a) and (b) show two possible enclosures where in each case the vertical height between the two supporting hyperplanes is equal to $\epsilon(E_{145})$. Since the sign of cofactor C_5 is undetermined, the point p_5 can either lie on the lowest supporting hyperplane, or on the highest supporting hyperplane.

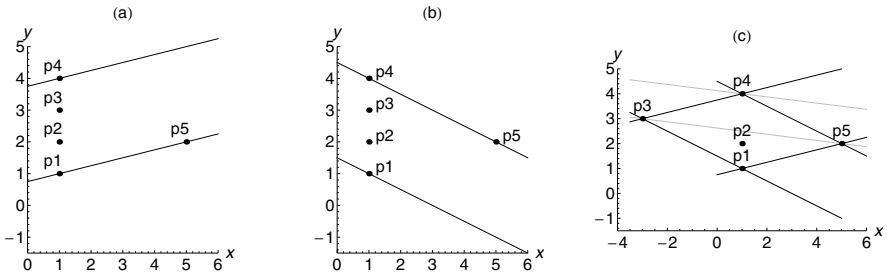


Fig. 2. Three special cases where some of the points have the same x -coordinate. As a result the enclosing (or supporting) hyperplanes are not unique.

Second, one can show that the unsigned enclosure of Theorem 1 is optimal in the sense that there is no signed enclosure that is tighter [11, 12].

Theorem 2. Let S^+, S^- be a partition of S and let E be an elemental subset in $S^+ \cup S^-$ for which all the n -point subsets are in general position. Then $\epsilon(E) \leq \epsilon(E^+, E^-)$.

The previous two results refer to elemental subsets, but can be extended to larger sets. Let S be a (large) finite subset of points that contains at least one elemental subset. The size of S may be much larger than $n + 1$. We define the residual of S as

$$\epsilon(S) := \max_{E \subseteq S} \epsilon(E),$$

where the maximum of $\epsilon(E)$ is taken over all elemental subsets E in S . The following result was proven in [11].

Theorem 3. *Let S be a finite set of points and let E be an elemental subset for which $\epsilon(E) = \epsilon(S)$. Let $f_a(p) = 0$ be a hyperplane such that $|f_a(p)| \leq \epsilon(E)$, for all $p \in E$. Then $|f_a(p)| \leq \epsilon(E)$, for all $p \in S$. Furthermore, for any $\epsilon < \epsilon(E)$ there is no hyperplane $f_a(p) = 0$ such that $|f_a(p)| \leq \epsilon$, for all $p \in S$.*

Whether the surface in Theorem 3 is unique depends on the cofactors of those elemental subsets E that yield the maximal residual $\epsilon(S)$. Figure 2(c) shows an example where $\epsilon(S) = \epsilon(E_{134}) = \epsilon(E_{145})$. However, both E_{345} and E_{135} have cofactors that are zero, and their enclosing hyperplanes are not unique. Therefore, the enclosing hyperplanes of S are also not unique. On the other hand, the elemental subset E_{345} has a unique pair of enclosing hyperplanes (shown as gray lines in Fig. 2(c)), but these hyperplanes do not enclose the entire set S , since $\epsilon(E_{345}) < \epsilon(S)$.

In the remainder, to simplify the exposition we will avoid these special cases, and we will assume that all the n point subsets of S are in general position. In practice, this can be accomplished by adding very small random offsets to the points.

Theorem 3 has an immediate corollary [12]. In fact, an elemental subset can only give rise to enclosing hyperplanes if its residual is maximal. Or conversely, if a point $p \in S$ is not enclosed by the supporting hyperplanes of some elemental subset E , then $E \cup \{p\}$ must have a residual that is larger than the residual of E (since otherwise the supporting hyperplanes of E would also enclose $E \cup \{p\}$).

Corollary 1. *Let S be a finite set that contains at least one elemental subset E_1 . Let $f_a(p) = \epsilon(E_1)$ and $f_a(p) = -\epsilon(E_1)$ denote the two hyperplanes that tightly enclose the points of E_1 . If there is a point $p_2 \in S$ for which $|f_a(p_2)| > \epsilon(E_1)$ then the set $E_1 \cup \{p_2\}$ contains at least one elemental subset E_2 for which $\epsilon(E_2) > \epsilon(E_1)$.*

Corollary 1 yields an efficient greedy algorithm for finding tight unsigned enclosures [12]. The key idea is to increase the residual of an elemental subset by replacing one of its points. Corollary 1 states that this is always possible as long as there remains at least one point that falls outside the enclosure. On the other hand, when all the points of S fall within the enclosure defined by E , we have found an optimal solution.

Enclosure algorithm.

Input: A finite set S .

Output: An elemental subset E_k such that $\epsilon(E_k) = \epsilon(S)$.

1. Select an arbitrary initial elemental subset E_1 and compute $\epsilon(E_1)$;
 2. Compute the best fit f_a of E_i ;
 3. Process all points of S until a point p is found at a distance further than $\epsilon(E_i)$ from the best fit;
 4. If no such point is found, return the current E_i ;
 5. Replace E_i by an elemental subset E_{i+1} in $E_i \cup \{p\}$ for which $\epsilon(E_{i+1}) > \epsilon(E_i)$. According to Corollary 1 there is at least one such subset in $E_i \cup \{p\}$.
 6. Proceed with step 2.
-

As explained in [12] the best fit f_a of step 2 can be computed by solving the system

$$f_a(p_j) = \text{sign}(C_j)\text{sign}(M)\epsilon, \quad p_j \in E_i,$$

which has $n + 1$ linear equations, and $n + 1$ unknowns $a_0, \dots, a_{n-1}, \epsilon$. When $\epsilon(E_i)$ is already known, ϵ can be replaced by $\epsilon(E_i)$, and one of the $n + 1$ equations can be discarded, no matter which equation.

4 Tight Separations

The above enclosure algorithm determines the unsigned enclosure and residual $\epsilon(S)$ of a finite set S . However, to solve a separation problem, we must either determine the maximal value of δ in (2), or after raising the points of S^- and lowering the points of S^+ , determine the minimal value of ϵ in (5). Both are signed problems, and a signed enclosure can differ from an unsigned enclosure. Figure 3 shows an example in which the dark points belong to the raising of S^- , and the light points belong to the lowering of S^+ . Figure 3(a) shows the unsigned tight enclosure. The supporting lines contain points of both types, and therefore they cannot be the supporting lines of a signed enclosure. Figure 3(b) shows the tightest signed enclosure, where the upper support line contains only dark points, and the lower support line only contains light points. This example shows that a tight unsigned enclosure, as produced by the enclosure algorithm, is not necessarily a solution for (5).

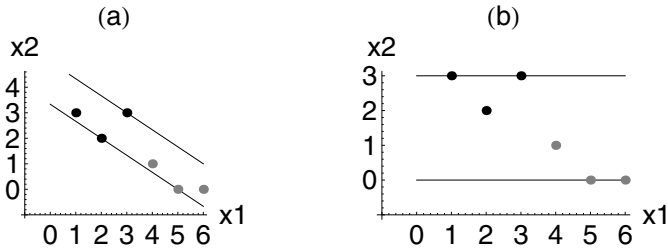


Fig. 3. The difference between (a) unsigned enclosure, and (b) signed enclosure. In the unsigned enclosure the supporting lines may contain points from both sets, while in the signed enclosure each line has to pass either through dark points or through light points. Clearly, in an unsigned enclosure the distance between the lines can be much smaller.

Figure 4 shows a second configuration of raised and lowered points. In this case the signed and unsigned enclosure coincide. Each supporting line only contains points from one part. Note that in Fig. 3 the points of S^+ and S^- are distributed differently than in Fig. 4. If we look at the projections $\pi(p)$ upon the x_1 -axis of the points in Fig. 3, then the two projected sets $\pi(S^+)$ and $\pi(S^-)$ can still be linearly separated, i.e, there is a real number α such that $\pi(p) < \alpha$ for all $p \in S^+$ and $\pi(p) > \alpha$ for all $p \in S^-$. In Fig. 4 the projected points cannot be linearly separated. It seems therefore that the algorithm for finding unsigned enclosures does not work for signed enclosures when the partition is biased, i.e, the projected points can already be separated linearly in a space of lower dimension. We will show that this is true in general. Furthermore, we will show that this is the only way in which the algorithm can fail.

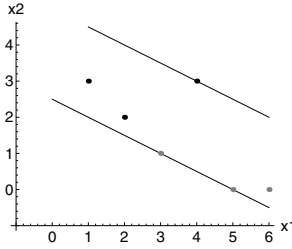


Fig. 4. When the projection on the x_1 axis produces two data sets that cannot be linearly separated (their convex hulls overlap), then the signed and unsigned enclosures coincide. In Theorem 4 we will show that this is true in general, provided τ is chosen large enough.

Proof of the Main Result

Given a partition S^+, S^- , for each elemental subset E in $S^+ \cup S^-$ we let E^+, E^- denote the partition of E induced by S^+, S^- . That is, $E^+ = E \cap S^+$ and $E^- = E \cap S^-$.

Definition 2. Let E be an elemental subset of S with all n point subsets in general position. We say that the signs of the cofactors coincide with the partition E^+, E^- if for any pair $p_i \in E^+$ and $p_j \in E^-$ we have $\text{sign}(C_i)\text{sign}(C_j) = -1$. In other words, if the points belong to distinct parts, their cofactors must have opposite signs.

We shall use $C^+(E)$ and $C^-(E)$ to denote the partition as induced by the cofactors, i.e $C^+(E) \cup C^-(E) = E$ and

$$C^+(E) = \{p_i : p_i \in E \text{ and } C_i > 0\},$$

$$C^-(E) = \{p_i : p_i \in E \text{ and } C_i < 0\}.$$

Recall that we reformulated the separation of two sets, as an enclosure of two sets where the points were raised or lowered by τ . The following lemma states that an elemental subset for which the signs of the cofactors do not coincide with the partition cannot be an enclosing subset when τ is sufficiently large.

Lemma 2. Let S^+ and S^- be finite subsets of \mathbb{R}^n , such that $S^+ \cup S^-$ contains at least one elemental subset for which the signs of the cofactors coincide with the partition. Let τ be a positive real number, and let T_τ^+ and T_τ^- denote the sets derived from S^+ and S^- as in (4). Let $T_\tau = T_\tau^+ \cup T_\tau^-$ and let E_τ^M be the elemental subset in T_τ with the largest unsigned residual, i.e.,

$$E_\tau^M = \text{argmax}_{E \subset T_\tau} \epsilon(E).$$

Then there exists a lower bound τ_0 such that for all $\tau \geq \tau_0$ we have $E_\tau^M \cap T_\tau^+ = C^+(E_\tau^M)$ and $E_\tau^M \cap T_\tau^- = C^-(E_\tau^M)$.

Proof. Let $E = \{p_1, \dots, p_{n+1}\}$ be an elemental subset of S . Let $I = \{1, \dots, n+1\}$ denote the set of indices of the points in E . We partition I into four different subsets:

$$\begin{aligned} I^{++} &= \{i : p_i \in E \cap S^+ \text{ and } C_i > 0\} \\ I^{--} &= \{i : p_i \in E \cap S^- \text{ and } C_i < 0\} \\ I^{+-} &= \{i : p_i \in E \cap S^+ \text{ and } C_i < 0\} \\ I^{-+} &= \{i : p_i \in E \cap S^- \text{ and } C_i > 0\} \end{aligned} \quad (9)$$

Then $I = I^{++} \cup I^{--} \cup I^{+-} \cup I^{-+}$, since $C_i \neq 0$ for all i . We will use C_I as a shorthand for $\sum_{i \in I} |C_i|$. The residual of E can then be written as

$$\epsilon(E) = \left(\left| \sum_{i \in I^{++}} x_{in} C_i + \sum_{i \in I^{--}} x_{in} C_i + \sum_{i \in I^{+-}} x_{in} C_i + \sum_{i \in I^{-+}} x_{in} C_i \right| \right) / C_I. \quad (10)$$

There are only two ways in which the signs of the cofactors can coincide with the partition, either $I^{+-} = I^{-+} = \emptyset$ or $I^{++} = I^{--} = \emptyset$. First assume that $I^{+-} = I^{-+} = \emptyset$. Then (10) is equal to

$$\epsilon(E) = \left(\left| \sum_{i \in I^{++}} x_{in} C_i + \sum_{i \in I^{--}} x_{in} C_i \right| \right) / C_I.$$

When we raise the points of S^- by τ , while we lower the points of S^+ also by τ the residual will change:

$$\epsilon(E_\tau) = \left(\left| \sum_{i \in I^{++}} (x_{in} - \tau) C_i + \sum_{i \in I^{--}} (x_{in} + \tau) C_i \right| \right) / C_I \quad (11)$$

where E_τ contains the raised and lowered points of E . Since $C_i \geq 0$ for $i \in I^{++}$ and $C_i \leq 0$ for $i \in I^{--}$, we obtain

$$\epsilon(E_\tau) = \left(\left| \sum_{i \in I} x_{in} C_i - \tau \sum_{i \in I} |C_i| \right| \right) / C_I.$$

Hence, for τ sufficiently large, i.e. $\tau > |\sum_{i \in I} x_{in} C_i| / C_I$, $\epsilon(E_\tau)$ increases as

$$\epsilon(E_\tau) = \tau - \epsilon(E).$$

The case $I^{++} = I^{--} = \emptyset$ leads to a similar result, where $\epsilon(E_\tau)$ increases at the same rate as τ .

Now let $F = \{q_1, \dots, q_{n+1}\}$ be a second elemental subset for which the signs of the cofactors do not coincide with the partition. That is, $I^{-+} \cup I^{+-} \neq \emptyset$ and $I^{++} \cup I^{--} \neq \emptyset$. We denote the coordinates of the points of F as $q_i = (y_{i1}, \dots, y_{in})$. We again partition the index set as in (9). If we let F_τ denote the set of lowered and raised points of F , we have

$$\begin{aligned} \epsilon(F_\tau) = & \left(\left| \sum_{i \in I^{++}} (y_{in} - \tau) C_i + \sum_{i \in I^{--}} (y_{in} + \tau) C_i + \right. \right. \\ & \left. \left. \sum_{i \in I^{+-}} (y_{in} - \tau) C_i + \sum_{i \in I^{-+}} (y_{in} + \tau) C_i \right| \right) / C_I. \end{aligned}$$

Since, $C_i \geq 0$ for $i \in I^{++} \cup I^{-+}$ and $C_i \leq 0$ for $i \in I^{--} \cup I^{+-}$ we obtain

$$\epsilon(F_\tau) = \left(\left| \sum_{i \in I} y_{in} C_i - \tau \left(\sum_{i \in I^{++} \cup I^{--}} |C_i| - \sum_{i \in I^{+-} \cup I^{-+}} |C_i| \right) \right| \right) / C_I. \quad (12)$$

Because in this case $I^{-+} \cup I^{+-} \neq \emptyset$ and $I^{++} \cup I^{--} \neq \emptyset$,

$$0 \leq \left| \sum_{i \in I^{++} \cup I^{--}} |C_i| - \sum_{i \in I^{+-} \cup I^{-+}} |C_i| \right| / C_I < 1. \quad (13)$$

Hence, for τ sufficiently large, $\epsilon(F_\tau)$ increases as

$$\epsilon(F_\tau) = c_0 + c_1 \tau,$$

where c_0, c_1 are constants and $0 \leq c_1 < 1$. Thus, when τ is sufficiently large, $\epsilon(E_\tau)$ will always be larger than $\epsilon(F_\tau)$. Hence if $E_\tau^M = \operatorname{argmax}_{E \subset T_\tau} \epsilon(E)$ the cofactor signs of E_τ^M must coincide with the partition when τ is sufficiently large. \square

We now derive explicit conditions that ensure that S^+, S^- contains an elemental subset for which the signs of the cofactors coincide with the partition. First we note that the cofactors provide linear dependencies between the points of E .

Lemma 3. *Let C_i be the cofactors of the last column of the matrix M . Then*

$$\sum_{1 \leq i \leq n+1} \pi(p_i) C_i = 0$$

and

$$\sum_{1 \leq i \leq n+1} C_i = 0$$

Proof. To prove the first part it suffices to replace the last column of M by the elements x_{ij} for some $j = 1, \dots, n-1$. Since the determinant of a matrix with two identical columns is zero, $\sum_{1 \leq i \leq n+1} x_{ij} C_i = 0$ for each $j = 1, \dots, n-1$. Hence $\sum_{1 \leq i \leq n+1} \pi(p_i) C_i = 0$. Second, if we replace the elements of the last column of M all by one, we again obtain a matrix with two identical columns, and therefore $\sum_{1 \leq i \leq n+1} C_i = 0$. \square

We now derive a simple condition for a set to contain at least one elemental subset that will trigger the application of Lemma 2. The proof is based in Caratheodory's theorem [13].

Lemma 4. *Let S^+, S^- be finite subsets of \mathbb{R}^n such that*

$$(\pi(S^+) \cap \operatorname{conv}\pi(S^-)) \cup (\pi(S^-) \cap \operatorname{conv}\pi(S^+)) \neq \emptyset$$

Then $S^+ \cup S^-$ contains at least one elemental subset E for which the signs of the cofactors coincide with the partition.

Proof. Without loss of generality we may assume that $\pi(S^+) \cap \text{conv}\pi(S^-) \neq \emptyset$, and that there is a point $p_1 \in S^+$ such that $\pi(p_1) \in \text{conv}\pi(S^-)$.

The convex set $\text{conv}\pi(S^-)$ lies in an $(n - 1)$ -dimensional space. Since $\pi(p_1) \in \text{conv}\pi(S^-)$ by Caratheodory's theorem there must be n points p_2, \dots, p_{n+1} in S^- such that $\pi(p_1)$ is also within the convex hull of the n projections of these points. Let E denote the elemental subset defined by $E = \{p_1, \dots, p_{n+1}\}$. By Lemma 3 we have

$$p_1 = -\frac{1}{C_1} \sum_{2 \leq i \leq n+1} \pi(p_i)C_i \quad (14)$$

Also by Lemma 3 $-\sum_{2 \leq i \leq n+1} C_i = C_1$. It follows that (14) represents p_1 as an affine span of p_2, \dots, p_{n+1} , that is, $p_1 = \alpha_2 p_2 + \dots + \alpha_{n+1} p_{n+1}$, where $\alpha_2 + \dots + \alpha_{n+1} = 1$. Furthermore, since p_1 lies in the convex hull of the points $\pi(p_i)$ we also have $\alpha_i \geq 0$. Hence $\text{sign}(C_i)\text{sign}(C_1) = -1$, for $2 \leq i \leq n + 1$, which means the cofactor signs coincide with the partition of the points into S^+ and S^- . \square

Note that the previous lemma formalizes our observation made in Figs. 3 and 4. In fact, the main theorem follows immediately from the previous results.

Theorem 4. *Let S^+, S^- be finite subsets of \mathbb{R}^n such that*

$$(\pi(S^+) \cap \text{conv}\pi(S^-)) \cup (\pi(S^+) \cap \text{conv}\pi(S^-)) \neq \emptyset$$

and let T_τ denote the set of raised and lowered points of S . Then there is a lower bound τ_0 such that for all $\tau \geq \tau_0$

$$\max_{E \subset T_\tau} \epsilon(E) = \epsilon(T_\tau) = \epsilon(T_\tau^+, T_\tau^-) = \tau - \delta(S^+, S^-)$$

Proof. By Lemma 4 there is at least one elemental subset E in S for which the partition of S coincides with the signs of the cofactors. Hence there is an elemental subset in T_τ that has the largest residual, which we will denote as E_τ^M , i.e. $\epsilon(E_\tau^M) = \max_{E \subset T_\tau} \epsilon(E)$. Then by Theorem 1 $\epsilon(E_\tau^M) = \epsilon(T_\tau)$. By Lemma 2 when τ is sufficiently large, we have $\epsilon(E_\tau^M) = \epsilon(E_\tau^M \cap T_\tau^+, E_\tau^M \cap T_\tau^-)$. By Theorem 1 the supporting hyperplanes of E_τ^M enclose T_τ , while for any elemental subset $F_\tau \subset T_\tau$ for which $\epsilon(F_\tau^+, F_\tau^-) < \epsilon(E_\tau^M)$, the enclosing hyperplanes of F cannot enclose the entire set T_τ . Hence, we also have $\epsilon(E_\tau^M) = \epsilon(T_\tau^+, T_\tau^-)$. \square

5 Application and Time Complexity

If the conditions of Theorem 4 are satisfied, we can compute the separating hyperplanes with the enclosure algorithm. The conditions require that the projection of the two data sets overlap, in the sense that the projection of one of the convex hulls must contain at least one projected point of the other data set.

This is a common situation. Figure 5 shows a typical example where the projected sets clearly overlap. The separating hyperplanes were found by (i) lowering and raising the data points, (ii) finding the enclosing hyperplanes, and (iii) raising one and lowering the other hyperplane so that they separate (and support) the original two sets.

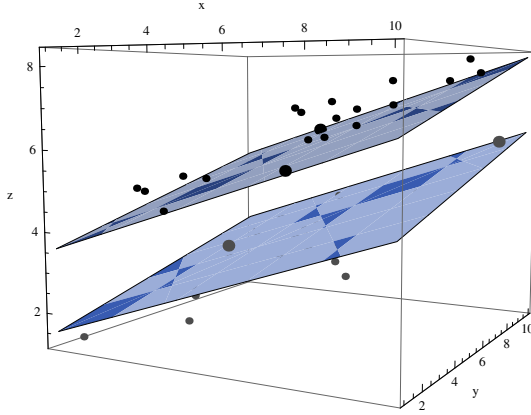


Fig. 5. The two supporting planes separate the positive points (dark) from the negative points (light). The support points are indicated as large points, two on each supporting plane.

The conditions required by Theorem 4 are violated when the two projected hulls do not overlap. In fact, this means that the projected sets can already be separated linearly in \mathbb{R}^{n-1} . In this sense, Theorem 4 admits that when the sets can already be separated in \mathbb{R}^{n-1} , the separation algorithm can give incorrect results in \mathbb{R}^n . In other words, the data sets should be separated in the subspace of lowest possible dimension in which the projected data points are already linearly separable.

To conclude, we briefly address the time complexity of the enclosure/separation algorithm. Suppose the data set consists of N lowered and raised points. In \mathbb{R}^n , this means that there are $O(M)$ elemental subsets, where $M = \binom{N}{n+1}$. Since the goal of the proposed algorithm is to find the elemental subset with the largest residual, the time complexity depends on the speed with which the elemental cost will increase when replacing points in the elemental subset by more distant points. Let $\epsilon_1, \epsilon_2, \dots, \epsilon_M$ represent the sequence of the residuals in increasing order of all the elemental subsets. If we select an elemental subset at random, then its residual will be one element of this sequence, denoted ϵ_j . Hence, if the algorithm replaces the elemental subset with a new one that has a larger residual, then a desirable property would be that the new residual would lie close to the median of ϵ_j and ϵ_M . If this would indeed be the case the algorithm would converge to the maximal residual in $\log_2 M = \log_2 \binom{N}{n+1}$ steps. Since for each elemental subset we have to evaluate, in the worst case all N points to find a new point behind the enclosing hyperplanes of the current elemental subset, it may take up to N operations to process one elemental subset. This yields a time complexity of $O(N \log_2 N^{n+1}) = O((n+1)N \log_2 N)$ operations.

The above time complexity analysis, however, depends heavily on the assumption that each time we replace one of the points of an elemental subset, the new residual will lie close to the median of the remaining subsequence $\epsilon_j, \dots, \epsilon_M$. Up to now we have no real rigorous proof that this is in fact the case. However, experiments with random data sets have indeed shown that the time number of iterations is on average equal to $k \log_2 N$, where k does not increase faster than $n+1$. Figure 6 shows the number of

iterations that are needed to find the enclosing elemental subset in \mathbb{R}^4 and \mathbb{R}^6 , for data sets similar to the one shown in Fig. 5 for varying sizes of the data set.

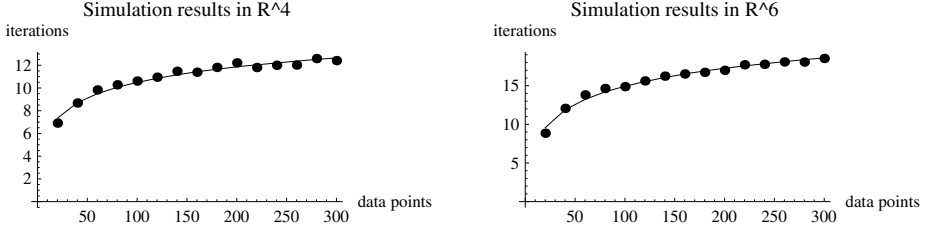


Fig. 6. Time complexity for varying size of the data set. From the above analysis, the expected number of iteration grows as $(n + 1) \log N$. The experimental results show the average number of iterations that were needed to find the enclosing elemental subset. The average was taken over 100 experiments for each $N = 10, 20, \dots$. The smooth curves show plots of the best fit of the form $a + b \log_2(N)$ to the experimental results, where N represents the number of data points. In \mathbb{R}^4 a good fit was $a + 1.35 \log_2(N)$ (shown left), in \mathbb{R}^6 a good fit was $a + 2.31 \log_2(N)$. In each case, the rate b is smaller than the expected value $n + 1$.

For support vector machines (SVM), approximative training algorithms have been reported that can find a separating hyperplane in linear time [8,9]. However, the linear time complexity of these iterative algorithms depends strongly on the parameter settings, such as the required accuracy of the approximation, the sparseness of the feature vectors, and the soft margin parameter [2,9]. In contrast, the combinatorial algorithm proposed in this paper is not an approximative method, but gives an exact result. Furthermore, the algorithm solves the primal optimization problem, which for large scale problems may have advantages over algorithms that solve the dual optimization problem, as discussed in [5].

Although the proposed method provides an exact solution, its major limitation is that it only provides separating hyperplanes for hard margins, not for soft margins [4]. Furthermore, as can be easily seen from the lowering and raising of the data points, when the two data sets are not linearly separable, the proposed algorithm will return a hyperplane for which the maximum distance between the misclassified data points and the hyperplane will be minimal.

6 Conclusion

We proposed a combinatorial algorithm for finding separating hyperplanes based on elemental subsets. However, some issues still have to be further explored to determine the method's qualities for use in practice. One important, but interesting problem is how the linear separability of the data set is affected when we introduce a lifting function to find a separating surface that is not linear. Since the proposed algorithm only works

correctly in the subspace of lowest dimension in which the data is separable, an interesting question is to find a lifting function for which the data set becomes just separable. Although the latter is one of the key problems in pattern recognition, a combinatorial approach may provide new insights.

References

1. Andres, E., Acharya, R., Sibata, C.: Discrete analytical hyperplanes. *Graphical Models and Image Processing* 59(5), 302–309 (1997)
2. Bialon, P.: A linear support vector machine solver for a large number of training examples. *Control and Cybernetics* 38(1), 281–300 (2009)
3. Brimkov, V.E., Coeurjolly, D., Klette, R.: Digital planarity – a review. *Discrete Applied Mathematics* 155(4), 468–495 (2007)
4. Burges, Ch.J.C.: A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* 2(2), 121–167 (1998)
5. Chapelle, O.: Training a support vector machine in the primal. *Neural Computation* 19(5), 1155–1178 (2007)
6. Hawkins, D.M., Bradu, D., Kass, G.V.: Location of several outliers in multiple regression data using elemental sets. *Technometrics* 26, 197–208 (1984)
7. Jamet, D., Toutant, J.-L.: Minimal arithmetic thickness connecting discrete planes. *Discrete Applied Mathematics* 157(3), 500–509 (2009)
8. Joachims, T.: Training linear svms in linear time. In: Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.) *Proc. Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226. ACM (2006)
9. Keerthi, S.S., Chapelle, O., DeCoste, D.: Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research* 7, 1493–1515 (2006)
10. Rousseeuw, P., Leroy, A.: *Robust Regression and Outlier Detection*. Wiley, New York (1987)
11. Veelaert, P.: Constructive fitting and extraction of geometric primitives. *Graphical Models and Image Processing* 59(4), 233–251 (1997)
12. Veelaert, P.: Separability and tight enclosure of point sets. In: Brimkov, V.E., Barneva, R.P. (eds.) *Digital Geometry Algorithms. Lecture Notes in Computational Vision and Biomechanics*, vol. 2, pp. 215–243. Springer (2012)
13. Ziegler, G.M.: *Lectures on Polytopes*. Graduate Texts in Mathematics, vol. 152. Springer, New York (1995)

Digital Curvatures Applied to 3D Object Analysis and Recognition: A Case Study

Li Chen¹ and Soma Biswas²

¹ University of the District of Columbia

lchen@udc.edu

² University of Maryland

somabapi@gmail.com

Abstract. This paper uses digital curvatures for 3D object analysis and recognition. For direct adjacency in 3D, digital surface points have only six types. It is easy to determine and classify the digital curvatures of each point on the boundary of a 3D object. This is simpler than the case of triangulation on the boundary surface of a solid; the curvature can be any real value. This paper focuses on the global properties of categorizing curvatures for small regions. We use both digital Gaussian curvatures and digital mean curvatures to characterize 3D shapes. Then propose a multi-scale method and a feature vector method for 3D similarity measurement. We found that Gaussian curvatures mainly describe the global features and average characteristics such as the five regions of a human face. However, mean curvatures can be used to find local features and extreme points such as nose in 3D facial data.

Keywords: Digital space, Digital Gaussian curvature, Digital mean curvature, Multi-scale, Feature vector, Classification.

1 Introduction

There are many applications related to geometric analysis in 3D image processing. Besl and Jain first proposed using Gaussian curvature and Mean curvature to classify 3D surface points [1,2]. Besl and Jain used the signs (positive, zero, negative) of Gaussian and mean curvatures to classify 3D surface points. Some related research can also be found in [3].

A 3D object can be represented by one or several closed surfaces (2D-manifolds). Curvatures that describe the degree of change at a point on the surface have been used for many years in 3D image processing [1,14]. The typical technology related to curvatures is the following: (1) triangulation of the surface, (2) fit the digital image to a continuous surface (using B-spline), and (3) calculate the standard Gaussian and/or mean Curvatures.

Worring and Smeulders showed that even in 2D, the interpretations of digitized curves may generate completely different curvatures [18]. In other words, the same image will generate different curvature maps if the triangulation of the is different.

On the other hand, methods in digital and discrete geometry for 3D computer graphics and 3D image processing have been developing in recent years. For example, recognition algorithms related to 3D medical images can be found in [4] in which Brimkov and Klette made extensive investigations in boundary tracking.

This paper presents a preliminary study of using digital curvatures of digital surfaces as features to classify different 3D objects. In this paper, we propose using curvatures in digital space for 3D object matching, classification, and recognition. Since direct adjacency only has six types of digital surface points in local configurations, it is much easier to determine and classify the digital curvatures for every point on the boundary surface of a 3D object.

2 Definitions of Digital Curvatures

In 1986, Besl and Jain presented a systematic method to use curvatures in image processing [2]. Besl and Jain used the signs (positive, zero, negative) of Gaussian curvature and Mean curvature to classify 3D surface points. Their technique uses triangulation for decomposition and then calculates the Gaussian curvature (K) and Mean curvature (H). Eight principle shapes can be identified: (1) Peak (surface) point $K > 0, H < 0$; (2) Flat point $H = 0, K = 0$; (3) Pit Point $H > 0, K > 0$; (4) Minimal point $H = 0, K < 0$; (5) Ridge point $K = 0, H < 0$; (6) Saddle Ridge point $H < 0, K < 0$; (7) Valley Point $H > 0, K = 0$; (8) Saddle Valley point $H > 0, K < 0$;

In fact, cases (1) and (3) are mirror images of the same shape and H changes the sign. This also applies to cases (5) and (7) and cases (6) and (8).

In this paper, we mainly study the curvature in digital space. Given a set of cloud points in 3D, we assume that they are connected. Since cubical space with direct adjacency, or (6,26)-connectivity space, has the simplest topology in 3D digital spaces, we will use it as the 3D image domain. It is also believed to be sufficient for the topological property extraction of digital objects in 3D. In this space, two points are said to be adjacent in (6,26)-connectivity space if the Euclidean distance between these two points is 1.

Let M be a closed (orientable) digital surface in the 3D grid space in direct adjacency. We know that there are exactly 6-types of digital surface points [5][6].

Assume that M_i (M_3, M_4, M_5, M_6) is the set of digital points with i neighbors. We have the following result for a simply connected M [5]:

$$|M_3| = 8 + |M_5| + 2|M_6|. \quad (1)$$

Mapping the digital surface points (Fig. 1) to the Besl-Jain classification, we will be able to see in the following that M_3 corresponds to cases (1) and (3), $M_4(a)$ to case (2), $M_4(b)$ to cases (5) and (7), M_5 to cases (6) and (8), and M_6 to case (4).

Digital configurations contain two types of minimal surface points. The correspondence between the shapes in the Besl-Jain paper and Fig. 1 is very interesting.

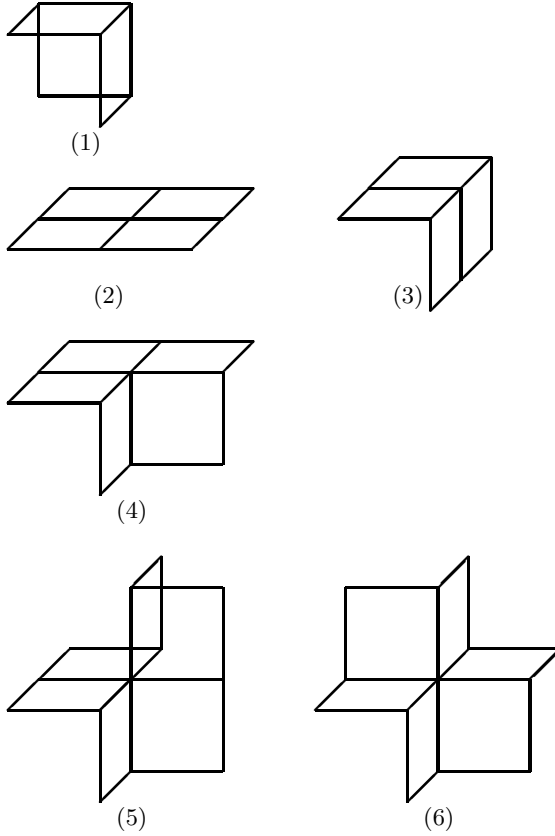


Fig. 1. Six types of digital surfaces points in 3D

Worring and Smeulders showed that even in 2D the interpretations of digitized curves may generate totally different curvatures [18]. In other words, the same image will generate different curvature maps if the triangulation is different.

2.1 Gaussian Curvatures and Mean Curvatures

If K_i denotes the Gaussian curvature of the elements in M_i , $i = 3,4,5,6$. We have (see discussion in [6].)

Lemma 1. (a) $K_3 = \pi/2$, (b) $K_4 = 0$, for both types of digital surface points, (c) $K_5 = -\pi/2$, and (d) $K_6 = -\pi$, for both types of digital surface points.

We also have a genus formula based on the Gauss-Bonnet Theorem [6]

$$g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8. \tag{2}$$

If H_i denotes the mean curvature of elements in M_i , $i = 3, 4, 5, 6$. We have

Lemma 2. (a) $H_3 = \frac{4}{\sqrt{3}}$. (b) $H_4 = 0$ for flat neighborhood, and $H_4 = \sqrt{2}$ for a bended neighborhood. (c) $H_5 = \frac{4}{5}$. (d) $H_6 = 0$, for both types of digital surface points.

This lemma can be proved easily based on the formula derived by Meyer et al in 2002 [13]. Goodman-Strauss and Sullivan used the $H_6 = 0$ points to construct cubic minimum surfaces [11]. A minimum surface can be defined as a surface whose mean curvature at every point is 0. There are several algorithms for obtaining minimum surfaces. Here we give a brief proof for Lemma 2. The formula derived in [13] for the mean curvature normal at point x is:

$$H(x) \cdot \mathbf{n} = \frac{1}{2 \cdot A} \sum_{x_i \in N(x)} (\cot \alpha_i + \cot \beta_i)(x - x_i) \tag{3}$$

where N is the set of (discrete) neighbors of x . α_i and β_i are angles in two different triangles that are both opposite to line segment xx_i ; which is shared by the triangles. A is called the voronoi region of x . For digital space, the voronoi region is easy to determine. See Fig 2.

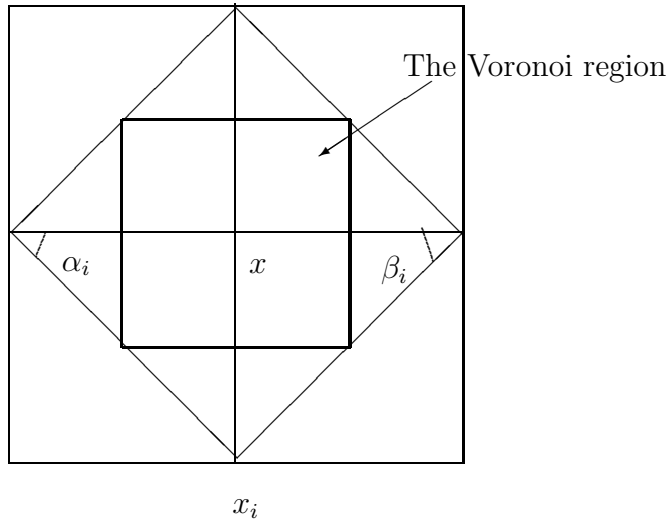


Fig. 2. The voronoi region in digital space

$$H(x) \cdot \mathbf{n} = \frac{1}{A} \sum_{x_i \in N(x)} (x - x_i) \tag{4}$$

Note that $H(x) \cdot \mathbf{n}$ is a vector. $|(x - x_i)| = 1$ and $\cot \alpha_i = \cot \beta_i = 1$. in Fig 2. $A = \frac{1}{4} \cdot i$ for M_i points. $|\sum_{x_i \in N(x)} (x - x_i)| = \sqrt{3}$ for M_3 ; $|\sum_{x_i \in N(x)} (x - x_i)| = 0$ for M_4 flat, and $\sqrt{2}$ for M_4 bended; $|\sum_{x_i \in N(x)} (x - x_i)| = 1$ for

M_5 ; $|\Sigma x_i \in N(x)(x - x_i)| = 0$ for M_6 ; Therefore, $H_3 = \frac{1}{3 \cdot \frac{1}{4}} \sqrt{3} = \frac{4}{\sqrt{3}}$. For bended M_4 points (Fig. 1 (c)), $H_4 = \frac{1}{4 \cdot \frac{1}{4}} \sqrt{2} = \sqrt{2}$. $H_5 = \frac{1}{5 \cdot \frac{1}{4}} \cdot 1 = \frac{4}{5}$.

2.2 Digital Principal Curvatures

The principal curvatures at a point p of a surface, denoted k_1 and k_2 , are the maximum and minimum values of the curve curvature on normal planes (intersecting with the surface). The relationship among the principal curvature, the Gaussian curvature, and the mean curvature are: $K = k_1 \cdot k_2$ and $H = (k_1 + k_2)/2$. Therefore [12] [13],

$$k_1 = H + \sqrt{H^2 - K}, k_2 = H - \sqrt{H^2 - K}$$

It is easy to get,

Lemma 3. (a) $k_1^{(3)} = \frac{4}{\sqrt{3}} + \sqrt{\frac{16}{3} - \frac{\pi}{2}} = 4.24913$, $k_2^{(3)} = \frac{4}{\sqrt{3}} - \sqrt{\frac{16}{3} - \frac{\pi}{2}} = 0.369675$. (b) $k_1^{(4b)} = 0$, $k_2^{(4b)} = 0$; $k_1^{(4c)} = 2\sqrt{2} = 2.82843$, $k_2^{(4c)} = 0$. (c) $k_1^{(5)} = \frac{4}{5} + \sqrt{\frac{16}{25} + \frac{\pi}{2}} = 2.28687$, $k_2^{(5)} = \frac{4}{5} - \sqrt{\frac{16}{25} + \frac{\pi}{2}} = -0.686875$. (d) $k_1^{(6)} = \sqrt{\pi} = 1.77245$, $k_2^{(6)} = -\sqrt{\pi} = -1.77245$ for both types of M_6 digital surface points.

From these lemmas, we can find some interesting information about digital principal curvatures. $k_1^{(4c)}$ does not have π involved in the formula. However, from Fig. 1c to Fig. 1a is just like making a 90 degree angle similar to the case from Fig. 1b to Fig. 1c. Where does the π came from? In classical differential geometry, the base area is usually a circle. But here we always consider the square. However, due to the definition of curvatures (or the intuitive meaning of curvatures), it is rounded. That may explain the digital curvatures. Sometime, π appears, and sometimes, it does not. Due to the fact that digitaldiscrete mean curvatures were obtained using an approximation, we think that the value of the mean curvatures for M_i points can be modified as $H' = c_1 \cdot H + c_2$ as long as we keep $H'_3 > H'_{4c} > H'_5 > H_6 = 0$, where c_1 and c_2 are two constants.

3 Digital Curvatures and 3D Image Analysis

Researchers have applied curvatures to image segmentation, shape recognition, and edge detection [15][8]. In this section, we investigate the digital Gaussian curvature and the digital mean curvature in classification of 3D objects. For instance in a closed digital surface, M_4 is independent to M_3 ; M_4 contains parabolic points (the bended ones). M_3 contains the elliptical points since the surface point is locally convex. M_5 and M_6 contain hyperbolic points; the Gaussian curvature is negative and the surface point will be locally saddle shaped.

3.1 Analysis Based on Digital Gaussian Curvatures

Let S be a subset of the 2D manifold M . We define a feature vector called the digital curvature vector $f_S = (m_3, m_4, m_5, m_6)$, where $m_i = |M_i|$ with respect to S . We can also split the vector into a six component vector to include two cases for M_4 and M_6 respectively. Basically, the vector now contains four components: $f_S = (|M_3|, |M_4|, |M_5|, |M_6|)$.

The motivation to define such a vector for a local region is based on the corresponding Gaussian-Bonnet theorem [12]. Suppose C is a boundary curve of S that is simply connected and k_g is the geodesic curvature of C , then

$$\int_C k_g dt + \int \int_S K dA = 2\pi \quad (5)$$

If C is a n -polygon and α_i is the interior angle, then

$$(n - 2) \cdot \pi + \int_C k_g dt + \int \int_S K dA = \sum_{i=1}^n \alpha_i \quad (6)$$

Therefore,

$$g_S = g(f_S) = |M_3| \cdot K_3 + |M_5| \cdot K_5 + |M_6| \cdot K_6 = \int \int_S K dA \quad (7)$$

can also be used to represent the total geodesic curvature of C :

$$\int_C k_g dt = 2\pi - g_S \quad (8)$$

In image processing, we can define S (or C) as a rectangular region. If S only contains one surface point, then g_S is just K . In practice, the vector f_S can be used for a closed surface or a small region centered at one point. Between two regions, there can be intersections or no intersection. The region can be 2D or 3D, depending on whether the problem can be projected into 2D easily. The most popular shapes of the domain regions are circles/spheres and rectangles/cubes.

We have used this technique to analyze human facial data. If the size of the region reduced by 1, 2, 2^2 , ..., 2^k times, we will get a sequence of f_S , or simple get g_S . Then we can see the change in the curvatures. Such a method is usually called a multi-scaling method.

Let us look at the following example. The two original images are shown in Fig 3.

In Fig. 4 and Fig. 5, we show the initial digital curvature calculation—projections from x - and y - directions. In these two figures, There are three symbols, $'*'$, $'+'$, and $'X'$. $'*'$ indicates M_3 -points on the face that are the positive curvature point ($\pi/2$). $'+'$ indicates M_5 -points ($-\pi/2$), and $'X'$ indicates M_6 -points ($-\pi$); they are negative curvature points. The calculation is for the whole 3D. The display only shows in two observation angles for each image.

In Fig. 6, we show the projected Gaussian curvature data for each scale for face-one from $64 \times 64 \times 64$ to $8 \times 8 \times 8$. Fig. 7 is for face-two. In order to be able to observe easily, we will display the same size image when the scale changes.

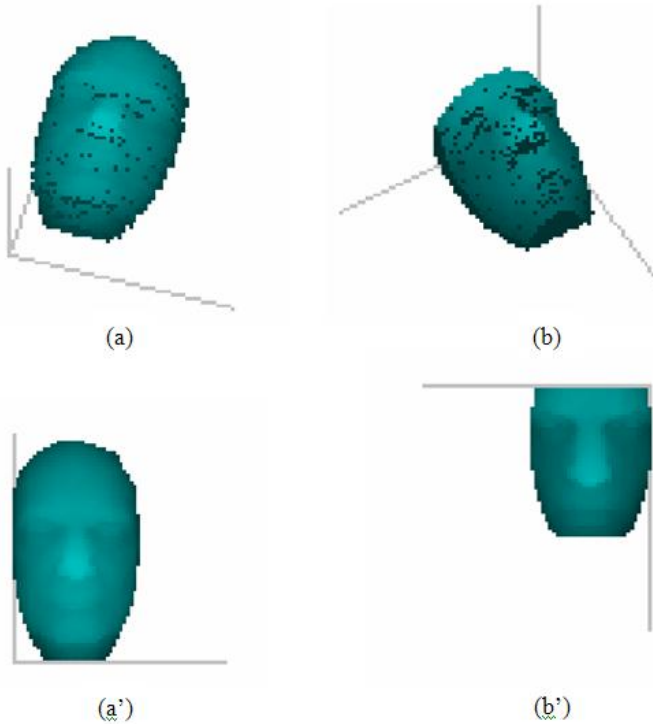


Fig. 3. Two original human faces from NIST-FRGC data sets

Analyzing Figs. 6 and 7, we see that the multi-scaling method can identify some of the interesting areas of the face. For example, we can identify five areas of interest for face-one and face-two. They both contain two areas in the top and in the bottom and one area in the middle. These five areas indicate two region on the head-sides, two cheeks, and the nose. The nonzero total of Gaussian curvature for a region that remains means that total change has not been canceled in this region.

The result shows that the first person has a flatter face than the second person since the method reaches the five interesting areas earlier. We can also identify the flat-or-bending regions in the human face as shown in Fig. 9. A total of four such areas can be easily found.

The advantage of Gaussian curvature-based calculation is that it is not a simple processing of pixel averages in the region. The curvature-based calculation is based on geometric and topological properties of the 3D object. For instance we know the total Gaussian curvature will be a constant as the selected region becomes the whole 3D data array. In summary, the method described in this section can identify the five regions of interest and four flat-or-bending regions

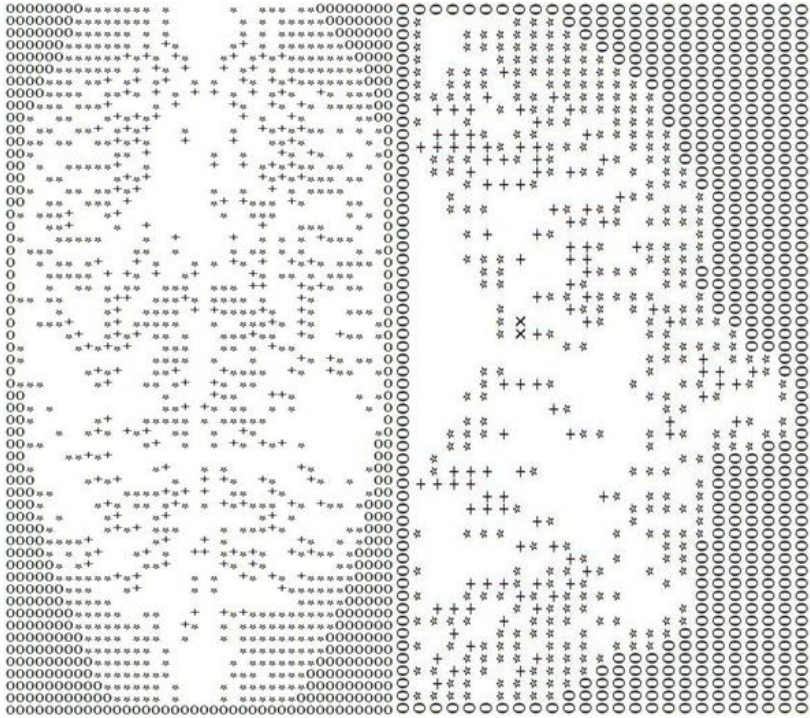


Fig. 4. Digital curvature on face-one’s surface by projection: ‘*’, ‘+’, and ‘X’ indicate M_3, M_5, M_6 -points, respectively

as the common similar characteristics of a human face. addition, the calculation is much easier than that of triangulation based images. The following lemma provides evidence for this.

Lemma 4. *The algorithm designed to obtain scaled local Gaussian curvatures for finding five areas of interest on the image of a human face is of complexity $O(n \log n)$. The space needed is $O(\log n)$.*

Proof: We first compute the curvature for higher resolution image then we reduce the resolution by half. Since the $2^k - 1$ -scaled Gaussian curvature data can be used to calculate the 2^k -scaled Gaussian curvature data. We can design a fast linear algorithm due to the reduction of the size of the array by half. In such a case, the space needed will also be $O(n)$ if we just use the input array and not the intermediate data. By using the output of the result for each resolution or scale, the time complexity will be $O(n \log n)$ and the space complexity will be $O(\log n)$ (consider only the space needed to run the algorithm).



Fig. 5. Digital curvature on face-two's surface by projection

3.2 Investigation Based on Digital Mean Curvatures

For the mean digital curvature application, we have investigated the calculation based on the absolute average for small regions. The geometric meaning of such a treatment for digital mean curvature is the zigzagged points on the surfaces. Mean curvature zero points indicate the critical points that change from inward to outward if it is not a flat point. The result of an example is shown in Fig. 9.

To summarize the findings, Gaussian curvatures describe the global features and the average characteristics such as the five regions of a human face, but mean curvatures find local features and extreme points such as the nose. In the future, we will use this method for analyzing more human facial data. We can normalize the image size and calculate the distance for a pair of faces in corresponding scale or nearby scales. The following section will present a method for 3D data "rough" classification using digital curvature vectors.

4 Object Classification Using Curvature Vectors

In this section, we will explore a method based on the digital curvature vectors for 3D shape similarity analysis and classification of 3D objects. There have been

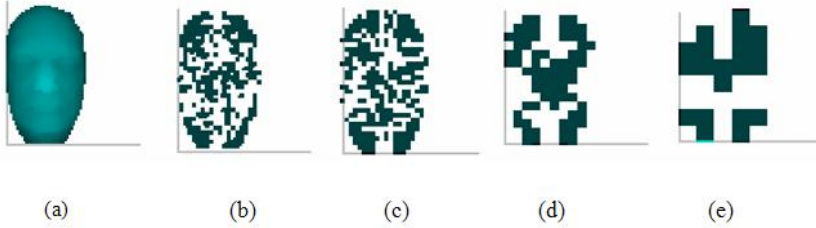


Fig. 6. Digital curvature scaling for Face-one

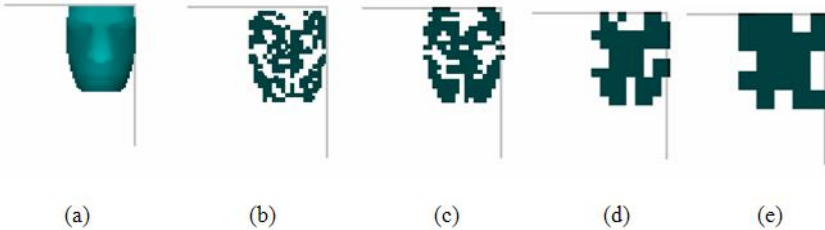


Fig. 7. Digital curvature scaling for Face-two

many research investigations that use local curvatures in 3D shape similarity analysis. Shum, Hebert, and Ikeuchi proposed a method that has $O(n^2)$ time complexity [16]. However, it is not very practical for the purpose of data retrieval. Another disadvantage of the continuous curvature method is that the calculation of curvatures return real numbers and that introduces precision errors [14]. A multi-scaling method was proposed to complete this task, but it takes more time.

The technique presented in this section is not an attempt to replace the methods developed before. We try to explore more applications using digital curvatures. As we discussed in Section 2 and 3, the local digital curvature is determined by the local shape of the digital surface. The number of each type of surface-point may indicate the features of a 3D object.

4.1 Feature Vectors of 3D Object Based on Digital Curvatures

We first define a First, form a feature vector that only contains the number of digital surface points in each of the categories M_3 , M_4 , M_5 , and M_6 . In general, a 3D digital object may not necessarily be a 3D digital manifold. This is because we have strict definitions for 3D manifolds where each local neighborhood must similar to a 3D Euclidean Space. However, this does not affect the use of curvatures in 3D object. For a safe claim, we can assume that the 3D object is simply connected. This is method is used for calculating the boundary surface of a 3D object in this section. The feature vector is

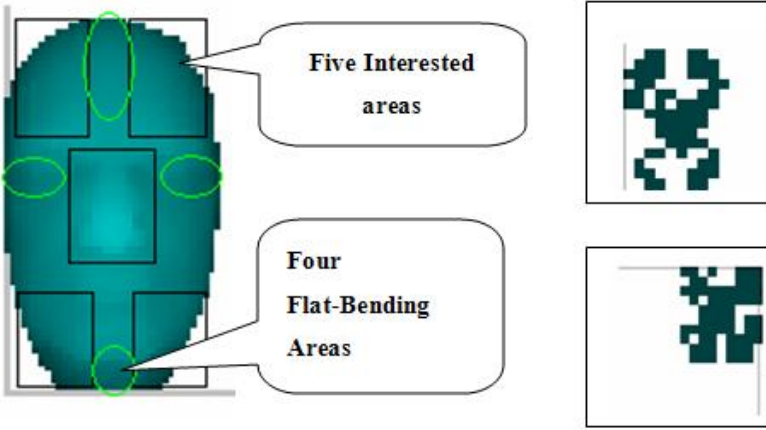


Fig. 8. Five areas of interests and four flat-or-bending areas are the common similar characteristics of human face

$$fv = \frac{1}{T}(|M_3|, |M_4|, |M_5|, |M_6|) = (r_3, r_4, r_5, r_6) \quad (9)$$

where T is the total surface points. For example, an object has a total of 1679 digital points on boundary surface: $|M_3| = 469$; $|M_4| = 995$; $|M_5| = 183$; $|M_6| = 32$. The non-manifold points (2D) are the points where the neighborhood of the point is not a 2D-configuration shown in Fig. 1. its Because of some non-manifold points, we corrected the data for $|M|$'s: 484; 995; 180; 28 ; 240. It also includes the total of 8 none manifold points. The method to delete those points is presented in [7]. Therefore, we get the Ratio $r_3 = 0.288267$, $r_4 = 0.592615$, $r_5 = 0.107207$, and $r_6 = 0.016677$. The Euclidean distance of two feature vectors: $d = \sqrt{\sum_{i=3}^6 ((x_i - y_i)^2)}$

We could also use the scaling method from Section 3 to get more features. Here we use this method to do a rough classification for 3D shapes. The computing examples are taken from the Princeton Benchmark Website [10].

4.2 Similarity and Distance Analysis Based on the Feature Vectors

For n samples of solid objects, we can calculate the feature ratio vectors for each of them. So we will have n feature vectors e_1, \dots, e_n . A simple calculation is to get the Euclidean distances for every pair of points. For the example in Fig. 10, the feature ratio vectors $e_x = (r_3, r_4, r_5, r_6)$ where $r_k = |M_k|/T$ are listed below:

$$\begin{aligned} e_1 &= (0.288267, 0.592615, 0.107207, 0.016677), \\ e_2 &= (0.262424, 0.508752, 0.193369, 0.044133), \\ e_3 &= (0.168149, 0.680220, 0.144854, 0.008895), \\ e_4 &= (0.152833, 0.711492, 0.122506, 0.013966), \\ e_5 &= (0.148500, 0.710425, 0.135432, 0.007128), \\ e_6 &= (0.162700, 0.688310, 0.140705, 0.010093). \end{aligned}$$

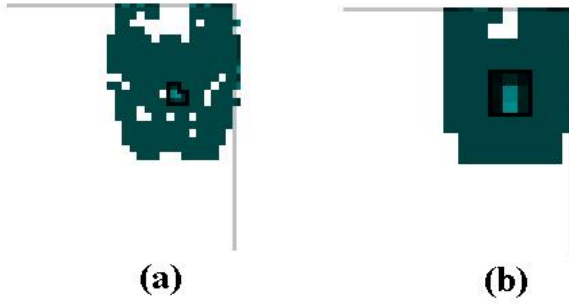


Fig. 9. Use digital mean curvature to find a peak point that indicates nose of human face: (a) Image is made by a 2×2 summation of the absolute values of digital mean curvatures, and (b) Image is made by a 4×4 summation of the absolute values of digital mean curvatures

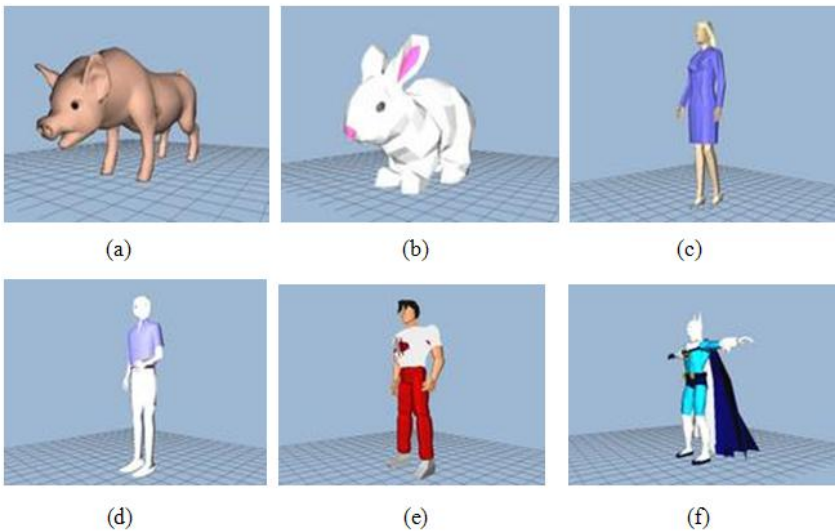


Fig. 10. Six 3D objects from Princeton database

The Distance matrix of two-vector pairs is

$$\begin{pmatrix}
 0 & & & & & & \\
 0.015878586 & 0 & & & & & \\
 0.023580826 & 0.041884473 & 0 & & & & \\
 0.032715518 & 0.059045308 & 0.001737666 & 0 & & & \\
 0.034301844 & 0.058376743 & 0.001390322 & 0.000233753 & 0 & & \\
 0.02609007 & 0.04611817 & 0.000113789 & 0.000980967 & 0.000727309 & 0 & \\
 & & & & & & &
 \end{pmatrix} \quad (10)$$

In the matrix, a_{ij} is the distance between object i and object j . The matrix is symmetric so we only present half of it. We can easily see that Objects 1 and Object 2 are closely related. Objects 4,5, and 6 are similar. From the data pictures displayed in Fig. 10, we can see that is correct. An interesting observation is that Object 3 does not go with three other objects in the second category.

5 Conclusion

In this paper, we have used digital Gaussian curvatures and digital mean curvatures to analyze 3D shapes. We have found that digital curvatures may have some power for identifying the significant features of 3D objects. For instance, we could identify five regions in some facial images. We also presented a method for similarity analysis using digital curvatures in Section 4. In future research, we need to do more investigations including the comparison with other techniques.

Acknowledgement. The authors would like to thank Professor Rama Chellappa for discussing and working with the authors in this research. We would also like to thank the NIST Face Recognition Grand Challenge (FRGC) and Princeton University for the 3D benchmark data sets. The authors would like to thank the reviewers for their comments and suggestions.

References

1. Besl, P.J., Jain, R.C.: Three-dimensional object recognition. *Comput. Surv.* 17(1), 75–145 (1985)
2. Besl, P.J., Jain, R.C.: Invariant surface characteristics for 3D object recognition in range images. *Computer Vision, Graphics, and Image Processing* 33(1), 33–80 (1986)
3. Biswas, S., Aggarwal, G., Chellappa, R.: Invariant geometric representations of 3D point clouds for registration and matching. In: *Proceedings of International Conference on Image Processing, ICIP* (October 2006)
4. Brimkov, V., Klette, R.: Border and surface tracing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(4), 577–590 (2008)
5. Chen, L.: *Discrete Surfaces and Manifolds*. Scientific and Practical Computing, Rockville (2004)
6. Chen, L., Rong, Y.: Digital topological method for computing genus and the Betti numbers. *Topology and its Applications* 157(12), 1931–1936 (2010)
7. Chen, L.: Genus computing for 3D digital objects: Algorithm and implementation. In: Kropatsch, W., Abril, H.M., Ion, A. (eds.) *Proceedings of the Workshop on Computational Topology in Image Context* (2009)
8. Colombo, A., Cusano, C., Schettini, R.: 3D face detection using curvature analysis. *Pattern Recognition* 39(3), 444–455 (2006)
9. Dudek, G., Tsotsos, J.K.: Shape representation and recognition from multiscale curvature. *Computer Vision and Image Understanding* 68, 170–189 (1997)
10. Funkhouser, T., Kazhdan, M., Min, P., Shilane, P.: Shape-based retrieval and analysis of 3D models. *Communications of the ACM* 48(6), 58–64 (2005)

11. Goodman-Strauss, C., Sullivan, J.M.: Cubic polyhedra. In: *Discrete Geometry: In Honor of W. Kuperberg's 60th Birthday. Monographs and Textbooks in Pure and Applied Mathematics*, vol. 253, pp. 305–330 (2003)
12. Kreyszig, E.: *Differential Geometry*. University of Toronto Press (1959)
13. Meyer, M., Desbrun, M., Schroder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: *Visualization and Mathematics III*, pp. 35–58. Springer (2003)
14. Mokhtarian, F., Bober, M.: *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization*. Kluwer (2003)
15. Mokhtarian, F., Khalili, N., Yuen, P.: Estimation of error in curvature computation on multi-scale free-form surfaces. *International Journal of Computer Vision* 48(2), 131–149 (2002)
16. Shum, H.Y., Hebert, M., Ikeuchi, K.: On 3D shape similarity. In: *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR 1996)*, pp. 526–531 (1996)
17. Tanaka, H.T., Ikeda, M., Chiaki, H.: Curvature-based face surface recognition using spherical correlation. In: *Proceedings of Third IEEE International Conference on Principal Directions for Curved Object Recognition Automatic Face and Gesture Recognition*, pp. 372–377 (1998)
18. Worring, M., Smeulders, A.W.M.: Digital curvature estimation. *CVGIP Image Understanding* 58(3), 366–382 (1993)

Discrete Polynomial Curve Fitting to Noisy Data

Fumiki Sekiya¹ and Akihiro Sugimoto²

¹ Graduate School of Advanced Integration Science, Chiba University

sekiya@chiba-u.jp

² National Institute of Informatics

Tokyo, Japan

sugimoto@nii.ac.jp

Abstract. A discrete polynomial curve is defined as a set of points lying between two polynomial curves. This paper deals with the problem of fitting a discrete polynomial curve to given integer points in the presence of outliers. We formulate the problem as a discrete optimization problem in which the number of points included in the discrete polynomial curve, i.e., the number of inliers, is maximized. We then propose a method that effectively achieves a solution guaranteeing local maximality by using a local search, called rock climbing, with a seed obtained by RANSAC. Experimental results demonstrate the effectiveness of our proposed method.

Keywords: Curve fitting, Discrete polynomial curve, Local optimal, Outliers, Consensus set, RANSAC, Discrete geometry.

1 Introduction

Fitting geometric models such as lines or circles is an essential task in image analysis and computer vision, and it is indeed used in feature detection and many other procedures. Though several methods exist for model fitting, they use continuous models even in a discrete environment. The method of least-squares is most common for curve fitting. This method minimizes the sum of squared residuals from all data, and the solution can be analytically computed. This method is, however, fatally susceptible to the presence of outliers: just one outlier can cause a great impact on estimation results. In order to enhance robustness, minimizing the sum of other functions has been proposed. For example, the method of least-absolute-values minimizes the sum of absolute residuals from all data. The method of least median of squares [5] minimizes the median of squared residuals, resulting in tolerating up to half the data being outliers. This means, however, that it does not work in the presence of more outliers. On the other hand, RANdom SAMple Consensus (RANSAC) [2] is commonly used in computer vision. This method maximizes the number of inliers, and work regardless of the fraction of outliers. However, its random approach takes a long time to ensure high accuracy.

In discrete spaces, it is preferable to use discretized models rather than continuous ones because the representation of the models is also discrete. Discrete model fitting in the 2D discrete space is studied for lines [1, 6], annuluses [7], and polynomial curves [4]. For lines and annuluses, methods that work for a data set including outliers, i.e., points that do not describe the model, have been developed, however, such a method that deals with outliers for discrete polynomial curves remains to be reported. This paper aims at developing a method for discrete polynomial curve fitting for a given set of discrete points including outliers.

We formulate the discrete polynomial curve fitting problem as a discrete optimization problem where the number of inliers is maximized. We then propose a method that guarantees its output to achieve local optimal. Our proposed method combines RANSAC and a local search. Namely, starting with a seed obtained by RANSAC, our method iteratively and locally searches for equivalent or better solutions to increase the number of inliers. Our method guarantees the obtained set of inliers is local maximum in the sense of the set inclusion. Experimental results demonstrate the efficiency of our method.

2 Discrete Polynomial Curve Fitting Problem

2.1 Definitions of Notions

A (continuous) polynomial curve of degree k in the Euclidean plane is defined by

$$P = \{(x, y) \in \mathbb{R}^2 : y = a_1x^k + a_2x^{k-1} + \cdots + a_kx + a_{k+1}, a_1 \neq 0\}, \quad (1)$$

where $a_1, \dots, a_{k+1} \in \mathbb{R}$.

We define the discretization of eq. (1), namely, a *discrete polynomial curve*, by

$$D = \{(x, y) \in \mathbb{Z}^2 : 0 \leq y - f(x) \leq w\}, \quad (2)$$

where $f(x) = a_1x^k + a_2x^{k-1} + \cdots + a_kx + a_{k+1}$, and w is a given constant real value. a_i , k and w are respectively called the *coefficient*, the *degree*, and the *width* of the discrete polynomial curve ($i = 1, \dots, k + 1$). Geometrically, D is a set of integer points lying between two polynomial curves $y = f(x)$ and $y = f(x) + w$, and w is the vertical distance between them. We remark that D is a Digital Level Layer (DLL) [3].

We define several notions for a discrete polynomial curve. For a finite set of discrete points

$$S = \{p_j \in \mathbb{Z}^2 : j = 1, 2, \dots, n\},$$

where the coordinates of p_j are finite values, and a discrete polynomial curve D , $p_j \in D$ is called an *inlier*, and $p_j \notin D$ is called an *outlier* of D . The set of inliers is called the *consensus set* of D which is denoted by C . Two polynomial curves $y = f(x)$ and $y = f(x) + w$ are called the *support lines* of D . In particular, we call $y = f(x)$ the *lower support line*, and $y = f(x) + w$ the *upper support line*.

Points on the support lines are called *critical points* of D . In particular, a point on the lower support line is called a *lower critical point*, while that on the upper support line is an *upper critical point*.

2.2 Description of the Discrete Polynomial Curve Fitting Problem

Let $\mathbb{D}_{k,w}$ be the set of all discrete polynomial curves of degree up to k with width w . The problem of discrete polynomial curve fitting is described as follows:

Input. A set of discrete points S , a degree k , and a width w .

Output. A $(k + 1)$ -tuple of coefficients (a_1, \dots, a_{k+1}) of $D \in \mathbb{D}_{k,w}$ having the maximum number of inliers.

A consensus set having the maximum number of inliers, denoted by C_{\max} , is called the *maximum consensus set*. We remark that not less than one optimal solution can exist.

2.3 Discrete Polynomial Curve Fitting in the Parameter Space

A discrete polynomial curve of $\mathbb{D}_{k,w}$ is identified as a point in the parameter space (a_1, \dots, a_{k+1}) . We formulate the problem of discrete polynomial curve fitting as an optimization problem in the parameter space to obtain the maximum consensus set.

Given a point $(x', y') \in S$, (a_1, \dots, a_{k+1}) determining $D \in \mathbb{D}_{k,w}$ such that $D \ni (x', y')$ satisfies

$$0 \leq -x'^k a_1 - \dots - x' a_k - a_{k+1} + y' \leq w . \quad (3)$$

We call the set of such points in the parameter space the *level layer* for (x', y') . (x', y') is a lower critical point when the left-hand side equality is satisfied, and is an upper critical point when the right-hand side equality is satisfied. For a consensus set $C = \{(x_1, y_1), \dots, (x_m, y_m)\}$, we have (a_1, \dots, a_{k+1}) that satisfies

$$\begin{cases} 0 \leq -x_1^k a_1 - \dots - x_1 a_k - a_{k+1} + y_1 \leq w , \\ \vdots \\ 0 \leq -x_m^k a_1 - \dots - x_m a_k - a_{k+1} + y_m \leq w . \end{cases} \quad (4)$$

Letting P_C be the convex polytope (the intersection of these level layers) defined by eq. (4), P_C is the set of (a_1, \dots, a_{k+1}) determining $D \in \mathbb{D}_{k,w}$ such that $D \supset C$ but not $S \cap D = C$. Therefore, D determined by (a_1, \dots, a_{k+1}) in P_C contains at least $|C|$ inliers. For an arbitrary consensus set C' such that $C' \supset C$, $P_{C'} \subset P_C$ since $P_{C'}$ is the intersection of P_C and the level layers for the points in $C' \setminus C$.

Finding C_{\max} is equivalent to finding the convex polytope(s) for C_{\max} in the parameter space. Fig. 1 shows an example of level layers in the case of $k = 1$. Note that an intersection of level layers is always a convex polygon in this case.

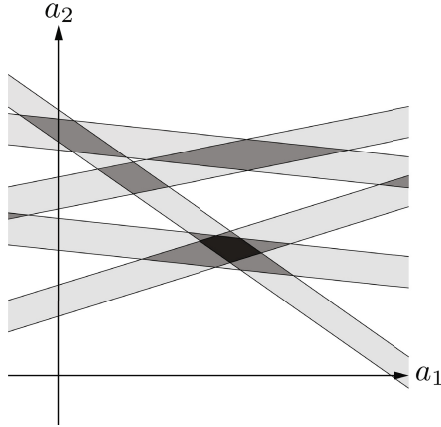


Fig. 1. An example of level layers in the case of $k = 1$. The darkness is proportional to the number of inliers.

If we define $F(a_1, \dots, a_{k+1}) =$ the number of inliers of D determined by (a_1, \dots, a_{k+1}) , then the discrete polynomial curve fitting problem is equivalent to seeking

$$\arg \max_{(a_1, \dots, a_{k+1})} F(a_1, \dots, a_{k+1}) \tag{5}$$

for given S, k , and w .

3 Properties of Discrete Polynomial Curves

A polynomial curve of degree up to k is uniquely determined by different $k + 1$ points on the curve. Theorem 1 states that a discrete polynomial curve also has a similar property.

Theorem 1. *A discrete polynomial curve $D \in \mathbb{D}_{k,w}$ is uniquely determined by $k + 1$ critical points having $k + 1$ different x -coordinates where each of them is specified whether it is an upper or a lower critical point.*

Proof. A discrete polynomial curve $D \in \mathbb{D}_{k,w}$ with $k+1$ critical points $(s_1, t_1), \dots, (s_{k+1}, t_{k+1})$ such that $s_i \neq s_j$ for all $i \neq j$, is identified as a point (a_1, \dots, a_{k+1}) in the parameter space satisfying

$$\begin{cases} -s_1^k a_1 - \dots - s_1 a_k - a_{k+1} + t_1 = c_1, \\ \vdots \\ -s_{k+1}^k a_1 - \dots - s_{k+1} a_k - a_{k+1} + t_{k+1} = c_{k+1}, \end{cases} \tag{6}$$

where

$$c_i = \begin{cases} 0 & \text{if } (s_i, t_i) \text{ is a lower critical point} \\ w & \text{if } (s_i, t_i) \text{ is an upper critical point} \end{cases} \quad (i = 1, \dots, k + 1).$$

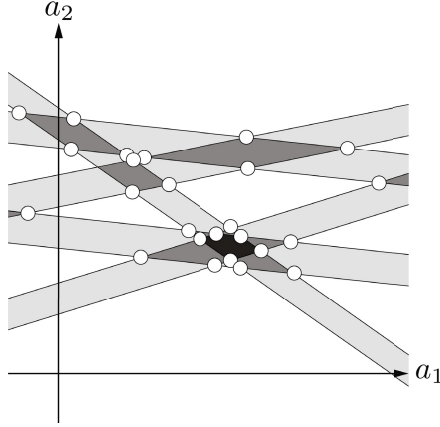


Fig. 2. Discrete polynomial curves of $\mathbb{G}_{S,k,w}$ in the parameter space. They are the intersection points of the boundaries of level layers; the white points represent them.

Eq. (6) has the unique solution in (a_1, \dots, a_{k+1}) because it has $k + 1$ linearly independent equations. \square

We remark that eq. (6) does not have a solution if $s_i = s_j$ for $\exists i, j$ ($i \neq j$).

Theorem 1 indicates that the set of all discrete polynomial curves in $\mathbb{D}_{k,w}$ generated from $k + 1$ points in S is finite where the $k + 1$ points are used as critical points. The set is denoted by $\mathbb{G}_{S,k,w}$. $\mathbb{G}_{S,k,w}$ is not empty iff the points in S have at least $k + 1$ different x -coordinates.

Assume that $\mathbb{G}_{S,k,w}$ is not empty. To identify a discrete polynomial curve in $\mathbb{G}_{S,k,w}$, we consider $2n$ hyperplanes that are the boundaries of the level layers for all points in a given $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$,

$$\begin{aligned} -x_i^k a_1 - \dots - x_i a_k - a_{k+1} + y_i &= 0 \\ -x_i^k a_1 - \dots - x_i a_k - a_{k+1} + y_i &= w \end{aligned} \quad (i = 1, \dots, n). \quad (7)$$

Note that the boundaries of the two level layers for $(x'_1, y'_1) \in S$ and $(x'_2, y'_2) \in S$ are parallel iff $x'_1 = x'_2$. Since $D \in \mathbb{G}_{S,k,w}$ has at least $k + 1$ critical points with $k + 1$ different x -coordinates, (a_1, \dots, a_{k+1}) determining D satisfies at least $k + 1$ independent equations in eq. (7). Therefore, D is an intersection point of the boundaries of the level layers identified by these equations. Fig. 2 shows an example of discrete polynomial curves of $\mathbb{G}_{S,k,w}$ in the parameter space. We remark that for an arbitrary consensus set C , any discrete polynomial curve of $\mathbb{D}_{k,w}$ determined by a vertex of P_C is an element of $\mathbb{G}_{S,k,w}$.

Since $\mathbb{G}_{S,k,w}$ is a finite set, if it contains an element having the maximum consensus set, then we can find the optimal (a_1, \dots, a_{k+1}) (in the sense that it maximizes the number of inliers) by brute-force search in $\mathbb{G}_{S,k,w}$.

Theorem 2. *If $\mathbb{G}_{S,k,w}$ is not empty, then there exists $D \in \mathbb{G}_{S,k,w}$ such that $S \cap D = C_{\max}$.*

To prove Theorem 2, we need the following lemma.

Lemma 1. *If $\mathbb{G}_{S,k,w}$ is not empty, then the points in C_{\max} have at least $k+1$ different x -coordinates.*

Proof. We show that a consensus set C whose points have $m(\leq k)$ different x -coordinates is not maximum. Let X_1, \dots, X_m be these x -coordinates. Then, P_C is written by

$$\begin{cases} L_1 \leq -X_1^k a_1 - \dots - X_1 a_k - a_{k+1} \leq U_1, \\ \vdots \\ L_m \leq -X_m^k a_1 - \dots - X_m a_k - a_{k+1} \leq U_m, \end{cases} \quad (8)$$

where $L_i, U_i \in \mathbb{R}$, and $U_i - L_i \leq w$ for $i = 1, \dots, m$. Since the points in S have at least $k+1$ different x -coordinates, there exists a point $(X, Y) \in S \setminus C$ such that $X \neq X_i$ for $i = 1, \dots, m$. The level layer for (X, Y) is

$$0 \leq -X^k a_1 - \dots - X a_k - a_{k+1} + Y \leq w. \quad (9)$$

There exists at least one solution (a_1, \dots, a_{k+1}) satisfying both of eq. (8) and eq. (9). Therefore, there exists at least one discrete polynomial curve $D' \in \mathbb{D}_{k,w}$ such that $D' \supset C \cup \{(X, Y)\}$, which concludes that C is not maximum. \square

Lemma 1 states that a consensus set whose points have less than $k+1$ different x -coordinates is not maximum. Therefore, we need not consider such consensus sets in proving Theorem 2. We now give the proof of Theorem 2.

Proof. If $P_{C_{\max}}$ is bounded, then each of its vertices corresponds to an element of $\mathbb{G}_{S,k,w}$, from which Theorem 2 is immediately obtained. Therefore, we only have to show that $P_{C_{\max}}$ is bounded.

Since $\mathbb{G}_{S,k,w}$ is not empty, there exist at least $k+1$ points $(u_1, v_1), \dots, (u_{k+1}, v_{k+1}) \in C_{\max}$ such that $u_i \neq u_j$ for all $i \neq j$ thanks to Lemma 1. Any (a_1, \dots, a_{k+1}) in $P_{C_{\max}}$ satisfies

$$\begin{cases} 0 \leq -u_1^k a_1 - \dots - u_1 a_k - a_{k+1} + v_1 \leq w, \\ \vdots \\ 0 \leq -u_{k+1}^k a_1 - \dots - u_{k+1} a_k - a_{k+1} + v_{k+1} \leq w, \end{cases} \quad (10)$$

which can be rewritten as

$$\begin{cases} -u_1^k a_1 - \dots - u_1 a_k - a_{k+1} + v_1 = d_1, \\ \vdots \\ -u_{k+1}^k a_1 - \dots - u_{k+1} a_k - a_{k+1} + v_{k+1} = d_{k+1}, \end{cases} \quad (11)$$

where $0 \leq d_i \leq w$ ($i = 1, \dots, k+1$). We thus obtain

$$\begin{pmatrix} a_1 \\ \vdots \\ a_{k+1} \end{pmatrix} = \begin{pmatrix} -u_1^k & \cdots & -u_1 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ -u_k^k & \cdots & -u_k & 1 \\ -u_{k+1}^k & \cdots & -u_{k+1} & 1 \end{pmatrix}^{-1} \begin{pmatrix} d_1 - v_1 \\ \vdots \\ d_k - v_k \\ d_{k+1} - v_{k+1} \end{pmatrix}. \quad (12)$$

Denoting the (i, j) entry of the inverse matrix by m_{ij} allows eq. (12) to be written as

$$a_i = \sum_{j=1}^{k+1} m_{ij}(d_j - v_j) \quad (i = 1, \dots, k+1). \quad (13)$$

Eq. (13) shows that a_i is linear in d_1, \dots, d_{k+1} . Therefore, the set of (a_1, \dots, a_{k+1}) satisfying eq. (10) is bounded since $0 \leq d_i \leq w$. $P_{C_{\max}}$ is its subset, and consequently is bounded. \square

Theorem 2 states that the consensus sets $\{S \cap D : D \in \mathbb{G}_{S,k,w}\}$ contain all the maximum consensus sets. Therefore, if $\mathbb{G}_{S,k,w}$ is not empty, then all the maximum consensus sets are found by the brute-forth search. Hereafter, we assume that $\mathbb{G}_{S,k,w}$ is not empty, which almost always holds.

4 Discrete Polynomial Curve Fitting Algorithm

RANSAC iteratively generates model parameters by randomly sampling points from a given set to find the ones describing a largest number of points in the set. Finding all the maximum consensus sets by RANSAC requires to compute the consensus sets for all the discrete polynomial curves of $\mathbb{G}_{S,k,w}$, which is computationally expensive and impractical. In fact, the brute-forth search requires up to $2^{k+1} \binom{|S|}{k+1}$ iterations. In this section, we propose a method that effectively achieves a solution guaranteeing local optimality in the sense of the set inclusion by introducing a local search.

We define neighbors in $\mathbb{G}_{S,k,w}$ for our local search. When $D \in \mathbb{G}_{S,k,w}$ is given, we define neighbors of D denoted by N_D as the discrete polynomial curves having k upper and lower critical points all of which are identical with those of D where the x -coordinates of the critical points are different from each other. Note that $D \notin N_D$. Then, (a_1, \dots, a_{k+1}) of $D' \in N_D$ satisfies the same k independent equations as that of D in eq. (7). Therefore, (a_1, \dots, a_{k+1}) corresponding to D' is on the intersection line of the k hyperplanes that are the boundaries of the level layers identified by these equations. Thus, the neighboring relations are determined by the intersection lines of k boundaries of level layers. We call these lines neighboring lines. Fig. 3 shows an example of neighbors in the parameter space when $k = 1$. In this case, the neighboring lines are identical to the boundaries of level layers themselves. We call D' having at least the same number of inliers a *good* neighbor of D .

Our method consists of two steps (Algorithm 1). In the first step, we use RANSAC to obtain a seed for the second step. In the second step, we introduce a local search, called *rock climbing*, to increase the number of inliers. Given an initial seed (discrete polynomial curve) obtained by RANSAC, rock climbing searches the discrete polynomial curves having a largest number of inliers among the seed and its neighbors, and then iterates this procedure using the obtained curves as new seeds. Algorithm 2 describes the concrete procedure of rock climbing.

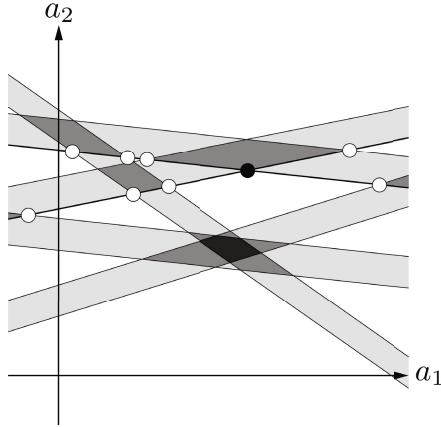


Fig. 3. An example of the neighbors ($k = 1$). The neighbors of the black point are depicted with white points. They are on the neighboring lines, i.e., lines passing through the black point.

Algorithm 1. Our method

Input: A set of discrete points S , a degree k , a width w , a number of iterations t for RANSAC.

Output: A set of discrete polynomial curves.

Run RANSAC with t iterations.

Run rock climbing using a seed obtained by RANSAC.

return The output of rock climbing.

A consensus set C is called *local maximum* when no consensus set exists that is a superset of C . We denote a local maximum consensus set by C_{local} .

Theorem 3. *Rock climbing outputs discrete polynomial curves that correspond to all the vertices of a $P_{C_{\text{local}}}$.*

Proof. Let C be the consensus set of the current discrete polynomial curve.

We first consider the case of $C = C_{\text{local}}$. Any two vertices of a convex polytope are reachable with each other by tracing edges of the polytope. This means that we can obtain all the vertices of $P_{C_{\text{local}}}$ by propagating the neighboring relation from the current vertex, since each edge of P_C is a part of a neighboring line. Furthermore, any (a_1, \dots, a_{k+1}) in $P_{C_{\text{local}}}$ satisfies $F(a_1, \dots, a_{k+1}) = |C_{\text{local}}|$. Consequently, we can obtain all the vertices of $P_{C_{\text{local}}}$ by iteratively searching good neighbors.

If $C \neq C_{\text{local}}$, then a consensus set $C' = C \cup (x', y')$ exists where $(x', y') \in S \setminus C$. $P_{C'}$ is the intersection of P_C and the level layer for (x', y') . Therefore, each vertex of $P_{C'}$ is on an edge or a vertex of P_C as illustrated in Fig. 4. This means that we can obtain all the vertices of $P_{C'}$ by propagating the neighboring relation

Algorithm 2. Rock climbing**Input:** S, k, w , an initial discrete polynomial curve $D_{\text{init}} \in \mathbb{G}_{S,k,w}$.**Output:** A set A of discrete polynomial curves. $A := \{D_{\text{init}}\}$ **loop** $A' :=$ A set of discrete polynomial curves in $\left(A \cup \bigcup_{D \in A} N_D\right)$ having a largest number

of inliers

if $A = A'$ **then**

Break out of the loop

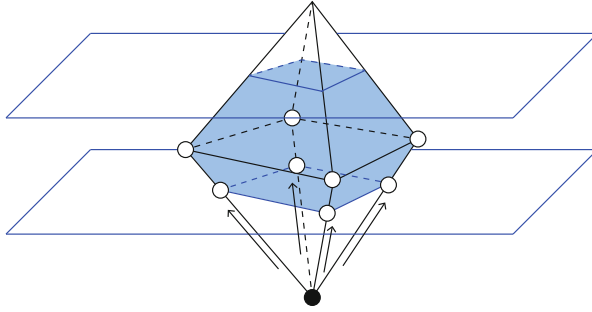
else $A := A'$ **end if****end loop****return** A 

Fig. 4. P_C (black) and $P_{C'}$ (blue). Each vertex of $P_{C'}$ is on an edge or a vertex of P_C . Suppose that the black point corresponds to the current polynomial curve. Then the white points are the neighbors in P_C .

from the current vertex of P_C . Furthermore, any (a_1, \dots, a_{k+1}) in P_C satisfies $F(a_1, \dots, a_{k+1}) \geq |C|$. Consequently, we can obtain all the vertices of $P_{C'}$ by iteratively searching good neighbors. This discussion holds as long as $C \neq C_{\text{local}}$. By repeating this procedure, we finally obtain $C' = C_{\text{local}}$. \square

From Theorem 3, we can always find all the vertices of a $P_{C_{\text{local}}}$ by rock climbing. Therefore, we can generate all (a_1, \dots, a_{k+1}) 's determining D such that $D \supset C_{\text{local}}$ from these vertices.

It should be noted that our method does not always terminate immediately at a local optimal consensus set. Rock climbing examines every neighbor to seek good ones, and rock climbing does not terminate as long as good neighbors exist.

Rock climbing has possibilities of not achieving a global optimum. Its output depends on an initial seed. Having a “good” seed will be preferable. That is why we use RANSAC to obtain an initial seed having as many inliers as possible.

5 Experiments

To demonstrate the effectiveness of our proposed method, we compared our method with RANSAC under two different scenarios. First, we fixed the ratio between inliers and outliers among input points and changed the number of input points. Then, we evaluated the computational time required to obtain the maximum number of inliers. Second, we fixed the number of input points and changed the ratio between inliers and outliers. Then, we evaluated the computational time. In both cases, we observed that our method outperforms RANSAC.

For the first scenario, we set $k = 2, w = 1$ and fixed the ratio of outliers among input points to be 25%, 50%, 75%. For each fixed ratio, we generated five different discrete input point sets S , where $|S|$ was changed by 40 from 40 to 200 (see Fig. 5 for examples). In each set, integer points satisfying $0 \leq y - 0.01x^2 \leq w$ ($-100 \leq x \leq 100$) were randomly generated for inliers (blue points) and integer points that do not satisfy this inequality were randomly generated within $[-100, 100] \times [-100, 100]$ for outliers (red points). We remark that we designed in each fixed outlier ratio, all the five input point sets have the same optimal solutions in $\mathbb{G}_{S,k,w}$ ($k = 2, w = 1$). (Data-sets having different outlier ratios do not have the same optimal solutions.)

To these data-sets, we applied our method 100 times independently where we set $t = 1000$ (the number of iteration for our RANSAC step). We then evaluated the computational time to obtain C_{\max} (a consensus set having the maximum number of inliers) in terms of the required number of samplings there. Note that one sampling takes the same computational time and thus the number of samplings can be a measurement for the computational time. For comparison, we applied RANSAC alone without setting any limited number of iterations, and terminated it when C_{\max} is obtained.

The average number of samplings over the 100 trials is given in Table 1 and illustrated in Fig. 6. We see that our method finds C_{\max} more than twice faster than RANSAC and that the difference of required numbers of samplings to find C_{\max} drastically becomes larger as the number of input points increases. From Fig. 6, we can also observe that regardless of outlier ratios, the required number of samplings has a similar behavior depending on the number of input points. Namely, the required number of samplings slowly increases and is not exponentially affected by the number of input points for our method while it exponentially increases for RANSAC. We can thus conclude that the number of input points has far less impact on our method than RANSAC.

For the second scenario, we again set $k = 2, w = 1$ and fixed the number of input points to be 200. We generated nine different discrete input point sets, where the ratio of outliers was changed by 10% from 10% to 90% (see Fig. 7 for examples). In each set, inliers and outliers are generated in the similar way as the first scenario. To these data-sets, we applied our method 100 times independently and evaluated the required number of samplings to obtain C_{\max} . We also applied RANSAC alone using the same condition as the first scenario case.

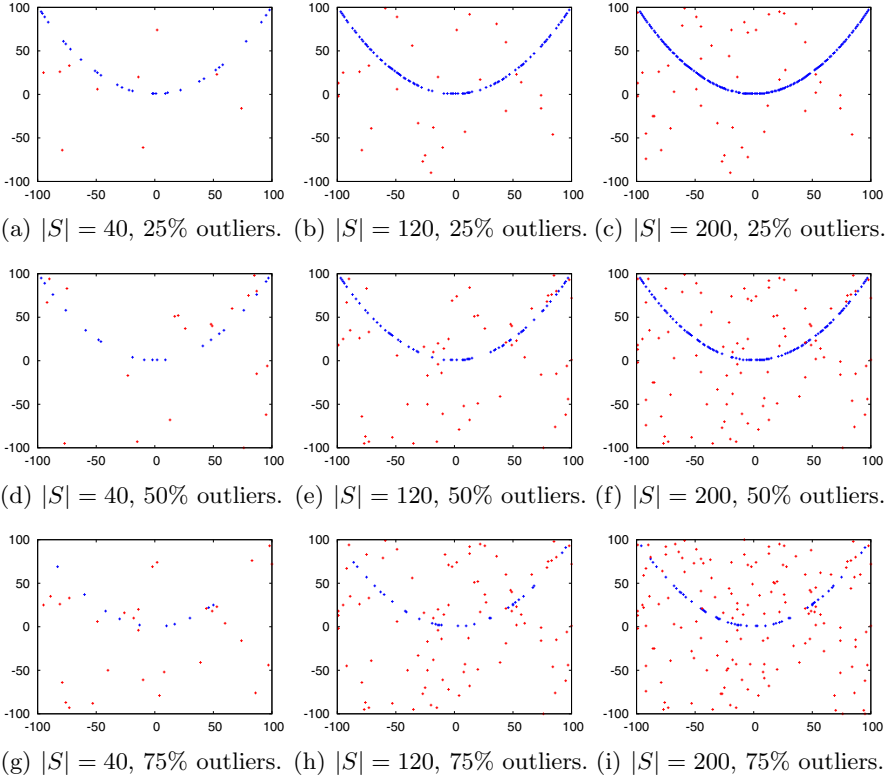


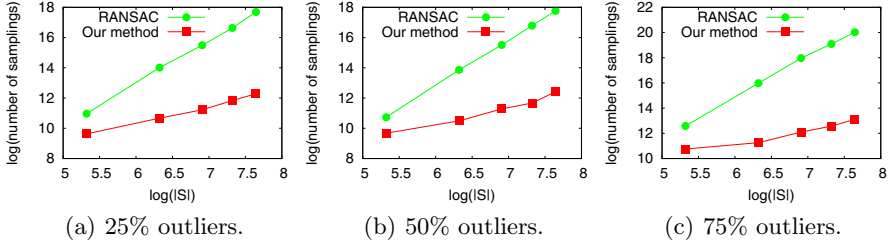
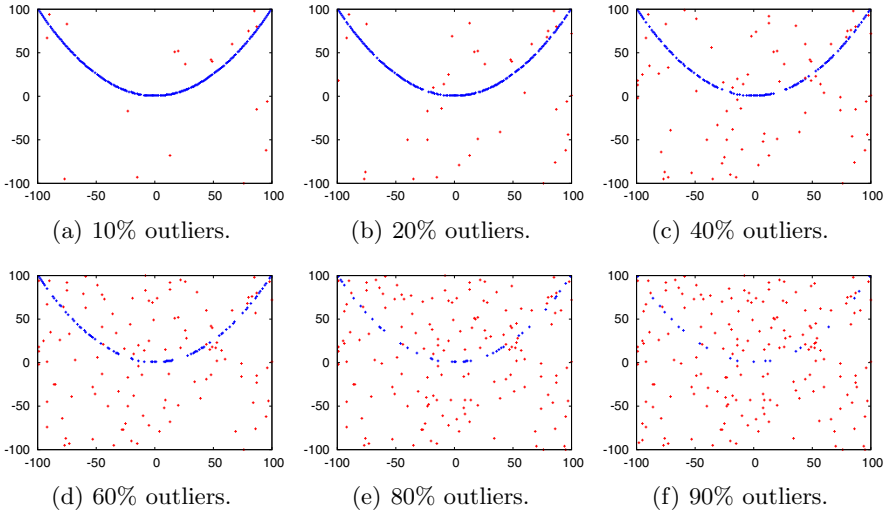
Fig. 5. Input set examples with different numbers of points and different outlier ratios ($k = 2$). (a), (b), (c) are for 25% outliers; (d), (e), (f) are for 50% outliers; (g), (h), (i) are for 75% outliers.

Table 2 and Fig. 8 show the average number of samplings over the 100 trials. From these results, we can see that our method finds C_{\max} more than ten times faster than RANSAC. We can also observe that in both methods, the outlier ratio does not affect the required number of samplings as far as the number of input points is the same. We remark that in our method, the required number of samplings in the case where the outlier ratio is 90% (in this case, the number of inliers is 20 while that of outliers is 180) becomes almost twice of that for the other cases. This suggests that there may be a minimum number of inliers required for our method to work effectively. Investigating this is left for future work.

So far, we had experiments only for quadratic curves ($k = 2$). In order to confirm our observations even for another order case, we conducted the same experiments under the condition of $k = 3$ and $w = 1$. As input points, we randomly generated inliers satisfying $0 \leq y - 0.0001x^3 \leq w$ ($-100 \leq x \leq 100$)

Table 1. Number of samplings ($\times 10^3$) required for achieving C_{\max} ($k = 2$)

ratio of outliers (%)	$ S $	40	80	120	160	200
25	our method	0.8	1.6	2.4	3.6	5.0
	RANSAC	2.0	16.5	46.1	101.7	211.2
50	our method	0.8	1.4	2.4	3.2	5.4
	RANSAC	1.7	14.9	46.6	113.1	223.4
75	our method	1.7	2.4	4.3	6.0	8.8
	RANSAC	6.1	64.4	256.1	560.1	1062.0

**Fig. 6.** Required number of samplings depending on $|S|$ ($k = 2$)**Fig. 7.** Input set examples with different outlier ratios under the same number of points ($|S| = 200$, $k = 2$)**Table 2.** Number of samplings ($\times 10^3$) under different outlier ratios ($k = 2$)

ratio of outliers (%)	10	20	30	40	50	60	70	80	90
our method	4.7	4.6	4.2	4.6	4.6	4.4	4.6	4.3	7.0
RANSAC	76.7	80.1	75.9	87.7	81.2	74.9	75.8	72.0	77.4

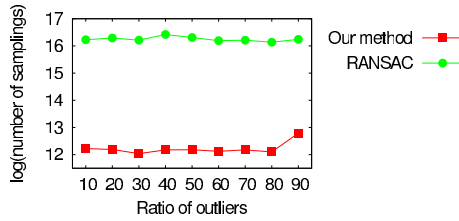


Fig. 8. Required number of samplings depending on outlier ratio ($k = 2$)

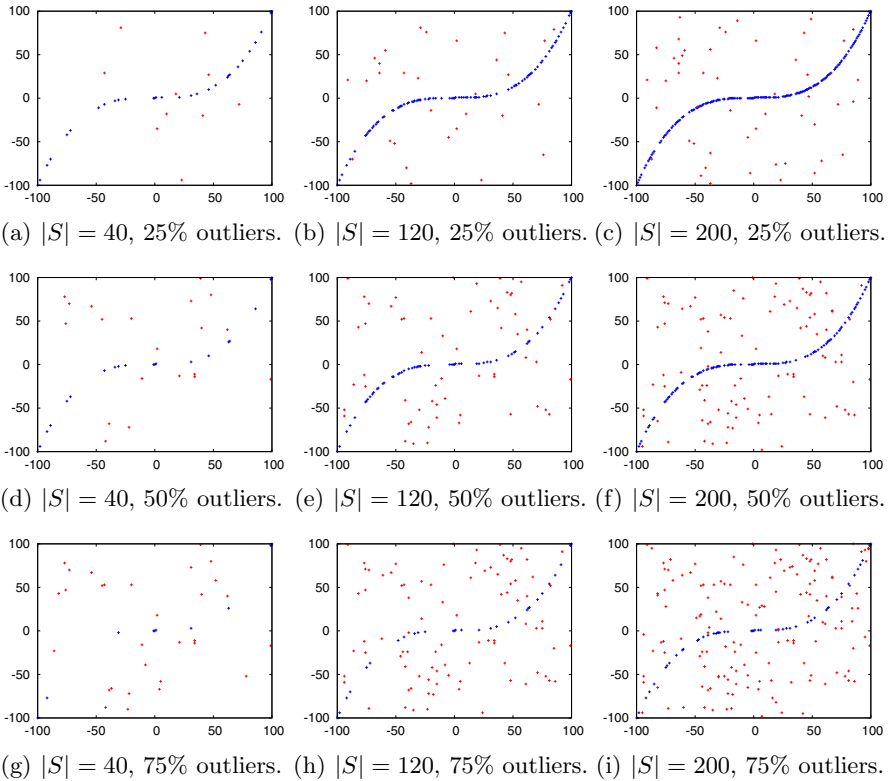


Fig. 9. Input set examples with different numbers of points and different outlier ratios ($k = 3$). (a), (b), (c) are for 25% outliers; (d), (e), (f) are for 50% outliers; (g), (h), (i) are for 75% outliers.

Table 3. Number of samplings ($\times 10^3$) required for achieving C_{\max} ($k = 3$)

ratio of outliers (%)	$ S $	40	80	120	160	200
25	our method	1.5	4.2	8.4	11.9	16.5
	RANSAC	23.5	374.2	2321.9	7380.0	14749.4
50	our method	2.1	6.3	11.4	15.8	20.2
	RANSAC	51.0	963.4	4784.3	24186.0	50754.9
75	our method	7.4	7.5	13.5	17.8	26.2
	RANSAC	55.8	1052.4	4890.6	14855.5	54525.0

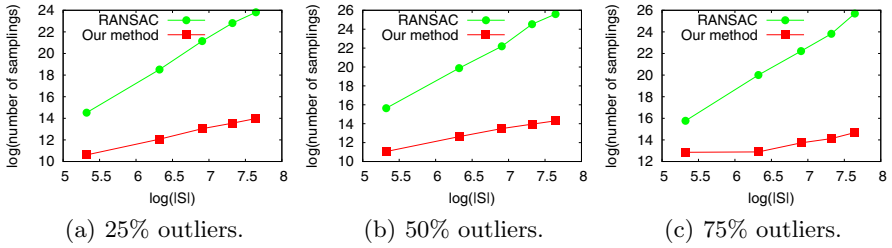


Fig. 10. Required number of samplings depending on $|S|$ ($k = 3$)

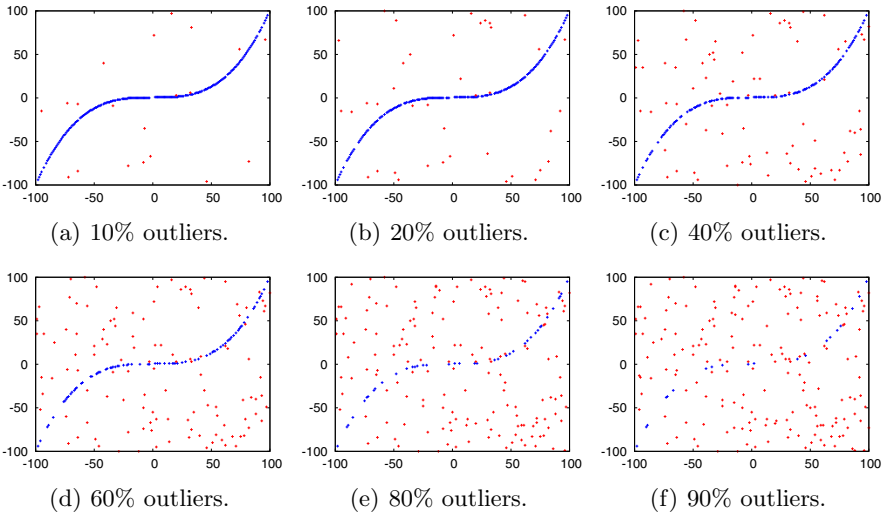


Fig. 11. Input set examples with different outlier ratios under the same number of points ($|S| = 200$, $k = 3$)

Table 4. Number of samplings under different outlier ratios ($k = 3$)

ratio of outliers (%)	10	20	30	40	50	60	70	80	90
our method ($\times 10^4$)	1.7	1.8	1.9	1.7	1.9	2.0	1.9	2.6	5.7
RANSAC ($\times 10^7$)	2.9	2.6	2.9	2.9	2.4	2.8	2.6	2.9	3.1

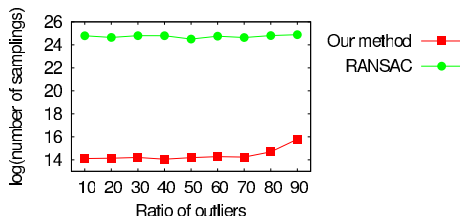


Fig. 12. Required number of samplings depending on outlier ratio ($k = 3$)

and outliers over $[-100, 100] \times [-100, 100]$ so that no outlier satisfies this inequality (see Figs. 9 and 11 for examples). The results are shown in Tables 3 and 4 and Figs. 10 and 12. From these results, we have the same observation as the quadratic curves case. We can thus conclude that our method significantly outperforms RANSAC.

6 Conclusion

This paper dealt with the problem of fitting a discrete polynomial curve to a given set of points including outliers. We formulated this problem as an optimization problem where the number of inliers is maximized. Our proposed method effectively searches solutions by rock climbing using an initial seed obtained by RANSAC. We showed that our method guarantees local maximality of inliers in the sense of the set inclusion. The effectiveness of our method was demonstrated using experiments.

Acknowledgements. This work is in part supported by Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan under the contract of 23650092 and by a Japanese-French joint research project called the SAKURA program.

References

1. Aiger, D., Kenmochi, Y., Talbot, H., Buzer, L.: Efficient Robust Digital Hyperplane Fitting with Bounded Error. In: Domenjoud, E. (ed.) DGCI 2011. LNCS, vol. 6607, pp. 223–234. Springer, Heidelberg (2011)
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)

3. Gérard, Y., Provot, L., Feschet, F.: Introduction to Digital Level Layers. In: Domenjoud, E. (ed.) DGCI 2011. LNCS, vol. 6607, pp. 83–94. Springer, Heidelberg (2011)
4. Provot, L., Gérard, Y.: Estimation of the Derivatives of a Digital Function with a Convergent Bounded Error. In: Domenjoud, E. (ed.) DGCI 2011. LNCS, vol. 6607, pp. 284–295. Springer, Heidelberg (2011)
5. Rousseeuw, P.: Least median of squares regression. *Journal of the American Statistical Association*, 871–880 (1984)
6. Zrour, R., Kenmochi, Y., Talbot, H., Buzer, L., Hamam, Y., Shimizu, I., Sugimoto, A.: Optimal consensus set for digital line and plane fitting. *International Journal of Imaging Systems and Technology* 21(1), 45–57 (2011)
7. Zrour, R., Largeteau-Skapin, G., Andres, E.: Optimal Consensus Set for Annulus Fitting. In: Domenjoud, E. (ed.) DGCI 2011. LNCS, vol. 6607, pp. 358–368. Springer, Heidelberg (2011)

A Probabilistic Measure of Circularity

Ana Marcela Herrera-Navarro¹,
Hugo Jiménez-Hernández², and Iván Ramón Terol-Villalobos³

¹ Facultad en Ingeniería, Universidad Autónoma de Querétaro,
76000, Querétaro, México

² CIDESI, Av. Playa Pie de la Cuesta No. 702, Desarrollo San Pablo, 76130,
Querétaro, México

³ CIDETEQ, S.C., Parque Tecnológico Querétaro S/N,
San Fandila-Pedro Escobedo, 76700, Querétaro, México
anaherreranavarro@gmail.com

Abstract. The circle is a useful morphological structure: in many situations, the importance is focused on the measuring of the similarity of a perfect circle against the object of interest. Traditionally, the well-known geometrical structures are employed as useful geometrical descriptors, but an adequate characterization and recognition are deeply affected by scenarios and physical limitations (such as resolution and noise acquisition, among others). Hence, this work proposes a new circularity measure which offers several advantages: it is not affected by the overlapping, incompleteness of borders, invariance of the resolution, or accuracy of the border detection. The propounded approach deals with the problem as a stochastic non-parametric task; the maximization of the likelihood of the evidence is used to discover the true border of the data that represent the circle. In order to validate the effectiveness of our proposal, we compared it with two recently effective measures: the mean roundness and the radius ratio.

Keywords: Measure, Circularity, Shape, Disk, Topology.

1 Introduction

The analysis of the shapes of objects has been of great interest in many areas, such as medicine [6], materials science [16] and industrial processing [24,4]. In fact, the measurement of shapes is an ongoing research topic, particularly in digital image processing and discrete geometry [18,10,23]. Even though several shape descriptors are useful for describing and differentiating a variety of objects – circles, ellipses, rectangles, and reclines [22] – its computation is still a tough task. The main reason is that many descriptors are affected by factors such as resolution in the representation, small irregularities in the contours (perimeter inaccuracy) and noise. Furthermore, these factors are sensitive to different aspects of the shape (for instance, regular or irregular shapes).

The majority of shape measures are focused on a two dimensional space and are commonly represented in a plane or as images in which one of the most useful

measures is the circularity of the shape [24,21,19,8]. The most widely used measure of circularity is the so-called shape factor (SF) given by $SF = 4\pi Area/P^2$ [3], where P is the perimeter. This measure is defined as the ratio of the section area of an object to its perimeter. The digital equivalent of this circularity measure was introduced in [13]. This measure has several drawbacks in practice: (i) it is not perfectly scale invariant, (ii) it is difficult to interpret, (iii) it is highly sensitive to small irregularities in the contours, (iv) it is dependent on the resolution, and (iv) it is affected by overlapping, rendering impossible to characterize partially visible objects. These problems have motivated many authors to propose new measures of circularity [19,13,1]. Recently, Ritter and Cooper [21] reviewed and compared seven measures of circularity in terms of resolution dependence, demonstrating that most of them are derivations of SF. In their work, the authors proposed two new measures of roundness: the mean roundness (MR) and the radius ratio (RR). These measures can be useful to assess the circularity of regular objects in a two-dimensional space; however, when the information of the object is given partially or there are abrupt changes in the contours they become inefficient.

As commented above, the different approaches try to exploit geometrical information for measuring circularity. In summary, the approaches found in the literature to compute the circularity can be divided in three main groups: *Approaches based on the circular Hough transformation* (HT) [9,7,26]; their main disadvantage is the fact that computational and storage requirements of the algorithm increase exponentially to the dimensionality of the curve.

Approaches based on the separation of the circle problem into discrete and computational geometry [18,10,25,5,2]; however, these algorithms need to be extended to measure the extent of the deviation with a digital arc. *Approaches based on the reference shape* [15,19,1]; which has the following drawbacks: the generation of a digitized disk adds more complexity and it is necessary to know the real object to generate a digitized disk according to the shape resolution. *Approaches based on circle fitting* [20,14,11]; in some cases these methods offer solutions with minimum error, but they are not necessarily the best solutions to the data. In sum, the reader may find in the literature several properties that a good measure of circularity should have. One of the most representative is the work of Haralick [13]. In his work, the author introduces four properties to construct a good measure of circularity of closed figures: (i) the more a figure becomes circular, the more the measure of its circularity increases, (ii) the values for digital figures follow the values for the corresponding continuous figures, (iii) the circularity measure is independent of the orientation, and (iv) the circularity measure is area independent.

This paper focuses on introducing a new framework to measure circularity, which is not affected by overlapping, incompleteness of the borders, invariance of the resolution, or accuracy of border detection, providing a good balance between measure accuracy and the computational resources needed. This framework is based on the generalization of the concept of disk in spaces with high dimension under a certain induced norm. This yields a framework in which the roundness

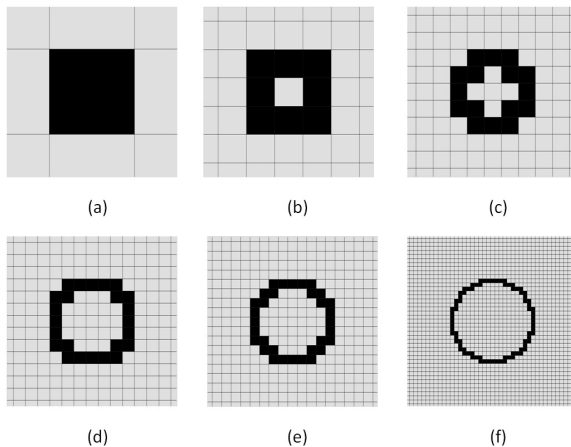


Fig. 1. The shape of a circle in a discrete space. The circle becomes deformed by the resolution, being hard to decide whether it represents a circle. The examples show different discrete circles with radius equal to: (a) 0, (b) 1, (c) 2, (d) 3, (e) 4, and (f) 10, given pixels.

is conceived by its properties instead of by the topology of a particular space, thus allowing the definition and measurement of the roundness of objects in a non-Euclidean space, which can be useful in a range of applications. Moreover, experiments are conducted to demonstrate the effects of the measure in a real example. In particular, the case of a two dimensional image is analyzed and matched with two circularity measures recently proposed.

This paper is organized as follows. In Section 2, the foundations of the circle and disk, together with their main properties, are introduced. In an initial part, possible situations which could affect the robustness of the measures are discussed, followed by a short description of the norm. In Section 3, a new discrete measure based on the distribution of the radius is proposed. In Section 4, the proposal is tested under the Euclidean norm with other well-known accepted circularity measures: radius ratio (RR) and mean roundness (MR). In addition, a real example to compare these measures is presented. Subsequently, the results are discussed showing the main advantages of our proposal with real digital images. Finally, comments and conclusions can be found in the last section.

2 Foundations

In practical scenarios, the existent circularity measures become insufficient because the discretization process of the image may deform the shape of the involved objects as is appreciated in Fig. [1](#).

2.1 Basic Notions and Preliminaries

The definition of a circle entails a center position c and a neighborhood of size r . Assuming that c is in \mathbb{R}^2 and $r \in \mathbb{R}^+$, the disk is defined as:

$$O(c, r) = \{x_i | d(c, x_i) \leq r\} \quad (1)$$

where $d(c, x_i) = \sqrt{(c_1 - x_{i1})^2 + (c_2 - x_{i2})^2}$, $r \in \mathbb{R}^+$ and $c, x_i \in \mathbb{R}^2$. As a consequence, the circle is defined as follows:

$$C(c, r) = \{x_i | d(c, x_i) = r\}. \quad (2)$$

Both equations (1) and (2) represent infinite sets \mathbb{H} under the corresponding restrictions. However, this definition is limited to a two dimensional space, that is, any measure based on these assessments is dependent on the dimensionality of the space.

2.2 The Shape of the Circle and the Norm

In general terms, by preserving the definitions of circle and disk, we can substitute the L_2 norm used to measure the distance by any L_k norm. As a result, the shape and form of the circle changes, i.e., the topology of the neighborhood is completely different and depends on each norm. Then, the norm L_k defines a distance function d_k as follows

$$d_k(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^k \right)^{\frac{1}{k}}, \quad (3)$$

where $||$ is the absolute value.

3 Circularity Measure Framework

Whenever the circle is discretized, the pdf becomes affected by the density of the discretization, changing the form to a Gaussian-like form, where the μ parameter represents the radius and the σ parameter the sparseness. These parameters have a direct relation to the discretization process, i.e. $f(r) \approx G(\mu, \sigma)$. In other words, when the circle becomes bigger the density function tends to a Gaussian. However, environment perturbations of the discretization process cause that the $f(r)$ of the radius is composed of distinct peaks or modes. Therefore, the radius distribution, without loss of generality, can be modeled as mixed distributions. This model can become more complicated when there are insufficient data to estimate the set of parameters for each density function.

¹ Under the assumption that they are represented in a continuous space.

3.1 The Circularity Measure

Let us consider that the magnitude of the radius can be represented as a random variable r with a distribution $f(r)$. Whenever the evidence of the elements belonging to the circle C' are the result of an acquisition process, they represent a subset of $C(c, r)$, and the $f(r)$ distribution becomes sparse. The expected value $E[f(r)]$ for $f(r)$ corresponds to the most probable value for the radius. The area surrounding the maximum of $f(r)$ denotes the probability $Pr(C')$, which represents the circle $C(c, r)$ of radius r and center c .

$$Pr(C') = \int_a^b f(r)dr \tag{4}$$

such that $C' = \{x_1, x_2, \dots, x_n\}$ and $x_i \in \mathbb{R}^n$, where a and b represent the interval of confidence. The a and b values are defined as $E[f(r)] - g_1(r, k)$ and $E[f(r)] + g_2(r, k)$ respectively, such that functions g_1 and g_2 denote the maximum sparse criterion.

Let $C' = \{x_1, x_2, \dots, x_n\}$ be such that each $x_i \in \mathbb{R}^2$ and $d_k(c, r)$ is a two dimensional norm k and $c = (x, y)$ is the center, then the roundness measure is defined as:

$$MOR(C') = \int_{E[f(r)]-g_1(r,k)}^{E[f(r)]+g_2(r,k)} f(r)dr \tag{5}$$

The proposed measure results invariant to the norm used, in the sense that it follows the circle restriction regardless of the type of norm used.

3.2 Parameter Estimation

In some cases note that parameters c and r of the measure must be previously estimated to make a good measurement; however, in real scenarios, it is difficult to provide a good estimation of these parameters. To figure out the estimation task, the measure of parameters will be considered in four main situations discussed below:

1. **The objects to be measured have closed borders.** For a given border denoted by $C' = \{x_1, x_2, \dots, x_n\}$, where each $x_i \in \mathbb{R}^2$ under the d_k norm, the center of the object is defined as:

$$c = E\{x_1, x_2, \dots, x_n\} \tag{6}$$

and the radius distribution is defined as follows:

$$r = \{d_k(c, x_1), d_k(c, x_2), \dots, d_k(c, x_n)\}, \tag{7}$$

where the pdf of $r \sim f(r)$

2. **The objects to be measured contain partial information.** Assuming that the evidence of the border C' can be represented as a differentiable

parametrized curve in \mathbb{R}^2 , the radius of circularity for the evidence C' is estimated as follows

$$r = \frac{|\gamma'|^3}{\sqrt{|\gamma'|^2|\gamma''|^2 - (\gamma' \cdot \gamma'')^2}} \quad (8)$$

such that $\gamma'_t = \gamma_{t+1} - \gamma_{t-1}$ and $\gamma''_t = \gamma'_{t+1} - \gamma'_{t-1}$, where $\gamma^{(n)}$ is expressed as centered differences for the discrete case. Once the radius is estimated, the center of the circle is located at a distance k from a vector orthogonal to its derivative, which is denoted as follows

$$c = \gamma'(t) + \frac{-\gamma'(t)r}{\|\gamma'(t)\|} \quad (9)$$

3. **The border of the circle is not connected.** The estimation of the center is computed as the expected value of the estimated centers of all disconnected borders, i.e., for a given set of borders $\Omega = \{C'_1, C'_2, \dots, C'_k\}$ corresponding to the same object, the estimated radius and centers are estimated as described above and denoted as $\Phi = \{r_1, r_2, \dots, r_k\}$, $\Psi = \{c_1, c_2, \dots, c_k\}$. Consequently, the radius of the semi arcs in Ω are defined as:

$$E[\Phi] = \arg \max\{r_1, r_2, \dots, r_k\} \quad (10)$$

4. **The amount of information in the borders.** The border of the object may be closed or opened. In the first case, the situation is quite similar to the one mentioned above, with the consideration that a discretization process involves a uniform sampling over the ideal border. However, when the cardinality $|r|$ is considerable, then to reduce the complexity of the computation, $f(r)$ can be estimated. Assuming a Gaussian parametric form, the estimation may be performed by considering an optimization process, in which the derivative of vector parameter $\theta_{\mathbf{r}}$ of the distribution $G(\theta)$ is computed, where $\theta_{\mathbf{r}} = [\mu, \sigma]$. Then, the optimal parameters $\theta_{\mathbf{r}}^*$ can be estimated using standard methods of differential calculus.

The parameter estimation is considered as a maximum-likelihood problem. However, the nature of the data make it difficult to guarantee the stability of the parameter estimation. To overcome this problem, each evidence can be considered as an incomplete estimation of the true parameters, where a function $Q(\theta_r, \theta_r^*)$ exists and represents the parameter estimated with incomplete data θ_r and the true parameter θ_r^* . Iteratively, it is feasible to estimate the value of θ_r^* from θ_r using the following expression for the Gaussian parameter estimation:

$$\theta_r^t = \left[\begin{array}{c} \rho\theta_{r1}^{(t-1)} + (1-\rho)x_{(t)} \\ \rho\theta_{r2}^{2(t-1)} + (1-\rho)(\theta_{r1}^{(t-1)} - x_{(t)})^2 \end{array} \right] \quad (11)$$

such that $\rho \in [0, 1]$. This is a better way to estimate the parameters of the distribution without the restriction of the loss of data.

The parameter estimation performed in the fourth case is suitable for the first and third cases, since it offers an economic way to compute the parameter distribution of the center of the circle in a closed situation and the real center, whenever partial information is presented.

4 Experimental Process

This section presents a set of experiments performed with the aim of showing the capabilities of the proposal and its behavior against other well-accepted measures in controlled and uncontrolled scenarios. The experimental process is divided in three parts. In the first part, the measure is tested under controlled situations, one hundred digital circles are drawn, in which parameters such as norm and radius are varied. Particular aspects such as resolution, occlusion, connectivity and amount of border information of the objects are addressed here. In the second part, the proposal is tested with other well-known measures – radius ratio and mean roundness [21] – under the L_2 norm (Euclidean distance). In the third part, real images of graphite nodules are used; the nodules are characterized by their circularity using our approach and recent well-known used approaches.

4.1 Synthetic Images

In this subsection, the results obtained in controlled situations are presented. They include the validation of the framework by evaluating the roundness measure when it is applied to various circles with different norms and radii. The test consists in applying the MOR measure over a set of ideal circles under different L_i norms. The radius is varied from 1 to 100 with increments of 1, whereas the norm corresponds to $L_{0.001}$, $L_{0.002}$, $L_{0.005}$, $L_{0.01}$, $L_{0.02}$, $L_{0.05}$, $L_{0.1}$, $L_{0.2}$, $L_{0.5}$, L_1 , L_2 , L_3 , L_4 , L_5 , and L_{10} . Results are illustrated in Fig. 2. As can be appreciated, the proposal obtains similar results for the different norms. The lowest measures correspond to norms $L_{1<}$, where low values imply loss of accuracy for the different drawn circles. Additionally, in the majority of cases the measure is located above 0.995 with circles of at least 15 pixels in radius (see next paragraph for a more detailed explanation of this fact). On the other hand, several norms with small radii are scored with values equal to 1. From this situation follows the sparseness process, which comprises those few data that do not provide enough evidence for locating the inflexion change over the pdfs conformed by the distance to the border. The independence of the framework over the norm used, makes it suitable to be employed with other induced norm topologies in which other forms such as squares, stars or even diamond shapes can be recognized with a simple change in the norm used.

Next, in Fig. 3, an emphasis is made of the results obtained with the L_2 norm. It can be observed that the MOR measure stabilizes around circles with 5-pixel radius. This means that the information in the border provides enough evidence to infer that the figure is a circle. This result is completely justified by the big numbers law [12], which in general terms states that enough evidence to model

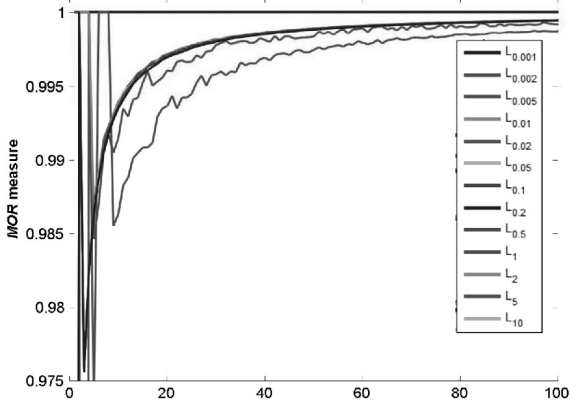


Fig. 2. Results obtained from different circles artificially generated with different norms. As can be appreciated, regardless of the norm used, the roundness measure converges faster for values nearer to one.

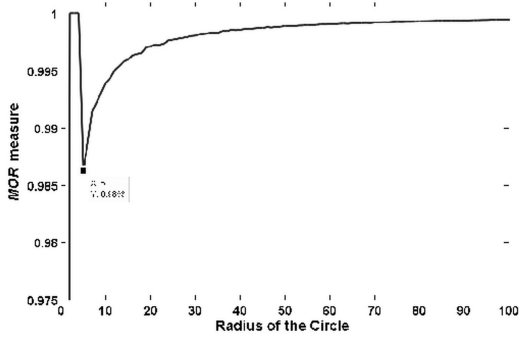


Fig. 3. Results for the particular case L_2 . The circles with small radii converge faster to one.

a normal distribution must consist of at least 30 elements. In our particular case, the perimeter of a 5 pixel radius is 31.415 pixels. Similarly, circles smaller than 5 pixels have a MOR measure approximately of 1 given by the limitations of locating the inflexion points over the pdf of the radius. The above results give the minimum criterion with respect to the resolution and the amount of information necessary to infer that the figure represents a circle and provide an issue to define a sampling process which reduces the computational complexity to estimate the measure discussed in the fourth case.

The following test consists in evaluating the accuracy in the recognition of a circle, even when there is a lack of information. This situation is common when the object is partially occluded or the quality of the acquisition is poor and the borders are not well-defined. The test consists in sampling an artificial circle using a percentage of the total of pixels that conform the border. The

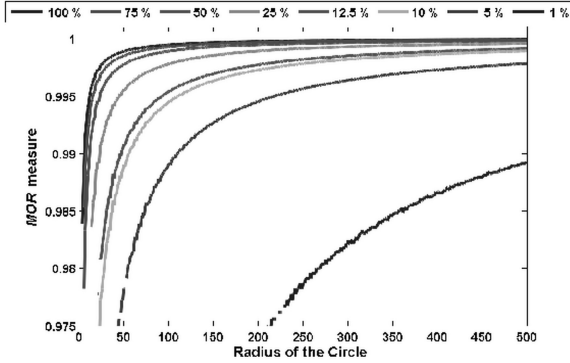


Fig. 4. Results obtained for different sampling levels. The degradation in the accuracy of the MOR measure is small, which confirms that it is robust, even when there is no considerable information on circle borders.

radius is varied and the circle is used under the L_2 norm. The total of the information used in the sampling is: 100%, 75%, 50%, 25%, 12.5%, 10%, 5% and 1%. The results are illustrated in Fig. 4. The sampling process entails a uniform sampling over the pixels that conform the border of the circle. The MOR measure is not significantly affected by the sampling process. When the sampling process uses only 10% or less of the information, there is a slight reduction in precision; however, in practice this represents extreme cases that are not likely to succeed. As can be appreciated, the proposal framework is robust even when the information of the object is partial or has been lost.

In conclusion, the above test shows that the framework is suitable to measure the roundness in various scenarios. In the following paragraphs we show the results on real scenarios, as well as the comparison of our proposal with other well accepted measures.

4.2 Comparison with Other Measures in Synthetic Images

In an attempt to validate the proposal, in this study, our measure is compared with two other measures: the radius ratio (RR) and the mean roundness (MR).

$$RR = \frac{rb_{\min}}{rb_{\max}}, \quad (12)$$

where rb_{\min} is the minimum radius from a border point to the center of the border and rb_{\max} is the maximum radius. The main disadvantage of this measure is that the proportions of the shortest and longest radius do not provide sufficient information to characterize the roundness of the object, and the measure can be affected by pixel aberrations as well.

The second measure is based on the theory of mean deviation (MD), it calculates the sum of the absolute differences between the radius of each border pixel and the average radius. This measure is defined as:

$$MR = \frac{1}{n} \sum \frac{\bar{r}_b}{|r_j - \bar{r}_b| + \bar{r}_b}, \quad (13)$$

where \bar{r}_b is the average radius from the border points to the center of the object, and r_j is the radius of border point j to the center of the border. The center of the object is usually expressed as the expected point from all points that conform the border. This expression is valid only if the expected value is equal to the mean. Therefore, the average is highly correlated for closed to round objects. Consequently, this measure is not reliable when assessing the roundness of irregular and partial shapes.

The results of the comparison are shown in Fig. 5. From this graph, it can be seen that the proposed measure converges quickly for values near to one. On the other hand, MOR and MR are equivalent when the resolution becomes higher. Conversely, the RR measure yields significant smaller results. This means that it is less robust to resolution changes. Moreover, the main disadvantage of this measure is that the proportions of the shortest and longest radii do not provide sufficient information to measure the roundness and the measure could be affected by pixel aberrations. The MR measure becomes numerically similar to the proposal; however, these results were obtained from regular shapes, but when there is a partial or not connected shape (for instance an arc), it is not reliable for measuring the roundness, because it is not possible to define the \bar{r}_b radius. Furthermore, the mean does not always represent a good estimator for characterizing the roundness of an object, especially when the objects to be measured are partially occluded or the borders are not complete. This fact becomes important when measuring the roundness of incomplete particles.

Commonly, several measures discuss the resolution as an invariance property of each of them; however, as it was commented above, in Fig. 11, the shape and the topology of the circle are related to the resolution and therefore unsuitable for several measures that are considered resolution invariant.

Next, one hundred polygons were generated, the number of sides ranging from 3 to 100. Figure 6 shows the measures obtained for regular polygons. As expected, the circularity increases with the number of sides and converges toward 1. The larger the number of sides, the more the polygons resemble a circle and the more the circularity approaches 1. Now, the RR measure is quite similar to MOR whenever the polygons are regular; instead MR results less robust than the other two. Hence, MR results more robust to resolution changes, but RR is better to characterize regular polygons. However, our proposal appears always equivalent to the best measure in the distinct tested scenarios.

4.3 Application to a Real Case

Finally, to show the reliability of the proposal, we have selected a real application oriented to compound materials. The application corresponds to the analysis of

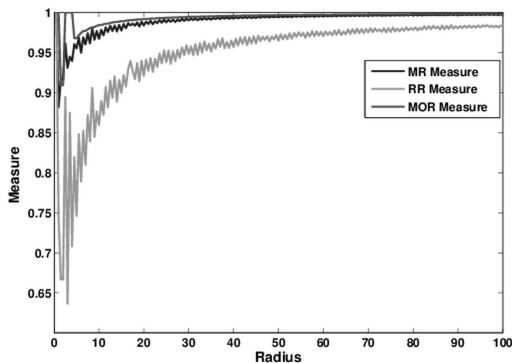


Fig. 5. Results obtained of different circles artificially generated with different measures

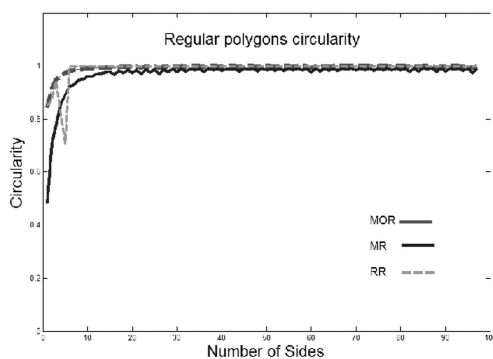


Fig. 6. Regular polygone circularity

nodule graphite circularity in images. The proposed approach is used under the L_2 norm (Euclidean distance), and for comparative purposes the two measures aforementioned are used. In Fig. 7, a typical image with graphite nodules is shown. Note that the nodules come with different shapes close to circular forms. In order to test our approach, we develop an experimental analysis in two stages: In the first scenario, the order of each measure is defined and analyzed over 10 distinctive graphite nodules (see Fig. 8). In a second scenario, 110 graphite particles are evaluated in a graphite specimen and the range of distribution for each measure is computed.

In the first stage, from a population of 10 typical shapes encountered in the nodules, the circularity measure of each shape is estimated. The image in Fig. 8 shows the distinctive graphite nodules; Table 1 displays the results for each measure. Note that all measures have a distinguishable difference, especially when the shape of the nodules is irregular. To illustrate this, observe the first element in the table, the minimum value of the roundness measure MOR is 0.4437, and those computed for MR and RR are 0.7563 and 0.0471, respectively.

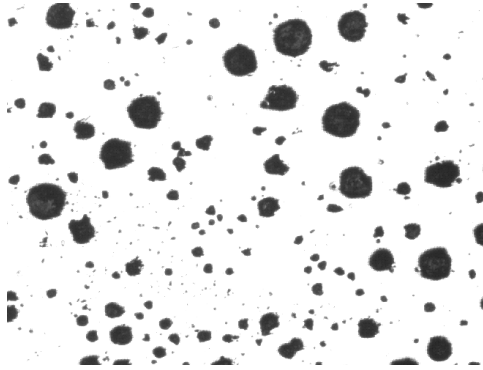


Fig. 7. Graphite specimen

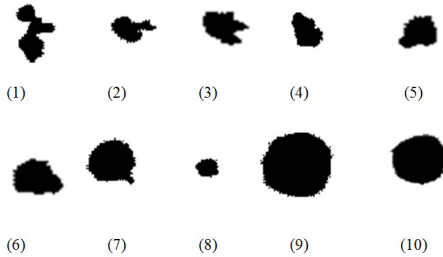
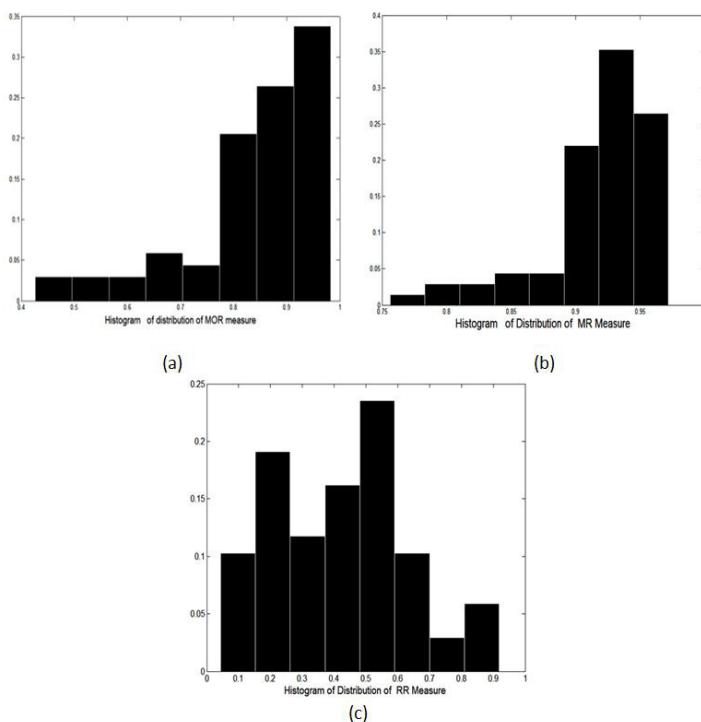


Fig. 8. Different graphite nodules

On the other hand, when the nodule shape tends to be more circular, MOR and MR show similar values; this can be seen in samples 9 and 10 of Table II. Analyzing these results with MR, the interval between the least roundness and the most roundness is small, approximately 0.2130; consequently, it cannot distinguish the effects of roundness. Alternatively, taking our proposed measure and RR, the interval is about 0.5314 and 0.8372, respectively, which means that it is possible to discriminate between the least circular nodule and the most circular one. On the other hand, to demonstrate the consistency of the measures when compared with human perception, the 10 nodules were evaluated by 50 people. In this part, each person accommodated them in order from the most circular shape to the least circular. The comparisons carried out between the participants and the measures obtained show that MOR and MR give a good match when human perception is utilized. A visual comparison evinces that not all the selected graphite particles are close to a circle, that is, some nodules cannot be considered circular. As a result, the MR measure is less accurate in characterizing the circularity of a graphite sample, especially when the shape is markedly irregular. Finally, as can be appreciated, our proposal maintains a relative order of the shapes, being more robust for irregular objects, unlike the other presented measures.

Table 1. Values of roundness estimated for 10 segmented nodules illustrated in Fig. 14

Nodule Number	MOR	MR	RR
1	0.4737	0.7563	0.0471
2	0.5520	0.7615	0.1180
3	0.5520	0.6315	0.5381
4	0.5934	0.7748	0.4588
5	0.6330	0.7803	0.6147
6	0.6390	0.7765	0.5047
7	0.7361	0.9424	0.6315
8	0.9654	0.9358	0.6852
9	0.9851	0.9661	0.8216
10	0.9751	0.9693	0.8847

**Fig. 9.** Histograms for the different measures

Finally, as an additional test, the sparseness of the measure is analyzed for each measure used for nodules in the image (Fig. 7). Using all nodules, the distribution of each measure is estimated. Observe in Fig. 9 that the MOR measure has the sparsest range. On the other hand, for MR and RR, the dispersion is less sparse. In metric spaces, sparse distribution is better because the probability to generate classes is higher. Then, when using MR and RR measures, it becomes

more complicated to distinguish the different classes of nodules. As aforementioned, the MR measure is based on the mean operator and MOR is based on the mode. This difference causes the outliers to affect the mean operator, whereas the mode becomes more robust, especially when outliers do not follow a normal distribution. It is noteworthy to emphasize that the proposed measure (MOR) is a generalized measure. Therefore, it satisfies the following properties: (1) it ranges within $([0, 1])$, where 1 is scored only by a perfect circle, (2) it is invariant with respect to resolution, so the measure becomes independent of the equipment, (3) it is tolerant to shape variations, (4) it is tolerant to noise or narrow intrusions, and (5) it can be easily compared to human perception. Hence, the proposed measure can be adopted as an alternative to measure the roundness of graphite nodules.

5 Conclusion

In this work, a new framework to measure the roundness is presented. This framework is based on the central concept of circle in a two dimensional space under certain induced norm. This framework results useful, because the roundness is conceived by its properties instead of by the topology of a particular space, which allows to match order and to measure the roundness of the objects in non-Euclidean spaces. The particular case of a two dimensional case is analyzed. Here, the proposed measure is matched with two of the most well accepted measures. The results demonstrate that the proposed measure is better behaved with regard to sparseness and measures more robustly the roundness of the objects under the tested scenario. As a result, we provide a reliable framework to deal with the task of measuring circularity.

References

1. Bottema, M.J.: Circularity of objects in images. In: International Conference on Acoustic, Speech and Signal Processing. ICASSP, Istanbul, vol. 4, pp. 2247–2250 (2000)
2. Coeurjolly, D., Gérard, Y., Reveilles, J.P., Tougne, L.: An elementary algorithm for digital arc segmentation. *Discrete Applied Mathematics* 139(1-3), 31–50 (2004)
3. Cox, E.P.: A method of assigning numerical and percentage values to the degree of roundness of sand grains. *Journal of Paleontology* 1(3), 179–183 (1927)
4. Chapman, S.B., Rowe, R.C., Newton, J.M.: Characterization of the sphericity of particles by the one plane critical stability. *Journal of Pharmacy and Pharmacology* 40(7), 503–505 (1988)
5. Damaschke, P.: The linear time recognition of digital arcs. *Pattern Recognition Letters* 16(5), 543–548 (1995)
6. Dasgupta, A., Lahiri, P.: Digital indicators for red cell disorder. *Current Science* 78(10), 1250–1255 (2000)
7. Davies, E.R.: A modified Hough scheme for general circle location. *Pattern Recognition Letters* 7(1), 37–43 (1987)
8. Di Ruperto, C., Dempster, A.: Circularity measures based on mathematical morphology. *Electronics Letters* 36(20), 1691–1693 (2000)

9. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. *Communications of the Association of Computing Machinery* 15, 587–598 (1972)
10. Fisk, S.: Separating point sets by circles, and the recognition of digital disks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(4), 554–556 (1986)
11. Frosio, I., Borghese, N.A.: Real time accurate vectorization circle fitting with occlusions. *Pattern Recognition* 41, 890–904 (2008)
12. Grimmett, G.R., Stirzaker, D.R.: *Probability and Random Processes*, vol. 2. Clarendon Press (1992)
13. Haralick, R.M.: A measure for circularity of digital figures. *IEEE Transactions on Systems, Man and Cybernetics* 4(4), 394–396 (1974)
14. Hilaire, X., Tombre, K.: Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(6), 217–223 (2006)
15. Kim, C.E., Anderson, T.A.: Digital disks and a digital compactness measure. In: *Annual ACM Symposium on Theory of Computing, Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, New York, USA, pp. 117–124 (1984)
16. Li, J., Lu, L., Lai, M.O.: Quantitative analysis of the irregularity of graphite nodules in cast iron. *Materials Characterization* 45, 83–88 (2000)
17. Lilliefors, H.: On the Kolmogorow-Smirnow test for normality with mean and variance unknown. *Journal of the American Statistical Association* 62, 399–402 (1967)
18. O’Rourke, J., Kosaraju, S.R., Meggido, N.: Computing circular separability. *Discrete and Computational Geometry* 1, 105–113 (1986)
19. Peura, M., Livarinen, J.: Efficiency of simple shape descriptors. In: *3rd International Workshop on Visual Form*, Capri, Italy, pp. 28–30 (1997)
20. Pegna, J., Guo, C.: Computational metrology of the circle. In: *Proceedings of IEEE Computer Graphics International*, pp. 350–363 (1998)
21. Ritter, N., Cooper, J.R.: New resolution independent measures of circularity. *Journal of Mathematical Imaging and Visio* 35(2), 117–127 (2009)
22. Rosin, P.L.: Measuring shape: ellipticity, rectangularity, and triangularity. *Machine Vision and Applications* 14(3), 172–184 (2003)
23. Sauer, P.: On the recognition of digital circles in linear time. *Computational Geometry: Theory and Application* 2(5), 287–302 (1993)
24. Whalley, W.B.: The description and measurement of sedimentary particles and the concept of form. *Journal of Sedimentary Petrology* 42(4), 961–965 (1972)
25. Worring, M., Smeulders, A.W.: Digitized circular arcs: characterization and parameter estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(6), 554–556 (1995)
26. Yip, R.K.K., Tam, P.K.S., Leung, D.N.K.: Modification of Hough transform for circles and ellipses detection using a 2-dimensional array. *Pattern Recognition* 25(9), 1007–1022 (1992)

A New Framework for Connected Components Labeling of Binary Images

Tetsuo Asano¹ and Sergey Bereg²

¹ Japan Advanced Institute of Science and Technology (JAIST)
Ishikawa 923-1292, Japan

t-asano@jaist.ac.jp

² Department of Computer Science, University of Texas at Dallas, USA
besp@utdallas.edu

Abstract. Given a binary image of n pixels, assign integral labels to all pixels so that any background pixel has label 0 and any two foreground pixels have the same positive integral labels, if and only if they belong to the same connected components. This problem is referred to as 'Connected Components Labeling' and it is one of the most fundamental problems in image processing and analysis. This paper presents a new algorithmic framework for the problem. From an algorithmic point of view, the problem can be solved in $O(n)$ time and $O(n)$ space. We propose new algorithms which use smaller work space without much sacrifice of the running time. More specifically, assuming that an input binary image is given by a read-only array, our algorithm outputs correct labels in the raster order in $O(n \log n)$ time using only $O(\sqrt{n})$ work space. Some applications of the algorithms are also given.

Keywords: Connected component, Limited work space, Computational complexity.

1 Introduction

There is increasing demand for highly functional consumer electronics such as printers, scanners, and digital cameras. To achieve this functionality they need sophisticated embedded software. One fundamental difference from software used in conventional computers is that there is little allowance of work space which can be used by the software. In this paper we propose several space-efficient algorithms designed for some fundamental image processing tasks. All those tasks are indeed easy, if sufficient memory space is available. In this paper we are interested in solving the same tasks using only restricted memory while keeping efficiency of the algorithms.

Computational Model with Limited Work Space

The space complexity of an algorithm is measured by the amount of work space used in addition to a read-only array to store an input binary image of n pixels. Such work space is composed of pointers and counters. The size of such a pointer

or a counter is $O(\log n)$ bits. In this paper we consider only those algorithms using $O(1)$ or $O(\sqrt{n})$ pointers and counters.

Throughout the paper we assume that an input binary image consists of $O(\sqrt{n})$ rows and $O(\sqrt{n})$ columns. We also assume a read-only array keeping an input binary image with random access to pixels.

Why Read-only Array?

Why do we assume a read-only array for an input binary image? The main reason is to exclude the possibility of a sophisticated mechanism to embed some information in an input array. Such a technique is used in implicit and succinct data structures (see, e.g., [14]).

Another reason is to save memory space.

It is known that predicates can be used to transform images and save memory. For example, suppose that we are interested in extracting an object region in a given color image \mathcal{C} of n pixels and we know some color information characterizing the object. Then, using this information we can define a predicate f which determines in constant time whether a pixel value (color) belongs to the object. The pair (\mathcal{C}, f) defines a binary image. In our settings, we can expect that the largest connected component in the binary image corresponds to the object region. To save space, we do not create the binary image since any pixel value can be computed in constant time. This is equivalent to assuming that a binary image is stored in a read-only array.



Fig. 1. Input color picture (a) and its largest component (b)

Fig. 1 shows such an example. The target object in Fig. 1(a) is a bridge which is characterized by color intervals, say Red $[80, 255]$, Green $[0, 80]$, and Blue $[0, 255]$. Fig. 1(b) shows the largest component.

Connected Components Labeling

In this paper we propose a new algorithmic framework for the problem known as connected components labeling in which integral labels are assigned to all pixels so that any background pixel has label 0 and any two foreground pixels have the same positive integral label, if and only if they belong to the same connected component.

Known Results on Connected Components Labeling

A number of algorithms have been proposed so far under several different computational models including pipelining and parallel computation [1, 5, 7–11, 15–19]. A nice survey [12] on this topics is also available. Since we may have $O(n)$ connected components for a binary image of n pixels, the whole label matrix takes $O(n \log n)$ bits in total. Our idea to save the work space is not to keep such a label matrix but just to report labels in a fixed order, say in the raster order without using any array.

Fig. 2 shows an example of a binary image. The right figure is the result of labeling using different colors for different components. Pixels in a connected component have the same label while any two pixels from different components have different labels. Colors are used for labels in the figure, but in fact they are integers $1, 2, \dots, c$ for c connected components.

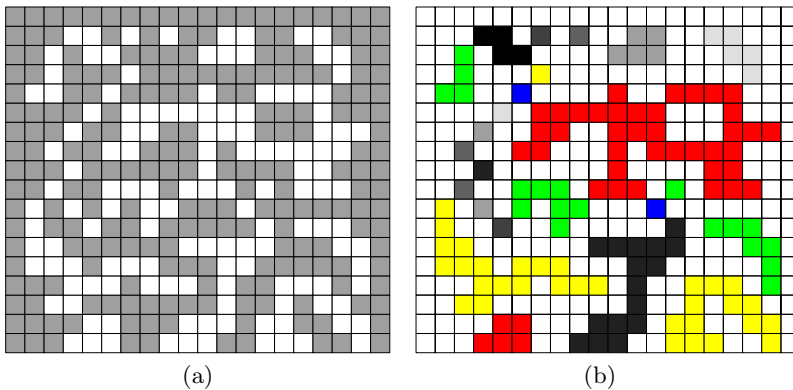


Fig. 2. Labeling a binary image. (a) input binary image, and (b) resulting labels of different colors. Note that black pixels are colored white in (b).

Results Obtained

A great number of algorithms have been presented for the connected components labeling, but none of them works in sublinear work space since they keep a label matrix in the algorithm. Since $O(n)$ different labels may be needed for a binary image of n pixels, we need $O(n)$ space for the matrix. To save the space we keep only two rows of the label matrix. Our algorithm outputs a labeling matrix, a result of connected components labeling, in raster order in $O(n \log n)$ time using only $O(\sqrt{n})$ work space.

The algorithm can be easily adapted to solve the following interesting problem: given a binary image and a constant $k > 0$, computes a binary image which contains only k largest connected components in the input image. The resulting binary image should be output row by row in the raster order without keeping the whole output image in the work space.

Novelty of Our Approach

Designing algorithms using limited work space has been studied for many years under the name of “log-space algorithms” in the complexity theory. A main concern for log-space algorithms has been of theoretical interest in a class of problems which can be solved in polynomial time using only constant work space for an input of size n in addition to a read-only array of size $O(n)$ to keep n input data. However, the constraint of the work space to $O(1)$ may be too severe for practical applications. It is quite natural to work on an image of n pixels to use some constant number of rows and columns as work space, which amounts to $O(\sqrt{n})$ for an image of size $O(\sqrt{n}) \times O(\sqrt{n})$.

From a theoretical point of view, it is rather easy to design a linear-time algorithm for connected components labeling, if we are allowed to use linear work space. A folklore algorithm using wave propagation is one such example. In this paper we propose a completely different framework for the problem. Our main purpose here is to reduce the amount of work space assuming that an input binary image is given on a read-only array. Our algorithm outputs a label matrix row by row in the raster order just using two rows of the matrix (and thus $O(\sqrt{n})$ work space for an image of n pixels) and runs in $O(n \log n)$ time. The running time depends on complexity of an input image. If it is simple enough, it runs in linear time.

2 Preliminary: Basic Assumptions and Definitions

A binary image of n pixels of size $O(\sqrt{n}) \times O(\sqrt{n})$ is given on a read-only array. Pixels are sequentially indexed from 0 to $n - 1$ in the raster order (from left to right and bottom to top). We embed the input image into a larger array filled with 0 on the four sides. When we denote the width and height of the extended image array by w and h , respectively, pixels in the array are indexed from 0 to $wh - 1 = O(n)$. A pixel is considered as a square with four sides. The top and the right sides (edges) of a pixel of index k are indexed with $2k$ and $2k + 1$, respectively. Thus, edges are also uniquely indexed using $O(\log n)$ bits.

Fig. 3 shows how pixels and edges are indexed. Pixels are all indexed, but some edges are not. For example, edges on the left and bottom sides of the extended image are not indexed. Since we are only interested in original input pixels which lie inside and their associated edges, those boundary edges are never accessed (or cause no problem).

Here we summarize basic definitions and terminologies (see Fig. 4).

Connectivity: Two foreground pixels are 4-connected (resp. 8-connected), if they are adjacent horizontally or vertically (resp., one of them lies in the 3×3 neighborhood of the other). We also say that two foreground pixels are 4-connected (resp. 8-connected), if there is a path of foreground pixels such that any two consecutive pixels are 4-connected (resp. 8-connected).

Connected Component: A maximal subset of foreground pixels such that any two of them are connected.

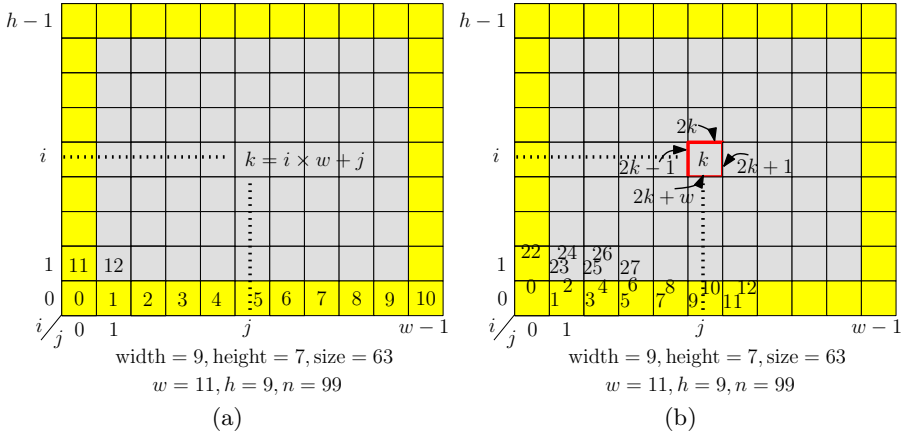


Fig. 3. Sequential indices of pixels in the raster order (a) and edge indices (b). A binary image of size 7×9 is embedded in a larger image of size 9×11 by filling background pixels in the margin.

Hole: A connected component of background pixels contained in a connected component.

Connectivity for Holes: Use alternate connectivities for foreground pixels and background ones. If connectivity for foreground pixels is 4-connectivity then that for background ones must be 8-connectivity.

Island: A connected component of foreground pixels contained in a hole.

External Boundary: A unique boundary of a connected component (of foreground pixels) with background pixels surrounding them.

Internal Boundary: Boundaries of a connected component with its holes.

Raster Order of Pixels: A pixel p_1 precedes another pixel p_2 in the raster order, if p_1 is lower than p_2 or they lie in the same row and p_1 lies to the left of p_2 .

Raster Order of Edges: A vertical edge e_1 is smaller than another vertical edge e_2 , if e_1 is lower than e_2 or they lie in the same row, but e_1 lies to the left of e_2 . We are not interested in ordering horizontal edges.

Orientation of Edges: Edges on a boundary are oriented so that foreground pixels always lie to the right. Thus, external boundaries are clockwise ordered while internal ones counterclockwise.

Canonical Edge [13]: The vertical edge of a boundary (external or internal) that is smallest in raster order. By our convention on orientation canonical edges of external (resp. internal) boundaries are upward (resp. downward).

NextEdge(e): Given any edge on a boundary, we can compute the next edge on the boundary in constant time (see Fig. 5). So, we assume a function NextEdge(e) to compute the next edge. The function depends on which connectivity is used to define a connected component of foreground pixels,

but it is an easy exercise to design such a function, and thus omitted here. (see Fig. 5(b))

- PrevEdge(e):** A function to compute the previous edge of e on a boundary.
- Run:** A maximal sequence of foreground pixels in a row.
- Lowest Run:** A run which is not adjacent to any run in its lower row.
- Labeling:** Assigning positive integral labels to pixels so that two pixels have the same label, if and only if they belong to the same connected component.
- Canonical Labeling:** A labeling of a binary image by sequential integral numbers in the raster order of their canonical edges.

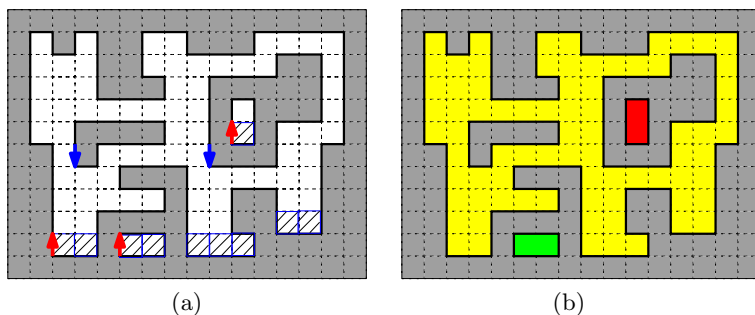


Fig. 4. An example of a binary image consisting of three connected components. The largest component contains two holes and the right hole contains an island. Each of five boundaries has a unique canonical edge indicated by an arrow. (a) Lowest runs are indicated by shading. (b) A result of labeling the image using different colors.

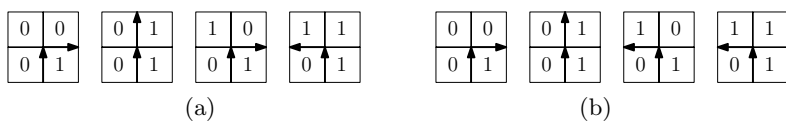


Fig. 5. The next edge of a vertical edge. (a) for 4-connectivity, and (b) for 8-connectivity.

3 $O(n)$ -Space and $O(n)$ -Time Algorithm

To explain our basic idea, we will first present a linear-time algorithm using linear work space. We use two linear-size arrays, one for a label matrix and the other for keeping labels on boundary edges.

We scan a given binary image in the raster order. Whenever we encounter an edge which has not been labeled yet, we create a new label and follow the boundary from the edge while putting the label on the array for edge labels. If two foreground pixels are consecutive in a row, we just propagate a label from left to right.

This simple algorithm works for a binary image without a hole. To deal with holes, we also follow a boundary whenever we have a transition from foreground to background (transition from '1' to '0'). If an edge associated with the transition has not been labeled, we first obtain a label from the pixel to the left of the edge (it must have been labeled) and then follow the boundary while putting the label on the edge array.

By the definition of canonical edges, for any boundary (external or internal) the first edge we encounter in the scan must be its canonical edge (see Fig. 6). If we keep a label of a component, such an edge can be detected when we encounter an edge which has not been labeled yet. Note that the canonical edge of an external boundary of a connected component must have been detected earlier than canonical edges of internal boundaries of the component. So, if we put a label when we find the canonical edge of the external boundary and put the new label on those edges on the boundary and propagate the label to the right, the label must reach the canonical edges of hole(s) in the component before we detect the edges by transitions '10'.

The running time of Algorithm 1 is $O(n)$ since we follow each boundary only once and the remaining operations are obviously done in linear time.

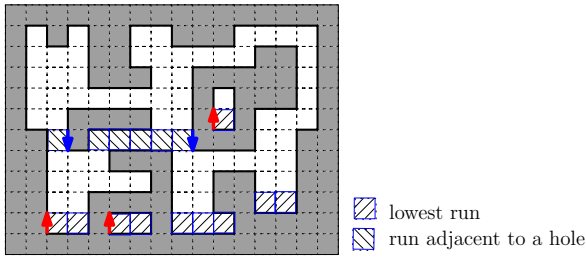


Fig. 6. Lowest runs and runs adjacent to holes

Lemma 1. *Given a binary image of n pixels, Algorithm 1 computes its correct canonical labeling in $O(n)$ time and $O(n)$ work space.*

4 $O(\sqrt{n})$ -Space and $O(n\sqrt{n})$ -Time Algorithm

Our main concern in this paper is to reduce space complexity while suppressing increase of time complexity. In Algorithm 1 we used two arrays of linear size. It is easy to reduce the one for a label matrix into $O(\sqrt{n})$. Recall that Algorithm 1 outputs the label matrix row by row. Thus, we do not need to keep the whole array, but one or two rows are enough.

How about the array to store labels in edges? Labels stored in edges are needed to propagate labels along (external and internal) boundaries. Is it possible to get

Algorithm 1. Compute the label matrix using $O(n)$ space

```

Input: A binary image  $B[i], i = 0, \dots, wh - 1$ .
Arrays:  $lab[0..wh - 1], edge[0..2wh - 1]$ .
 $L = 0$ ; // the number of labels used so far
Initialize arrays  $lab[.]$  and  $edge[.]$ 
for  $i = 1$  to  $h - 1$  do
  for  $j = 1$  to  $w - 1$  do // pixel in the raster order
     $p = i \times w + j$ ; // pixel at  $(i, j)$ .
    switch the value of  $B[p - 1]B[p]$  do
      case '00' // background pixel  $p$ 
      |  $lab[p] = 0$ ; break;
      case '11' // next pixel of the run
      |  $lab[p] = lab[p - 1]$ ; break;
      case '01' // boundary pixel  $p$ 
      | if  $edge[2p - 1] = 0$  then // new component
      | |  $L = L + 1$ ; // new label
      | |  $e_s = 2p - 1$ ; //  $e_s$ : start edge
      | |  $e = e_s$ ; //  $e$ : current edge  $e$ 
      | | repeat // traverse the boundary of component  $L$ 
      | | |  $edge[e] = L; e = NextEdge(e)$ ;
      | | until  $e = e_s$ ;
      | end
      | else // old component labeled  $edge[2p - 1]$ 
      | |  $lab[p] = edge[2p - 1]$ ;
      | end
      | break;
      otherwise // case '10', pixel  $p - 1$  is on a boundary
      |  $lab[p] = 0$ .
      | if  $edge[2p - 1] = 0$  then // unlabeled boundary
      | |  $e_s = e = 2p - 1$ ; // start edge  $e_s$  and current edge  $e$ 
      | | repeat // traverse the unlabeled boundary
      | | |  $edge[e] = lab[p - 1]; e = NextEdge(e)$ ;
      | | until  $e = e_s$ ;
      | end
    endsw
  endsw
end
end
Output the array  $lab[.]$ 

```

the label information without using the array? Yes, it is possible. Our idea is very simple. Whenever we encounter a boundary edge we follow the boundary while checking pixels along the boundary whether they are labeled or not. Since we examine pixels in the raster order, if we start from any edge which is not the canonical edge of an external boundary we must reach a boundary edge which

is smaller in the raster order than the starting edge. In our algorithm we create a new label when we find the canonical edge of the external boundary of a new component. If the current boundary edge e is not canonical, then we follow the boundary and get a label, which is the correct label of the pixel adjacent to e . We can determine whether an edge e is canonical or not just by following the boundary from e to see whether we see any edge $e' < e$.

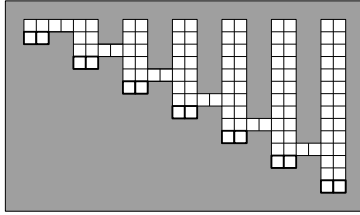


Fig. 7. Worst case example for which Algorithms 2 takes $\Omega(n\sqrt{n})$

Lemma 2. *Given a binary image of n pixels, Algorithm 2 computes its correct canonical labeling in $O(n\sqrt{n})$ time using $O(\sqrt{n})$ work space. There is an example for which the algorithm takes $\Omega(n\sqrt{n})$ time.*

Proof. We follow boundaries whenever we encounter an edge associated with a transition '01'. There may be $O(\sqrt{n})$ such edges in a row and each boundary may consist of $O(n)$ edges. However, we never follow the same boundary more than once since we can stop whenever we come to an edge which is smaller in the raster order than the starting edge. Thus, the total length of boundaries we follow in a row is bounded by $O(n)$. Since we have $O(\sqrt{n})$ rows, the total running time is bounded by $O(n\sqrt{n})$.

The correctness of the algorithm can be proved by induction.

There is an example for which the algorithm takes $\Omega(n\sqrt{n})$ time. See Fig. 7 (a) for an worst case example. The binary image contains only one connected component, which has $O(\sqrt{n})$ lowest runs depicted by bold lines. Whenever we encounter the left edge of each such lowest run we follow the boundary in a clockwise order. For a lowest run starting at $x = k$, the length of our traverse is $O(k^2)$ by the construction. Thus, it takes time $O(\sum_{k=1}^{O(\sqrt{n})} k^2) = O(n\sqrt{n})$.

5 $O(\sqrt{n})$ -Space and $O(n \log n)$ -Time Algorithm

It is surprisingly easy to accelerate Algorithm 2 while keeping the size of the work space. See the worst case example shown in Fig. 7 (a). It is easy to see that it took much time since we follow the boundary clockwise manner. If we follow it counterclockwise instead, then we immediately find a lower edge and thus the total running time concerning the boundary is just linear. This suggests a bidirectional search for a lower boundary edge. Bose and Morin [6] achieved

Algorithm 2. Compute the label matrix using $O(\sqrt{n})$ space.

```

Input: A binary image  $B[i], i = 0, \dots, wh - 1$ .
Array:  $\text{lab}[2][0..w - 1]$ . // label array for two rows
 $L = 0$ ; // the number of labels used so far
Initialize the array  $\text{lab}[2][\cdot], a = 0, b = 1$ 
for  $i = 1$  to  $h - 1$  do
    // the labels in the previous row are computed in  $\text{lab}[b][\cdot]$ 
    for  $j = 1$  to  $w - 1$  do // pixel in the raster order
         $p = i \times w + j$ ; // pixel at  $(i, j)$ .
        switch the value of  $B[p - 1]B[p]$  do
            case '00' // empty pixel  $p$ 
                |  $\text{lab}[a][p] = 0$ ; break;
            case '11' // next pixel of the run
                |  $\text{lab}[a][p] = \text{lab}[a][p - 1]$ ; break;
            case '01' // boundary pixel  $p$ 
                |  $e_s = e = 2p - 1$ ; // start edge  $e_s$  and current edge  $e$ 
                | repeat // traverse the boundary from  $e_s$ 
                    | |  $e = \text{NextEdge}(e)$ ;
                    | until  $e = e_s$  or  $e$  is adjacent to a pixel with a positive label  $k$  in
                    | | the row  $i$  (by  $\text{lab}[a][\cdot]$ ) or  $i - 1$  (by  $\text{lab}[b][\cdot]$ );
                    | | if  $e = e_s$  then // new component
                        | | |  $L = L + 1, k = L$ ; // new label
                    | | end
                    | |  $\text{lab}[a][p] = k$ ; break;
                | otherwise // case '10',
                    | |  $\text{lab}[a][p] = 0$ ;
                | endsw
        endsw
    end
    Output the content of the array  $\text{lab}[a][\cdot]$ .
    Exchange the roles of  $a$  and  $b$  ( $a = 1 - a, b = 1 - b$ ).
end

```

$O(n \log n)$ time using the bidirectional traverse on a planar subdivision. More exactly, at each boundary edge we examine the boundary in opposite directions using the two functions, $\text{NextEdge}()$ and $\text{PrevEdge}()$, to find the next and previous edges on the boundary. If the starting edge e is not canonical then one of the pointers reaches an edge which is smaller in the raster order than e , which must be adjacent to a labeled pixel. Otherwise the two pointers meet without finding such pixels. Bose and Morin [6] showed that the bidirectional search runs in $O(n \log n)$ time without using any extra array. Thus, the algorithm is described as follows.

Theorem 1. Given a binary image of n pixels, Algorithm 3 computes the canonical labeling in $O(n \log n)$ time using $O(\sqrt{n})$ work space. There is an example for which the algorithm takes $\Omega(n \log n)$ time.

Algorithm 3. Compute the label matrix using bidirectional search

```

Input: A binary image  $B[i], i = 0, \dots, wh - 1$ .
Array:  $\text{lab}[2][0..w - 1]$ . // label arrays for two rows
 $L = 0$ ; // the number of labels used so far
Initialize the array  $\text{lab}[2][\cdot], a = 0, b = 1$ 
for  $i = 1$  to  $h - 1$  do
    // the labels in the previous row are computed in  $\text{lab}[b][\cdot]$ 
    for  $j = 1$  to  $w - 1$  do // pixel in the raster order
         $p = i \times w + j$ ; // pixel at  $(i, j)$ .
        switch the value of  $B[p - 1]B[p]$  do
            case '00' // empty pixel  $p$ 
            |  $\text{lab}[a][p] = 0$ ; break;
            case '11' // next pixel of the run
            |  $\text{lab}[a][p] = \text{lab}[a][p - 1]$ ; break;
            case '01' // boundary pixel  $p$ 
            |  $e_f = e_b = 2p - 1$ ; // two pointers in two directions
            |  $d = 0$ ; // search direction: 0 for forward
            repeat // traverse the boundary in two directions
            | if  $d = 0$  then  $e_f = \text{NextEdge}(e_f)$  else  $e_b = \text{PrevEdge}(e_b)$ 
            |  $d = 1 - d$ .
            until  $e_f = e_b$  or  $e_f$  or  $e_b$  is adjacent to a pixel with a positive
            label  $k$  in the row  $i$  or  $i - 1$ ;
            if  $e_f = e_b$  then // new component
            |  $L = L + 1, k = L$ ; // new label
            end
             $\text{lab}[a][p] = k$ ; break;
            otherwise // case '10',
            |  $\text{lab}[a][p] = 0$ ;
            endsw
        endsw
    end
    Output the array  $\text{lab}[a][\cdot]$ .
    Exchange the roles of  $a$  and  $b$  ( $a = 1 - a, b = 1 - b$ ).
end

```

6 Some Applications

The algorithm can be easily adapted to solve the following interesting problem: given a binary image and a constant $k > 0$, computes a binary image which contains only k largest connected components in the input image. It is done by using a priority queue of size k . It is done by using a priority queue of size k . It suffices to update, if necessary, the content of the queue whenever a connected component finishes. The resulting binary image should be output row by row in the raster order without keeping the whole output image in the work space.

Here is another example. Suppose we have a gray-tone image \mathcal{G} and we have a target size A of an object area. When we apply a threshold T to \mathcal{G} , we have a binary image. The size of the largest connected component in the resulting binary image is anti-proportional to the threshold. Problem here is to find an

Algorithm 4. Compute k-largest connected components

```

Input: A binary image  $B[i], i = 0, \dots, wh - 1$ .
Array: lab[2][0..w - 1]. // label arrays for two rows
 $L = 0$ ; // the number of labels used so far
Initialize the array lab[2][.],  $a = 0, b = 1$ 
for  $i = 1$  to  $h - 1$  do
    // the labels in the previous row are computed in lab[b][.]
    for  $j = 1$  to  $w - 1$  do // pixel in the raster order
         $p = i \times w + j$ ; // pixel at  $(i, j)$ .
        switch the value of  $B[p - 1]B[p]$  do
            case '00' // empty pixel  $p$ 
            | lab[a][p] = 0; break;
            case '11' // next pixel of the run
            | lab[a][p] = lab[a][p - 1]; break;
            case '01' // boundary pixel  $p$ 
            |  $e_f = e_b = 2p - 1$ ; // two pointers in two directions
            |  $d = 0$ ; // search direction: 0 for forward
            repeat // traverse the boundary in two directions
            | if  $d = 0$  then  $e_f = \text{NextEdge}(e_f)$  else  $e_b = \text{PrevEdge}(e_b)$ 
            |  $d = 1 - d$ .
            until  $e_f = e_b$  or  $e_f$  or  $e_b$  is adjacent to a pixel with a positive
            label  $k$  in the row  $i$  or  $i - 1$ ;
            if  $e_f = e_b$  then // new component
            |  $L = L + 1, k = L$ ; // new label
            end
            lab[a][p] =  $k$ ; break;
            otherwise // case '10',
            | lab[a][p] = 0;
            endsw
        endsw
    end
    Output the array lab[a][.].
    Exchange the roles of  $a$  and  $b$  ( $a = 1 - a, b = 1 - b$ ).
end

```

optimal threshold so that the largest connected component size is closest to the target size A . This problem can be solved by the algorithm below in $O(n \log^2 n)$ time using only $O(\sqrt{n})$ space.

7 Conclusions

In this paper we have presented a space efficient algorithm for connected components labeling. It runs in $O(n \log n)$ time using only $O(\sqrt{n})$ work space assuming that an input image of n pixels is stored in a read-only array. It is simple and easy to implement. An open problem is to prove the algorithm is theoretically optimal in the sense that $\Omega(n \log n)$ is the lower bound of any algorithm for the problem when $O(\sqrt{n})$ work space is allowed.

Acknowledgment. The part of this research of T.A. was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B).

References

1. Alnuweiri, H.M., Prasanna, V.K.: Parallel architectures and algorithms for image component labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* 14(10), 1024–1034 (1992)
2. Asano, T.: Do We Need a Stack to Erase a Component in a Binary Image? In: Boldi, P. (ed.) *FUN 2010. LNCS*, vol. 6099, pp. 16–27. Springer, Heidelberg (2010)
3. Asano, T., Bereg, S., Kirkpatrick, D.: Finding Nearest Larger Neighbors. In: Albers, S., Alt, H., Näher, S. (eds.) *Festschrift Mehlhorn. LNCS*, vol. 5760, pp. 249–260. Springer, Heidelberg (2009)
4. de Berg, M., van Kreveld, M., van Oostrum, R., Overmars, M.: Simple traversal of a subdivision without extra storage. *International Journal of Geographic Information Systems* 11, 359–373 (1997)
5. Bhattacharya, P.: Connected component labeling for binary images on a reconfigurable mesh architectures. *J. Syst. Arch.* 42(4), 309–313 (1996)
6. Bose, P., Morin, P.: An improved algorithm for subdivision traversal without extra storage. *Int. J. Comput. Geometry Appl.* 12(4), 297–308 (2002)
7. Chang, F., Chen, C.J., Lu, C.J.: A linear-time component-labeling algorithm using contour tracing technique. *Comput. Vis. Image Understand.* 93, 206–220 (2004)
8. Choudhary, A., Thakur, R.: Connected component labeling on coarse grain parallel computers: An experimental study. *J. Parallel Distrib. Comput.* 20, 78–83 (1994)
9. Dillencourt, M.B., Samet, H., Tamminen, M.: A general approach to connected-component labeling for arbitrary image representations. *J. ACM* 39(2), 253–280 (1992)
10. Goto, T., Ohta, Y., Yoshida, M., Shirai, Y.: High speed algorithm for component labeling. *Trans. IEICE J72-D-II(2)*, 247–255 (1989) (in Japanese)
11. He, L., Chao, Y., Suzuki, K.: A run-based two-scan labeling algorithm. *IEEE Trans. on Image Processing* 17(5), 749–756 (2008)
12. Klette, R., Rosenfeld, A.: *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Elsevier (2004)
13. Malgouyres, R., More, M.: On the computational complexity of reachability in 2D binary images and some basic problems of 2D digital topology. *Theoretical Computer Science* 283, 67–108 (2002)
14. Munro, J.I., Suwanda, H.: Implicit data structures for fast search and update. *J. of Computer and System Sciences* 21(2), 236–250 (1980)
15. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. *J. ACM* 13(4), 471–494 (1966)
16. Rosenfeld, A.: Connectivity in digital pictures. *J. ACM* 17(1), 146–160 (1970)
17. Rosenfeld, A., Kak, A.C.: *Digital Picture Processing*, 2nd edn., vol. 2. Academic, San Diego (1982)
18. Samet, H.: Connected component labeling using quadtrees. *J. ACM* 28(3), 487–501 (1981)
19. Suzuki, K., Horiba, I., Sugie, N.: Linear-time connected-component labeling based on sequential local operations. *Comput. Vis. Image Understand.* 89, 1–23 (2003)

Small Work Space Algorithms for Some Basic Problems on Binary Images

Tetsuo Asano¹, Sergey Bereg², and Lilian Buzer³

¹ School of Information Science, JAIST, Japan

² Department of Computer Science, University of Texas at Dallas, USA

³ Université Paris-Est, LABINFO-IGM, UMR CNRS 8049,
and Department of Computer Science, ESIEE, France

Abstract. This paper presents space-efficient algorithms for some basic tasks (or problems) on a binary image of n pixels, assuming that an input binary image is stored in a read-only array with random-access. Although efficient algorithms are available for those tasks if $O(n)$ work space (of $O(n \log n)$ bits) is available, we aim to propose efficient algorithms using only limited work space, i.e., $O(1)$ or $O(\sqrt{n})$ space. Tasks to be considered are (1) CCC to count the number of connected components, (2) MERR to report the minimum enclosing rectangle of every connected component, and (3) LCCR to report a largest connected component. We show that we can solve each of CCC, MERR, and LCCR in $O(n \log n)$ time using only $O(1)$ space. If we can use $O(\sqrt{n})$ work space, we can solve them in $O(n)$, $O(n)$, and $O(n + m \log m)$ time, respectively, where m is the number of pixels in the largest connected component.

Keywords: Connected component, Minimum enclosing rectangle, Largest connected component, Space-efficient algorithms.

1 Introduction

Demand for embedded software is growing toward intelligent hardware such as scanners, digital cameras, etc. One of the most important aspects and also a difference between ordinary software in computers and embedded software comes from the constraints on the size of local memory. For example, to design an intelligent scanner, a number of algorithms should be embedded in the scanner. In most of the cases, the size of the pictures is increasing while the amount of work space available for such software is severely limited. Thus, algorithms which require a restricted amount of work space and run reasonably fast are desired.

In this paper we propose several space-efficient algorithms designed for some fundamental image processing tasks. All of the tasks that we consider have straightforward solutions if sufficient memory (typically, of size proportional to the size of the image) is available (see, for example, [5,8]). Solving the same tasks with restricted memory, without severely compromising the running time, is more of a challenge.

1.1 Computational Model with Limited Work Space

We measure the space efficiency of algorithms by the amount of work space used. Such space typically takes the form of pointers and counters (whose number of bits is at most a logarithm of the image size). Our objective is to design efficient algorithms that use only $O(1)$ or $O(\sqrt{n})$ such pointers and counters.

Throughout the paper we assume that an input binary image consists of n pixels in $O(\sqrt{n})$ rows and columns, and it is stored in a read-only array in a random-access manner.

1.2 Three Basic Tasks Considered in This Paper

Three basic tasks for an input binary image are considered in this paper:

CCC (Connected Components Counting). Count the number of connected components.

MERR (Minimum Enclosing Rectangles Reporting). Report the minimum enclosing (axis-parallel) rectangle of every connected component.

LCCR (Largest Connected Component Reporting). Report all pixels of a largest connected component.

We have to output a number of pieces of information except for the task CCC. MERR requires us to output rectangles, and in the case of LCCR, we have to output the pixels in the largest component. It is expensive to use an array for the output. Therefore, we output the information directly without using any array to store it.

1.3 Important Concepts and Terminology

Several important concepts and notions are used in order to design space efficient algorithms for those tasks.

Lexicographical Order: We assume a lexicographical order among all pixels.

A pixel at (x, y) precedes the one at (x', y') if $y < y'$ or $y = y'$ and $x < x'$. The same order is defined for all vertical edges.

Canonical Edge: Connected components are described by external and internal boundaries. The canonical edge of a boundary is the lexicographically smallest vertical edge on the boundary, which is uniquely defined.

Run: A maximal sequence of foreground (white) pixels in a row is called a run.

Run Adjacency Graph: Run adjacency graph represents incidence relations among runs in two consecutive rows. The graph plays an important role in the algorithms to be presented in the paper. It contains $O(\sqrt{n})$ vertices and edges since there are only $O(\sqrt{n})$ columns by the assumption.

Provisional Label: We put labels to runs to maintain information on connected components. Once we know two runs belong to the same component, their labels must be merged. This way a number of labels are created and removed. These temporary labels are called provisional labels. We use some labels many times in our labeling process. A basic idea is to save the total number of labels.

1.4 Results Obtained

We present space efficient algorithms for the tasks listed above using $O(1)$ or $O(\sqrt{n})$ work space. For the tasks CCC (Connected Components Counting) and MERR (Minimum Enclosing Rectangles Reporting) our algorithms run in $O(n \log n)$ time with $O(1)$ work space or $O(n)$ time with $O(\sqrt{n})$ space. For the task LCCR (Largest Connected Components Reporting) our algorithm runs in $O(n \log n)$ time using $O(1)$ work space. If we can use $O(\sqrt{n})$ space, the running time is reduced to $O(n + m \log m)$ time where m is the size of the largest component.

2 Known Results for $O(1)$ -Space Algorithms and Their Extensions

2.1 Basic Algorithm

First, we give an intuitive explanation of an algorithm for counting the number of connected components in a given binary image.

Basic algorithm for counting the number of connected components

```

 $c = 0.$  // counter for the number of connected components
for each pixel  $p$  in the lexicographic order (raster order)
    if it is a unique pixel in a component then increment the counter  $c$ .
Report the counter value  $c$  as the number of connected components.

```

We assume a **lexicographical order** among all pixels. A pixel at (x, y) precedes one at (x', y') if $y < y'$ or $y = y'$ and $x < x'$. The same order is defined for vertical edges.

The algorithm correctly counts the number of connected components if a unique pixel is defined for each connected component and we can determine in some reasonable time whether a given pixel is the unique one. For this purpose we introduce the notion of **canonical edge** instead of a unique pixel.

2.2 Canonical Edge

A connected region in a binary image refers to a maximal set of foreground pixels in which any two of them are connected by a 4-connected path of foreground pixels. It may contain holes, islands, and further holes of islands. Thus, the boundary of a connected component is defined by a single external boundary and possibly more than one internal boundary. In this paper we assume a small square for each pixel, which has four sides. Two adjacent pixels share a side. A side between two pixels of different values is called an edge. Then, a boundary is a sequence of such edges, horizontal or vertical. Any boundary must have a unique vertical edge that is lowest (and leftmost if there are ties). This uniquely defined edge is referred to as the **canonical edge** of the boundary, as it was defined in the literature [46]. Fig. 1 shows how canonical edges are defined using a simple binary image when each boundary is oriented so that foreground pixels always lie to the right.

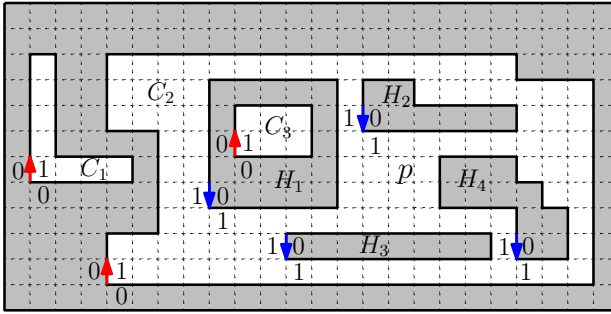


Fig. 1. Geometric model of a binary image. There are three connected components C_1, C_2 and C_3 . C_2 containing four holes H_1, \dots, H_4 , and H_1 including an island C_3 . Edges are directed so that foreground pixels lie to their right. Canonical edges are indicated by arrows.

2.3 Bidirectional Search

It is known that we can enumerate all the canonical edges in $O(n \log n)$ time using the algorithm [3,4] which was originally designed for traversing a planar map without using any mark bits. By the definition of a canonical edge, an edge e is canonical if and only if there is no other boundary edge e' on the same boundary which is lexicographically smaller than e , that is, $e' < e$. The condition can be tested by following the boundary until we find a smaller edge, but it takes time. A magic for acceleration is to search in two opposite directions (**Bidirectional Search**) [4].

Since we can distinguish canonical edges on the external boundaries from those on the internal ones only using local information around them (see Fig. 1), we can count the number of connected components by counting that of canonical edges on external boundaries. Thus, the task CCC can be done in $O(n \log n)$ time using constant work space [1,2].

Algorithm for counting the number of connected components

```

c = 0. // counter for the number of connected components
for each pixel p in the lexicographic order (raster order)
  Let e be the vertical edge to the right of p.
  if LocalCondition(p) and IsCanonical(e) then increment the counter c.
  Report the counter value c as the number of connected components.
Boolean LocalCondition(p){ // Local condition for a canonical pixel
  if p is background and p's right pixel is foreground and p's lower right pixel
  is background then return True else return False.
}
Boolean IsCanonical(e){ // Is an edge e canonical?
  ef = NextEdge(e). // the next edge of e on the boundary.
  eb = PrevEdge(e). // the previous edge of e on the boundary.
  while(ef > e and eb > e) do{

```

```

    e_f = NextEdge(e_f). // forward search
    if e_f = e_b then return True. // if two pointers meet then canonical
    e_b = PrevEdge(e_b). // backward search
    if e_f = e_b then return True. // if two pointers meet then canonical
  }
  return False.
}

```

In the algorithm above, $\text{NextEdge}(e)$ is a function to compute the next edge of e on the boundary (since each boundary is singly connected, the next element is uniquely determined). $\text{PrevEdge}(e)$ is a function for the previous edge of e . $\text{NextEdge}(e)$ and $\text{PrevEdge}(e)$ can be computed in constant time. $\text{IsCanonical}(e)$ is a function to determine whether an edge is canonical or not. This function cannot be computed in constant time, but the total time we need to evaluate the function for every edge is bounded by $O(n \log n)$ due to bidirectional search.

We now know that the first problem CCC – count the number of connected components – is easily solved using only constant work space. It is rather straightforward to extend the algorithm for the task MERR. Whenever we find a canonical edge of an external boundary, we follow the boundary. Then, we can compute the minimum enclosing rectangle of the corresponding component in linear time by maintaining the smallest and largest x and y coordinates of the edges on the boundary. Thus, MERR can be solved in $O(n \log n)$ time using $O(1)$ space.

On the other hand, it is not easy to extend it so that it reports component sizes. The difficulty comes from the existence of holes. If a component has no hole, it suffices to follow its boundary to compute its area. Fortunately, it is also known that the problem can be solved in $O(n \log n)$ time with $O(1)$ space by applying the algorithm due to Bose and Morin [4], which is described below.

2.4 Component Pixel Traversal

The algorithm of Bose and Morin [4], which works on a graph, can be modified to deal with a binary image where the graph structure is implicitly given. We start from the canonical edge of the external boundary of a connected component C and follow the boundaries associated with C . At each upward vertical edge e we walk to the east until we reach a boundary edge f . If f is a canonical edge, then we move to f with f as the current edge. Otherwise, we first check whether the next edge of e on the boundary is canonical or not. If it is the canonical edge of the external boundary (by further checking whether it is upward), then we are done. If it is the canonical edge of a hole, then we walk to the west until we encounter a boundary edge e' and let $e = e'$.

Fig. 2 illustrates how the algorithm traverses pixels in a connected component. Fig. 2(a) shows the first two rows and the entire traversal is given in (b). A more formal description algorithm follows.

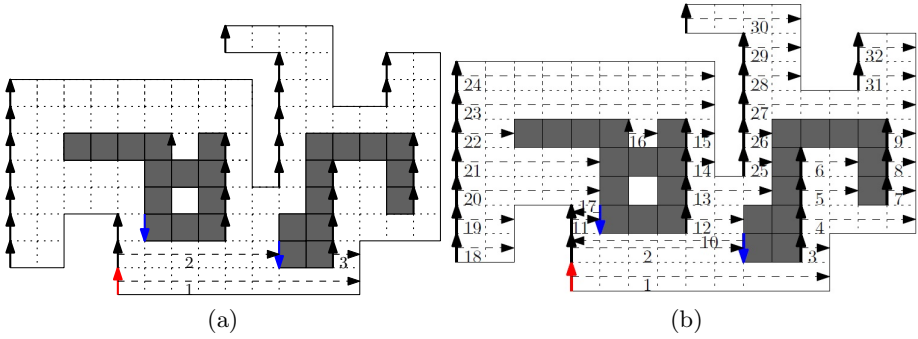


Fig. 2. Component pixel traversal: (a) the first two rows, and (b) the entire traversal

Algorithm for visiting every pixel starting from an edge e_s .

```

 $e = e_s$ . //  $e_s$  is the starting edge.
repeat{
  if  $e$  is horizontal then  $e = \text{NEXTEDGE}(e)$ .
  else if  $e$ 's eastern pixel is foreground then{
    Report consecutive foreground pixels until we reach a boundary edge  $f$ .
    if  $\text{ISCANONICAL}(f)$  then  $e = f$  else  $e = \text{NEXTEDGE}(e)$ .
  }else if  $\text{ISCANONICAL}(e)$  then
    Walk to the west until we reach an edge  $f$  and let  $e = f$ .
    else  $e = \text{NEXTEDGE}(e)$ .
}until( $e = e_s$ )

```

Theorem 1. Let \mathcal{I} be a binary image. When a canonical edge of an external boundary of a connected component C_i is known, we can report all pixels of C_i in $O(a_C + b_C \log b_C)$ time using $O(1)$ work space, where a_C denotes the area of C and b_C denotes the number of edges defining the boundaries of C .

Using the results listed above, we have the following theorem.

Theorem 2. Given a binary image of n pixels, we can solve the three basic problems (CCC, MERR, and LCCR) in $O(n \log n)$ time using $O(1)$ work space.

Proof. Given a binary image, we examine every boundary edge whether it is the canonical edge of an external boundary. It is done in $O(n \log n)$ time in total. Thus, counting the number of connected components is straightforward. To solve MERR it suffices to maintain the minimum and maximum x and y coordinates of edges on an external boundary.

A largest connected component can be computed in two phases. In the first phase we detect all canonical edges of external boundaries. For each such canonical edge e we count the number of pixels of the component associated with e by applying the function for component pixel traversal. In this way we maintain a canonical edge having the largest count. Then, in the second phase we apply the function for component pixel traversal again with the canonical edge obtained in the first phase. \square

3 $O(\sqrt{n})$ -Space Algorithms Using Run Adjacency Graph

From now on we will consider algorithms using more work space, $O(\sqrt{n})$ space. But, before introducing such algorithms we will start with a linear-space algorithm for the labeling to explain our basic ideas of run adjacency graph and provisional labels.

In our algorithm we read an input image row by row in the raster manner and convert each row into a sequence of runs. More exactly, a run r is a maximal sequence of foreground pixels (of value 1) in a row. It is associated with an interval $I(r) = \{s, s + 1, \dots, t\}$ when it starts at the s -th column and ends at the t -th column. So, a white run (s, t) in the i -th row means that foreground pixels continue from the s -th column to the t -th column and $(s - 1)$ -st and $(t + 1)$ -st pixels are both black pixels in the row (if they exist).

Definition 1. *A foreground run r_1 “intersects” another foreground run r_2 if they are in consecutive rows and their associated intervals denoted by $I(r_1)$ and $I(r_2)$ have non-empty intersection.*

We construct a graph called a **Run Adjacency Graph** with vertices being runs and edges between two intersecting runs in consecutive rows. Then, we partition the graph into connected components $\{C_1, C_2, \dots, C_k\}$ by applying a depth-first search and assign the integral label i to each run in the component C_i . To distinguish connected components in a graph from those in a binary image, we call the former as **graph components** and the latter as **image components**. There is a one-to-one correspondence between graph components and image components. So, the last phase is to convert the run representation into a labeling matrix using the label for each run.

The algorithm described above is almost the same as the old one by Rosenfeld and Pfalts [7], which consists of two phases, horizontal scan to partition each row into runs and vertical scan to merge vertically adjacent runs using a union-find data structure. Differences are (1) we use horizontal scan twice (instead of horizontal scan followed by vertical scan) and (2) we use a depth-first algorithm for computing graph components after building a run adjacency graph (instead of using a union-find tree data structure). Since the depth-first algorithm runs in linear time, the whole algorithm runs in linear time. This is a folklore knowledge although such a formal statement is rather rare in the literature.

In this paper we are interested in space-efficient algorithms, especially using $O(\sqrt{n})$ space. Due to the space constraint we cannot build the whole run adjacency graph. A key idea is to use the $O(\sqrt{n})$ work space to keep a set of runs in two consecutive rows.

Provisional Label: An Idea to Save Space

Unfortunately, there are $O(n)$ runs in a binary image and thus $O(n)$ vertices in its associated run adjacency graph. There are two ideas to reduce the work space. The first idea is to introduce a notion of **provisional labels** which are

labels temporarily used in the algorithm and may be different from the final labels to be reported. More important is that we can use the same provisional label for two graph components if they are "clearly" separated.

We read an input binary image in the lexicographic order (raster order) row by row. We read the first row and put provisional labels to those runs in the row. Then, in the second row, we construct a modified run adjacency graph for a set of runs in the two rows and then partition it into connected components (graph components). In this way we know how those runs in the first two rows are connected and thus we can put provisional labels to the runs in the second row. In general, assuming that those runs in the previous row have been labeled, we put provisional labels to the runs in the current row by examining connectivities among those runs. Hereafter, those runs with provisional labels in the previous row are called **colored** runs, and those runs with no provisional labels yet **white** runs. A set of colored runs which have the same provisional label is replaced by a path connecting them in a line.

The graph can be partitioned into connected components (graph components) by applying depth-first search which runs in linear time. The resulting graph components are classified into three kinds (see Fig. 3):

Starting Component: consisting of a single white run in the current row,

Terminating Component: consisting of colored runs in the previous row, and

Extending Component: including both of colored and white runs in the two rows.

1. Starting Component: A graph component C of a single white run.

If a graph component C is a singleton consisting of a single white run in the current row, then it means a new image component starts at the current row, which may be merged in the future scan with another existing image component. So, we need a procedure to create a new image component with a new label. We maintain labels using integers with the current largest label L . Hence, whenever we create a new image component, we increment the value of L and assign the value to the new image component. We also maintain the size of each image component by an array $size[]$, which is initially determined by the run length.

2. Terminating Component: A graph component C consisting of colored runs and containing no white run. It may have two or more colored runs, which must be connected in the graph by a path. Therefore, all the colored runs in the component must have a single common provisional label. Since no colored run in the component intersects any white run in the current row, it does not extend to the current row. That is, the corresponding image component terminates in the previous row. This event is called a **death** of the image component.

3. Extending Component: A graph component C having both of colored and white run(s).

If a component C has at least one colored run and at least one white run, then the corresponding image component extends to the next row. If the label k is the smallest label among those colored runs in the graph component, then all

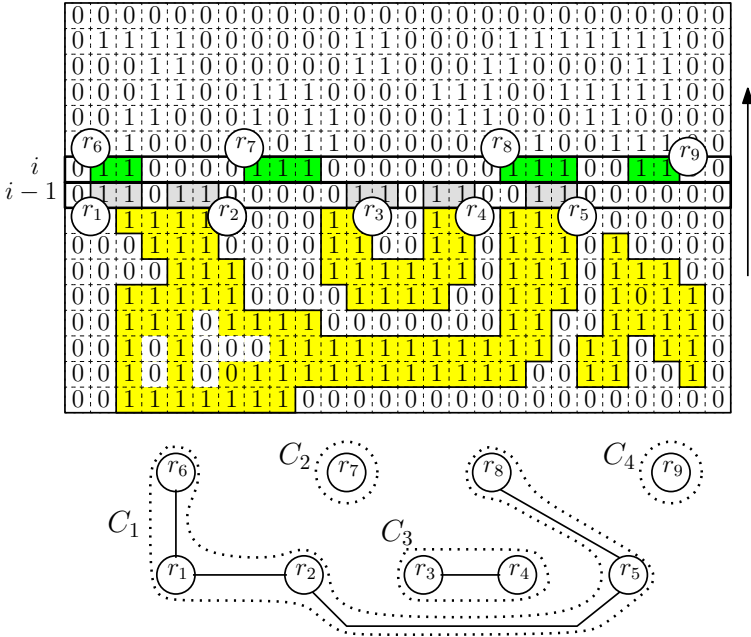


Fig. 3. A set of runs in two consecutive rows (rows $i - 1$ and i). Runs r_1, \dots, r_5 in the previous row (row $i - 1$) have been labeled by connectivities established in the already scanned part. The run r_7 and r_9 in the current row intersect no run in the previous row, and hence they may create new components. On the other hand, the component associated with the runs r_3 and r_4 does not extend to the current row, and hence the component terminates here. The corresponding run adjacency graph is given below in the figure.

the pixels associated with those labels and runs in the graph components should be labeled as k in the next row. At the same time, we update the size of the merged graph component C labeled k by the sum of the sizes of all associated image components.

Fig. 4 illustrates how provisional labels are created, propagated, and terminate during raster scan.

As is seen in Fig. 4, we can save a number of provisional labels. Unfortunately, however, this is not enough to achieve $O(\sqrt{n})$ work space. We need another idea.

A key idea is to reuse provisional labels again and again. A provisional label disappears in two ways. A colored component of a label terminates at some row or it is merged into another terminating component of a different (and smaller) label. The latter case happens when a white run in the current row intersects two or more terminating components of different provisional labels in the previous row. In this case those labels are merged into one of the smallest label together with their associated information such as their sizes.

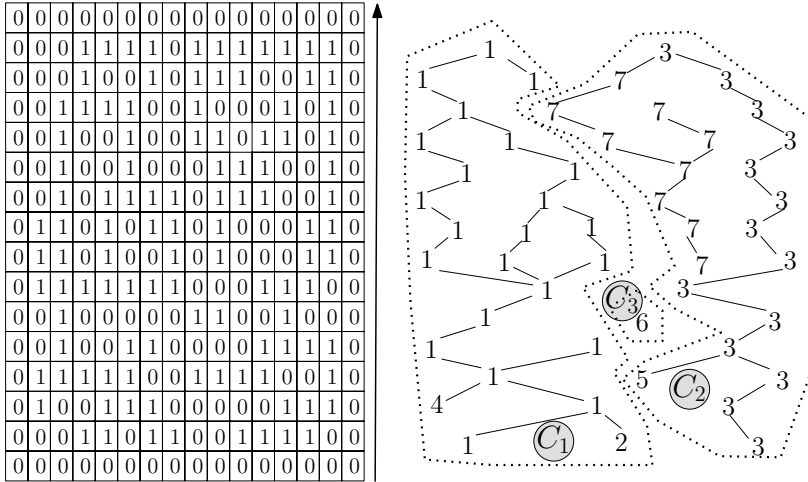


Fig. 4. Provisional labels propagated during raster scan

3.1 $O(\sqrt{n})$ -Space Algorithm for CCC

Now we are ready to present an $O(\sqrt{n})$ -space algorithm for the task CCC for counting the number of connected components in a given binary image of n pixels.

Lemma 1. *For an arbitrary binary image consisting of $O(\sqrt{n})$ rows and columns, the maximum number of image components in a row (or a column) is $O(\sqrt{n})$.*

Proof. If 0's and 1's alternate in a row and all 1-pixels belong to different components we have $O(\sqrt{n})$ components in the row, which is the worst case. Thus, the lemma follows. \square

In the above algorithm we have created a new label whenever we find a run which is not connected with any component in the part already scanned. A label may be merged into another. To save work space, we maintain two sets of labels, U and V . The set U keeps a set of labels currently used. The set V is a set of those labels which have been used before but are not used currently. Whenever we need a new label, we check the set V and takes one label out of V unless it is empty. If V is empty then we create a new label and use it by incrementing the value of L . Whenever a label has terminated or is merged into another, we move the label from U to V . In this way we maintain a set of labels. Then, Lemma 1 guarantees that the maximum size of $U \cup V$ is $O(\sqrt{n})$. Thus, we have

Theorem 3. *Given a binary image of n pixels, we can report the number of connected components and the size of each component in $O(n)$ time using $O(\sqrt{n})$ work space.*

Proof. In each row we build a run adjacency graph which contains $O(\sqrt{n})$ vertices and edges. We can decompose it into connected components in linear time using depth-first search. Other operations are done in constant time by standard techniques for such data structures. Thus, the total time required is linear in the number of pixels. \square

3.2 $O(\sqrt{n})$ -Space Algorithm for MERR

It is rather straightforward to modify the algorithm for CCC so that it can also report the minimum enclosing rectangle of every connected component using $O(\sqrt{n})$ work space. What we should do is to maintain rectangle information when two or more provisional labels are merged.

Theorem 4. *Given a binary image of n pixels, we can report the minimum enclosing rectangle of every connected component in $O(n)$ time and $O(\sqrt{n})$ work space.*

3.3 $O(\sqrt{n})$ -Space Algorithm for LCCR

We have already had an $O(n \log n)$ -time and $O(1)$ -space algorithm for solving the problem LCCR (Largest Connected Component Reporting). Can we solve it in linear time using $O(\sqrt{n})$ work space? Unfortunately, it seems very hard, but some small improvement is possible. Exactly speaking, we can design an algorithm which runs in $O(n + m \log m)$ time using $O(\sqrt{n})$ work space, where m is the size of a largest connected component.

We modify our algorithm for reporting the smallest enclosing rectangle for each connected component. By the definition of the canonical edge of the external boundary of a connected component C it must be located at the leftmost lowest corner. We modify the algorithm so that we maintain the following information

- (1) the leftmost lowest edge,
- (2) the number of pixels for each label. Then, in one scan over an input image we get the leftmost lower edge e of a largest component in $O(n)$ time. Then, we apply the $O(1)$ -space algorithm for component pixel traversal which runs in $O(m \log m)$ time for a connected component of m pixels.

Theorem 5. *Given a binary image of n pixels, we can report all pixels of a largest connected component of size m in $O(n + m \log m)$ time and $O(\sqrt{n})$ work space.*

4 Conclusions

In this paper we have presented space efficient algorithms for some basic tasks on binary images. We have shown that such basic tasks can be done in linear or almost linear time even if work space is limited to $O(\sqrt{n})$ or further to $O(1)$. Counting the number of connected components is rather easy. In fact we had a

linear-time algorithm if $O(\sqrt{n})$ work space is available. However, it is not known whether we can report all pixels in a largest connected component in linear time or not.

In this paper we implicitly assumed 4-connectivity for foreground pixels, that is, two foreground pixels are directly 4-connected if they are horizontally or vertically adjacent. In the 8-connectivity the neighborhood includes all eight neighbors around a pixel. It is rather easy to adapt our algorithms for 8-connectivity. It suffices to modify the definition of intersection between two runs in two consecutive rows. For the 4-connectivity a run $[s_1, t_1]$ intersects another run $[s_2, t_2]$ if their associated intervals have non-empty intersection, that is, $[s_1, t_1] \cup [s_2, t_2] \neq \emptyset$. For the 8-connectivity it is the case if $[s_1, t_1] \cap [s_2 - 1, t_2 + 1] \neq \emptyset$ or $[s_1 - 1, t_1 + 1] \cap [s_2, t_2] \neq \emptyset$.

Acknowledgment. The part of this research of T.A. was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B).

References

1. Asano, T.: Do we need a stack to erase a component in a binary image? In: Fifth International Conference on Fun with Algorithms, pp. 16–27 (2010)
2. Asano, T., Bereg, S., Kirkpatrick, D.: Finding Nearest Larger Neighbors. In: Albers, S., Alt, H., Näher, S. (eds.) Efficient Algorithms. LNCS, vol. 5760, pp. 249–260. Springer, Heidelberg (2009)
3. de Berg, M., van Kreveld, M., van Oostrum, R., Overmars, M.: Simple traversal of a subdivision without extra storage. *International Journal of Geographic Information Systems* 11, 359–373 (1997)
4. Bose, P., Morin, P.: An improved algorithm for subdivision traversal without extra storage. *Int. J. Comput. Geometry Appl.* 12(4), 297–308 (2002)
5. Klette, R., Rosenfeld, A.: *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Elsevier (2004)
6. Klette, R., Rosenfeld, A.: *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Elsevier (2004)
7. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. *J. ACM* 13(4), 471–494 (1966)
8. Rosenfeld, A., Kak, A.C.: *Digital Picture Processing*, 2nd edn., vol. 2. Academic, San Diego (1982)

Adjacencies for Structuring the Digital Plane

Josef Šlapal

Brno University of Technology, Institute of Mathematics
616 69 Brno, Czech Republic
slapal@fme.vutbr.cz

Abstract. We study graphs with the vertex set \mathbb{Z}^2 which are subgraphs of the 8-adjacency graph and have the property that certain natural cycles in these graphs are Jordan curves, i.e., separate \mathbb{Z}^2 into exactly two connected components. For the minimal graphs with this property, we discuss their quotient graphs, too.

Keywords: simple graph, adjacency, digital plane, Jordan curve.

1 Introduction

In digital image processing, the well-known binary relations of 4-adjacency and 8-adjacency are used for structuring the digital plane \mathbb{Z}^2 . A disadvantage of this approach is that neither 4-adjacency nor 8-adjacency itself allows for an analogue of the Jordan curve theorem – cf. [5]. To eliminate this deficiency, a combination of the two adjacencies has to be used. Despite this inconvenience, the approach based on using a combination of the 4-adjacency and 8-adjacency proved to be useful for solving many problems of digital image processing and for writing efficient graphic software.

In [2], a new, purely topological approach to the problem of structuring the digital plane was proposed which utilizes a single convenient structure on \mathbb{Z}^2 , namely the Khalimsky topology. The topological approach was then developed by many authors – see, e.g., [3-6] and [9-16]. Since the Khalimsky topological space is an Alexandroff space (i.e., has a completely additive closure), its connectedness coincides with the connectedness in a certain graph with the vertex set \mathbb{Z}^2 , the so-called connectedness graph of the topology. Thus, when studying the connectedness of digital images, the adjacency of this graph, rather than the Khalimsky topology, may be used for structuring the digital plane. A well-known analogue of the Jordan curve theorem is then valid in the graph – cf. [2]. A disadvantage of this approach is that Jordan curves in the (connectedness graph of the) Khalimsky topology cannot turn, at any point of \mathbb{Z}^2 , to form an acute angle of $\frac{\pi}{4}$. It would therefore be useful to find some new, more convenient adjacencies on \mathbb{Z}^2 . In the present note, we introduce a certain natural graph with the vertex set \mathbb{Z}^2 whose cycles are eligible for Jordan curves in \mathbb{Z}^2 and we solve the problem of finding adjacencies on \mathbb{Z}^2 with respect to which these cycles are Jordan curves. We will be, in particular, interested in the minimal adjacencies having this property. We will show that their quotient adjacencies provide a number

of convenient adjacencies on \mathbb{Z}^2 including the one given by the connectedness graph of the Khalimsky topology. The results obtained suggest new background structures on \mathbb{Z}^2 with natural Jordan curves which may be used for studying and processing digital images.

2 Preliminaries

For the graph-theoretic concepts used in the sequel, see, for instance, [1]. By a *graph* on a set V we always mean an undirected simple graph without loops whose vertex set is V . Thus, such a graph is a pair (V, E) where $E \subseteq \{\{a, b\}; a, b \in V, a \neq b\}$ is the set of edges of the graph. An edge $\{a, b\} \in E$ is said to *join* the vertices a and b . We also say that a and b are *incident* with $\{a, b\}$ and that $\{a, b\}$ is *incident* with each of the vertices a and b . For an arbitrary vertex $a \in V$, we denote by $E(a)$ the set of all vertices joined by an edge with a , i.e., $E(a) = \{b \in V; \{a, b\} \in E\}$. Clearly, $\{a, b\} \in E$ if and only if $b \in E(a)$ or, equivalently, $a \in E(b)$. Thus, the set E of edges of a graph may be given by determining the set $E(a)$ for every $a \in V$.

A graph (U, F) is a *subgraph* of (V, E) if $U \subseteq V$ and $F \subseteq E$. In this case, we also say that (V, E) is a *supergraph* of (U, F) . If, moreover, $F = E \cap \{\{a, b\}; a, b \in U\}$, then (U, F) is said to be an *induced subgraph* of (V, E) and is denoted briefly by U . Given graphs (V, E) and (U, F) , their *union* is the graph $(V \cup U, E \cup F)$. We represent graphs by demonstrating vertices as points and edges as line segments whose end points are just the vertices they join.

Let (V, E) and (W, F) be graphs. A map $f : V \rightarrow W$ is called a *homomorphism* of (V, E) into (W, F) if $\{f(a), f(b)\} \in F$ whenever $\{a, b\} \in E$. The graph (W, F) is said to be *finer* than (V, E) and the graph (V, E) is said to be *coarser* than (W, F) if $W = V$ and $E \subseteq F$ (so that (V, E) is a subgraph of (W, F)).

Given a graph (V, E) and a surjection $e : V \rightarrow W$, a graph (W, F) is called the *quotient graph* of (V, E) generated by e if (W, F) is the coarsest graph on W for which e is a homomorphism of (V, E) into (W, F) . It is obvious that (W, F) is the quotient graph of (V, E) generated by a surjection $e : V \rightarrow W$ if and only if, for every pair of vertices $c, d \in W$, $\{c, d\} \in F \Leftrightarrow$ there exist $a \in e^{-1}(\{c\})$ and $b \in e^{-1}(\{d\})$ such that $\{a, b\} \in E$.

Recall that a *path* in a graph (V, E) is a finite (nonempty) sequence $(a_i \mid i \leq n) = (a_0, a_1, \dots, a_n)$ of pairwise different vertices such that $\{a_{i-1}, a_i\} \in E$ whenever $i \in \{1, 2, \dots, n\}$. The number n is then called the *length* of the path. Thus, also a single vertex is considered to be a path (of length 0). A *cycle* in (V, E) is any finite set of at least three vertices which can be ordered into a sequence to form a path whose first and last members are joined by an edge. A subset $X \subseteq V$ is said to be *connected* if, for every pair $a, b \in X$, there is a path $(a_i \mid i \leq n)$ such that $a_0 = a$, $a_n = b$ and $a_i \in X$ for all $i \in \{0, 1, \dots, n\}$. A maximal (with respect to set inclusion) connected subset of V is called a *component* of the graph (V, E) .

A nonempty, finite and connected subset C of V is said to be a *simple closed curve* in (V, E) if the set $E(a) \cap C$ has precisely two elements for every $a \in C$. Clearly, every simple closed curve is a cycle. A simple closed curve in (V, E) is

called a *Jordan curve* if it separates the set V into precisely two components, i.e., if the induced subgraph $V - C$ of (V, E) has exactly two components.

We will need the following statement:

Lemma 1. *Let (V, E) be a graph, $e : V \rightarrow W$ be a surjection and let (W, F) be the quotient graph of (V, E) generated by e . Let e have the property that $e^{-1}(\{y\})$ is connected in (V, E) for every point $y \in W$ and let $B \subseteq W$ be a subset. Then B is connected in (W, F) if and only if $e^{-1}(B)$ is connected in (V, E) .*

Proof. If $e^{-1}(B)$ is connected in (V, E) , then B is connected in (W, F) because $B = e(e^{-1}(B))$ (clearly, a homomorphic image of a connected set is connected). Conversely, let B be connected in (W, F) and let $a, b \in e^{-1}(B)$. Then the pair $e(a), e(b)$ of points of B may be joined by a path $(e(a) = c_0, c_1, \dots, c_n = e(b))$ in (W, F) contained in B . Since (W, F) is the quotient graph of (V, E) generated by E , there exist, for every $i \in \{1, 2, \dots, n\}$, points $a_{i-1} \in e^{-1}(\{c_{i-1}\})$ and $b_i \in e^{-1}(\{c_i\})$ such that $\{a_{i-1}, b_i\} \in E$. Put $a = b_0$ and $b = a_n$. Then we get a sequence $(b_0, a_0, b_1, a_1, b_2, a_2, \dots, b_{n-1}, a_{n-1}, b_n, a_n)$ where $\{a_{i-1}, b_i\} \in E$ for every $i \in \{1, 2, \dots, n\}$ and $\{b_i, a_i\} \subseteq e^{-1}(\{c_i\})$ for every $i \in \{0, 1, \dots, n\}$. As $e^{-1}(\{c_i\})$ is connected in (V, E) for every $i \in \{0, 1, \dots, n\}$, the points $b_i, a_i \in V$ may be joined by a path $(b_i = a_0^i, a_1^i, \dots, a_{n_i}^i = a_i)$ in (V, E) contained in $e^{-1}(\{c_i\}) \subseteq e^{-1}(B)$ for every $i \in \{0, 1, \dots, n\}$. Consequently, the path $(a = b_0 = a_0^0, a_1^0, \dots, a_{n_0}^0 = a_0, b_1 = a_1^1, a_1^1, \dots, a_{n_1}^1 = a_1, b_2 = a_2^2, a_2^2, \dots, a_{n_2}^2 = a_2, \dots, a_{n_{n-1}}^{n-1} = a_{n-1}, b_n = a_n^n, a_n^n, \dots, a_{n_n}^n = a_n = b)$ in (E, V) connects a and b and is contained in $e^{-1}(B)$. Thus, $e^{-1}(B)$ is connected in (V, E) .

In the sequel, we will consider graphs on \mathbb{Z}^2 only. For every point $(x, y) \in \mathbb{Z}^2$, we denote by $H_2(x, y)$ and $V_2(x, y)$ the sets of all points that are *horizontally 2-adjacent* and *vertically 2-adjacent* to (x, y) , respectively, i.e., $H_2(x, y) = \{(x-1, y), (x+1, y)\}$ and $V_2(x, y) = \{(x, y-1), (x, y+1)\}$. Further, we denote by $A_4(x, y)$ and $A_8(x, y)$ the sets of all points that are *4-adjacent* and *8-adjacent* to (x, y) , respectively. Thus, $A_4(x, y) = \{(x+i, y+j); i, j \in \{-1, 0, 1\}, ij = 0, i+j \neq 0\}$ and $A_8(x, y) = A_4(x, y) \cup \{(x+i, y+j); i, j \in \{-1, 1\}\}$. The graphs (\mathbb{Z}^2, A_4) and (\mathbb{Z}^2, A_8) are called the *4-adjacency graph* and *8-adjacency graph*, respectively. An arbitrary subset $A \subseteq A_8$ is said to be an *adjacency* on \mathbb{Z}^2 and the graph (\mathbb{Z}^2, A) is said to be an *adjacency graph*. Thus, adjacency graphs are exactly the graphs on \mathbb{Z}^2 that are subgraphs of the 8-adjacency graph, i.e., the graphs (\mathbb{Z}^2, A) with the property $A(z) \subseteq A_8(z)$ for every $z \in \mathbb{Z}^2$. In an adjacency graph (\mathbb{Z}^2, A) , vertices $a, b \in \mathbb{Z}^2$ are said to be *adjacent* if $\{a, b\} \in A$, i.e., if they are joined by an edge. If an adjacency graph (\mathbb{Z}^2, B) is a quotient graph of an adjacency graph (\mathbb{Z}^2, A) generated by a surjection $e : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$, then we say that the adjacency B is the *quotient adjacency* of A generated by e .

Definition 1. The *square-diagonal graph* is the adjacency graph in which two points $z_1 = (x_1, y_1), z_2 = (x_2, y_2) \in \mathbb{Z}^2$ are adjacent if and only if one of the following four conditions is fulfilled:

1. $|y_1 - y_2| = 1$ and $x_1 = x_2 = 4k$ for some $k \in \mathbb{Z}$,
2. $|x_1 - x_2| = 1$ and $y_1 = y_2 = 4l$ for some $l \in \mathbb{Z}$;
3. $x_1 - x_2 = y_1 - y_2 = \pm 1$ and $x_1 - 4k = y_1$ for some $k \in \mathbb{Z}$,
4. $x_1 - x_2 = y_2 - y_1 = \pm 1$ and $x_1 = 4l - y_1$ for some $l \in \mathbb{Z}$.

A section of the square-diagonal graph is shown in Fig. 1.

When studying digital images, it may be advantageous to equip \mathbb{Z}^2 with a structure with respect to which all or most of the cycles in the square-diagonal graph are Jordan curves. Such a convenient structure given by a topology was introduced and studied in [12]. In this note, we focus on those convenient structures on \mathbb{Z}^2 which are given by adjacencies.

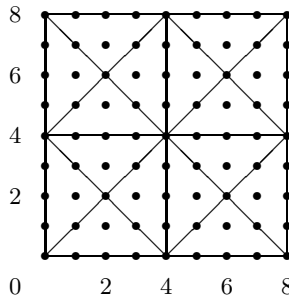


Fig. 1. A section of the square-diagonal graph

3 Convenient Adjacencies on \mathbb{Z}^2

Definition 2. An adjacency A on \mathbb{Z}^2 is said to be an *sd-adjacency* if every cycle in the square-diagonal graph is a Jordan curve in (\mathbb{Z}^2, A) .

The two adjacencies on \mathbb{Z}^2 that are most frequently used for the study of digital images, namely the 4-adjacency and 8-adjacency, are not *sd-adjacencies*. Another adjacency on \mathbb{Z}^2 , which has been used since late 1980's, is given by the connectedness graph of the Khalimsky topology on \mathbb{Z}^2 (i.e., the graph on \mathbb{Z}^2 in which arbitrary vertices $z_1, z_2 \in \mathbb{Z}^2$ are joined by an edge if and only if $z_1 \neq z_2$ and $\{z_1, z_2\}$ is a connected subset of the Khalimsky topological space). This adjacency will be called the *Khalimsky adjacency* and the corresponding adjacency graph will be called the *Khalimsky graph*.

The Khalimsky graph coincides with the adjacency graph (\mathbb{Z}^2, K) given as follows:

For any $z = (x, y) \in \mathbb{Z}^2$,

$$K(z) = \begin{cases} A_8(z) & \text{if } x + y \text{ is even,} \\ A_4(z) & \text{if } x + y \text{ is odd.} \end{cases}$$

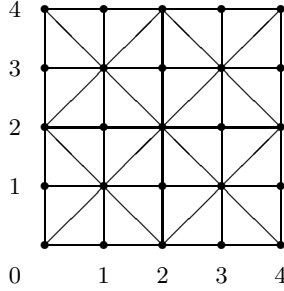


Fig. 2. A section of the Khalimsky graph

A section of the Khalimsky graph is shown in Fig. 2.

The main result of [2] implies that, in the Khalimsky graph, every simple closed curve with at least four points is a Jordan curve. But it is readily verified that the Khalimsky adjacency is not an *sd*-adjacency. More precisely, a cycle in the square-diagonal graph is a Jordan curve in the Khalimsky graph if and only if it does not turn, at any of its points, to form an acute angle of $\frac{\pi}{4}$ - cf. [2]. It could therefore be useful to replace the Khalimsky adjacency with some *sd*-adjacencies allowing Jordan curves to form acute angles of $\frac{\pi}{4}$ at some points. To this end, we define:

Definition 3. An adjacency A on \mathbb{Z}^2 is said to be *basic* if, for every point $z = (x, y) \in \mathbb{Z}^2$,

$$A(z) = \begin{cases} A_8(z) & \text{if } x = 4k, y = 4l, k, l \in \mathbb{Z}, \\ (A_8(z) - A_4(z)) & \text{if } x = 2 + 4k, y = 2 + 4l, k, l \in \mathbb{Z}, \\ \{(x - 1, y), (x + 1, y), (m, n)\} & \text{where } (m, n) \in \{(u, v)\} \cup H_2(u, v) \text{ if} \\ & z \in V_2(u, v) \text{ where } u = 4k + 2, v = 4l, k, l \in \mathbb{Z}, \\ \{(x, y - 1), (x, y + 1), (m, n)\} & \text{where } (m, n) \in \{(u, v)\} \cup V_2(u, v) \text{ if} \\ & z \in H_2(u, v) \text{ where } u = 4k, v = 4l + 2, k, l \in \mathbb{Z}, \end{cases}$$

$$A(z) - H_2(z) = V_2(z) \text{ if } x = 4k, y = 4l + 2, k, l \in \mathbb{Z}, \text{ and} \\ A(z) - V_2(z) = H_2(z) \text{ if } x = 4k + 2, y = 4l, k, l \in \mathbb{Z}.$$

Basic adjacency graphs are demonstrated in Fig. 3. A section of a basic graph is obtained by just choosing, for every vertex demonstrated by a bold dot, exactly one of the three edges denoted by the bold line segments that are incident with this vertex.

Theorem 1. *The basic adjacencies on \mathbb{Z}^2 are precisely the minimal *sd*-adjacencies.*

Proof. Let A be a basic adjacency on \mathbb{Z}^2 . Clearly, any cycle in the square-diagonal graph is a simple closed curve in (\mathbb{Z}^2, A) . Let $z = (x, y) \in \mathbb{Z}^2$ be a

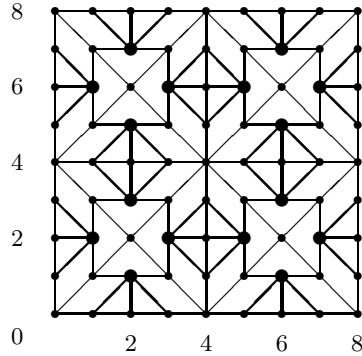
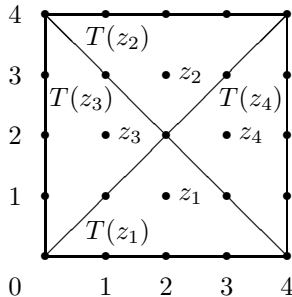


Fig. 3. A demonstration of (sections of) basic adjacency graphs

point such that $x = 4k + p$ and $y = 4l + q$ for some $k, l, p, q \in \mathbb{Z}$ with $pq = \pm 2$. Then we define the *fundamental triangle* $T(z)$ to be the nine-point subset of \mathbb{Z}^2 given below:

$$T(z) = \begin{cases} \{(r, s) \in \mathbb{Z}^2; y - 1 \leq s \leq y + 1 - |r - x|\} \text{ if} \\ x = 4k + 2 \text{ and } y = 4l + 1 \text{ for some } k, l \in \mathbb{Z}, \\ \{(r, s) \in \mathbb{Z}^2; y - 1 + |r - x| \leq s \leq y + 1\} \text{ if} \\ x = 4k + 2 \text{ and } y = 4l - 1 \text{ for some } k, l \in \mathbb{Z}, \\ \{(r, s) \in \mathbb{Z}^2; x - 1 \leq r \leq x + 1 - |s - y|\} \text{ if} \\ x = 4k + 1 \text{ and } y = 4l + 2 \text{ for some } k, l \in \mathbb{Z}, \\ \{(r, s) \in \mathbb{Z}^2; x - 1 + |s - y| \leq r \leq x + 1\} \text{ if} \\ x = 4k - 1 \text{ and } y = 4l + 2 \text{ for some } k, l \in \mathbb{Z}. \end{cases}$$

Graphically, the fundamental triangle $T(z)$ consists of the point z and the eight points lying on the triangle surrounding z - the four types of fundamental triangles are represented in the following figure:



Given a fundamental triangle, we speak about its sides - it is clear from the above picture which sets are understood to be the sides (note that each side consists of five or three points and that two different fundamental triangles may have at most one side in common).

Now, one can easily see that:

1. Every fundamental triangle is connected (so that the union of two fundamental triangles having a common side is connected) in (\mathbb{Z}^2, A) .
2. If we subtract from a fundamental triangle some of its sides, then the resulting set is still connected in (\mathbb{Z}^2, A) .
3. If S_1, S_2 are fundamental triangles having a common side D , then the set $(S_1 \cup S_2) - M$ is connected in (\mathbb{Z}^2, A) whenever M is the union of some sides of S_1 or S_2 different from D .
4. Every connected subset of \mathbb{Z}^2 having at most two points is a subset of a fundamental triangle.

Analogously to the proof of Theorem 1 in [14], we may show that the following is also true:

5. For an arbitrary cycle C in the square-diagonal graph, there are sequences $\mathcal{S}_F, \mathcal{S}_I$ of fundamental triangles, \mathcal{S}_F finite and \mathcal{S}_I infinite, such that, whenever $\mathcal{S} \in \{\mathcal{S}_F, \mathcal{S}_I\}$, the following two conditions are satisfied:
 - (a) Each member of \mathcal{S} , excluding the first one, has a common side with at least one of its predecessors.
 - (b) C is the union of those sides of fundamental triangles in \mathcal{S} that are not shared by two different fundamental triangles from \mathcal{S} .

Given a cycle C in the square-diagonal graph, let S_F and S_I denote the union of all members of \mathcal{S}_F and \mathcal{S}_I , respectively. Then $S_F \cup S_I = \mathbb{Z}^2$ and $S_F \cap S_I = C$. Let S_F^* and S_I^* be the sequences obtained from \mathcal{S}_F and \mathcal{S}_I by subtracting C from each member of \mathcal{S}_F and \mathcal{S}_I , respectively. Let S_F^* and S_I^* denote the union of all members of \mathcal{S}_F^* and \mathcal{S}_I^* , respectively. Then S_F^* and S_I^* are connected by (1), (2) and (3) and it is clear that $S_F^* = S_F - C$ and $S_I^* = S_I - C$. So, S_F^* and S_I^* are the two components of $\mathbb{Z}^2 - C$ by (4) ($S_F - C$ is called the *inside* component and $S_I - C$ is called the *outside* component). Therefore, A is an *sd*-adjacency.

To show that A is a minimal *sd*-adjacency, let B be an *sd*-adjacency such that $B \subseteq A$. Suppose that there is an edge $\{z_1, z_2\} \in A - B$. Since (\mathbb{Z}^2, B) is a supergraph of the square-diagonal graph, there is a fundamental triangle $T(z)$ with $z \in \{z_1, z_2\}$. Thus, $\{z_1, z_2\}$ is one of the three edges incident with z and the point $z' \in \{z_1, z_2\} - \{z\}$ lies on a side D of $T(z)$. Let S be the fundamental triangle different from $T(z)$ such that one of the sides of S is D . Then the union C of all sides of $T(z)$ and S different from D is a cycle in the square diagonal graph but it is not a Jordan curve in (\mathbb{Z}^2, B) because the inside part of C , i.e., the set $(T(z) \cup S) - C$, is evidently not connected in the subgraph $\mathbb{Z}^2 - C$ of (\mathbb{Z}^2, B) . Thus, the subgraph $\mathbb{Z}^2 - C$ of (\mathbb{Z}^2, B) has more than two components. This is a contradiction. Therefore, $A = B$ and the minimality of (\mathbb{Z}^2, A) is proved.

Remark 1. It follows from the proof of Theorem 1 that, if A is a basic adjacency on \mathbb{Z}^2 , then an adjacency B on \mathbb{Z}^2 is an *sd*-adjacency whenever, for every edge $\{a, b\} \in B - A$, there exists a fundamental triangle T with $\{a, b\} \subseteq T$ such

that the union of all sides of T is a simple closed curve in (\mathbb{Z}^2, B) . Three sd -adjacency graphs are obtained as connectedness graphs of the closure operators on \mathbb{Z}^2 studied in [11] and [14-15]. Sections of these graphs are shown in Fig. 4. Note that only the first of the three adjacencies given by these graphs is basic and that the third adjacency graph is the union of the first two.

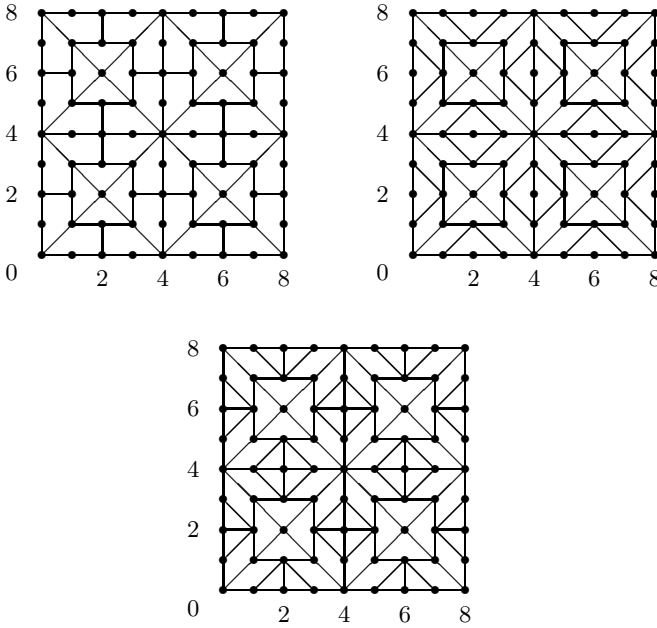


Fig. 4. Sections of the sd -adjacency graphs that coincide with the connectedness graphs of the closure operators discussed in [11] and [14-15]

4 Quotients of the Basic Adjacencies

We will show that certain interesting adjacencies on \mathbb{Z}^2 may be obtained as quotient adjacencies of the basic adjacencies.

Let G be the adjacency \mathbb{Z}^2 defined as follows:

For any $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}^2$, $\{(x_1, y_1), (x_2, y_2)\} \in G$ if and only if one of the following four conditions is fulfilled:

1. $|y_1 - y_2| = 1$ and $x_1 = x_2 = 2k$ for some $k \in \mathbb{Z}$,
2. $|x_1 - x_2| = 1$ and $y_1 = y_2 = 2l$ for some $l \in \mathbb{Z}$;
3. $x_1 - x_2 = y_1 - y_2 = \pm 1$ and $x_1 - 2k = y_1$ for some $k \in \mathbb{Z}$,
4. $x_1 - x_2 = y_2 - y_1 = \pm 1$ and $x_1 = 2l - y_1$ for some $l \in \mathbb{Z}$.

A section of the adjacency graph (\mathbb{Z}^2, G) is shown in Fig. 5.

The following statement follows from [12], Theorem 12:

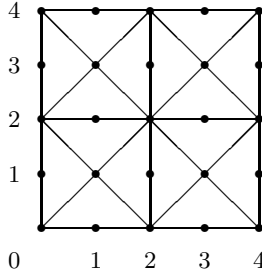


Fig. 5. A section of the adjacency graph (\mathbb{Z}^2, G)

Theorem 2. G is the quotient adjacency of any of the basic adjacencies on \mathbb{Z}^2 generated by the surjection $g : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ given as follows:

$$g(x, y) = \begin{cases} (2k, 2l) & \text{if } (x, y) \in \{(4k, 4l)\} \cup A_8(4k, 4l), \quad k, l \in \mathbb{Z}, \\ (2k, 2l + 1) & \text{if } (x, y) \in \{(4k + i, 4l + 2); i \in \{-1, 0, 1\}\}, \quad k, l \in \mathbb{Z}, \\ (2k + 1, 2l) & \text{if } (x, y) \in \{(4k + 2, 4l + j); j \in \{-1, 0, 1\}\}, \quad k, l \in \mathbb{Z}, \\ (2k + 1, 2l + 1) & \text{if } (x, y) = (4k + 2, 4l + 2), \quad k, l \in \mathbb{Z}. \end{cases}$$

Let H be the adjacency on \mathbb{Z}^2 defined as follows:

For any $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}^2, \{(x_1, y_1), (x_2, y_2)\} \in H$ if and only if one of the following four conditions is fulfilled:

1. $|y_1 - y_2| = 1$ and $x_1 = x_2 = k$ for some $k \in \mathbb{Z}$,
2. $|x_1 - x_2| = 1$ and $y_1 = y_2 = l$ for some $l \in \mathbb{Z}$;
3. $x_1 - x_2 = y_1 - y_2 = \pm 1$ and $x_1 - 4k = y_1$ for some $k \in \mathbb{Z}$,
4. $x_1 - x_2 = y_2 - y_1 = \pm 1$ and $x_1 = 4l - y_1$ for some $l \in \mathbb{Z}$.

A section of the adjacency graph (\mathbb{Z}^2, H) is shown in Fig. 6.

The following statement follows from [12], Theorem 14:

Theorem 3. H is the quotient adjacency of any of the basic adjacencies on \mathbb{Z}^2 generated by the surjection $h : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ given as follows:

$$h(x, y) = \begin{cases} (2k + 2l + 1, 2l - 2k + 1) & \text{if } (x, y) \in \{(4k, 4l + 2)\} \cup A_4(4k, 4l + 2), \\ & k, l \in \mathbb{Z}, \\ (2k + 2l + 1, 2l - 2k - 1) & \text{if } (x, y) \in \{(4k + 2, 4l)\} \cup A_4(4k + 2, 4l), \\ & k, l \in \mathbb{Z}, \\ (\frac{x+y}{2}, \frac{y-x}{2}) & \text{if } x, y \text{ are odd or } (x, y) = (4k + 2l, 2l), \quad k, l \in \mathbb{Z}. \end{cases}$$

Remark 2. It follows from [12], Theorem 10, that also the Khalimsky adjacency is a quotient adjacency of any of the basic adjacencies on \mathbb{Z}^2 , namely that one generated by the surjection $f : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ given as follows:

$$f(x, y) = \begin{cases} (2k, 2l) & \text{if } (x, y) = (4k, 4l), \quad k, l \in \mathbb{Z}, \\ (2k, 2l + 1) & \text{if } (x, y) \in A_4(4k, 4l + 2), \quad k, l \in \mathbb{Z}, \\ (2k + 1, 2l) & \text{if } (x, y) \in A_4(4k + 2, 4l), \quad k, l \in \mathbb{Z}, \\ (2k + 1, 2l + 1) & \text{if } (x, y) \in \{(4k + 2, 4l + 2)\} \cup (A_8(4k + 2, 4l + 2) - \\ & A_4(4k + 2, 4l + 2)), \quad k, l \in \mathbb{Z}. \end{cases}$$

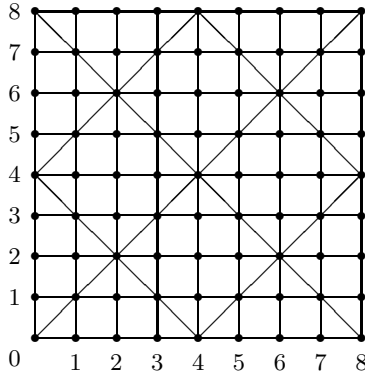


Fig. 6. A section of the adjacency graph (\mathbb{Z}^2, H)

It may easily be seen that the surjections g and h from Theorems 2 and 3, respectively, have the property that $g^{-1}(\{y\})$ and $h^{-1}(\{y\})$ are connected in (\mathbb{Z}^2, A) for every point $y \in \mathbb{Z}^2$. Therefore, using Lemma 1 and Theorems 1-3, we may identify Jordan curves among the simple closed curves in the adjacency graphs (\mathbb{Z}^2, G) and (\mathbb{Z}^2, H) .

Theorem 4. *Let D be a simple closed curve in (\mathbb{Z}^2, G) having more than four points and such that every pair of different points $z_1, z_2 \in D$ with both coordinates even satisfies $A_4(z_1) \cap A_4(z_2) \subseteq D$. Then D is a Jordan curve in (\mathbb{Z}^2, G) .*

Proof. It may easily be seen that there is precisely one cycle C in the square-diagonal graph satisfying $g(C) = D$. Let (\mathbb{Z}^2, E) be the first of the three sd -adjacency graphs from Fig. 4. By Theorem 1, C is a Jordan curve in (\mathbb{Z}^2, E) . C consists of the center points of the sets $g^{-1}(z)$, $z \in C$, and the points laying between the pairs of center points of the sets $g^{-1}(z_1)$ and $g^{-1}(z_2)$ where $z_1, z_2 \in C$ are adjacent points in (\mathbb{Z}^2, G) (clearly, for every pair of points $z_1, z_2 \in C$ adjacent in (\mathbb{Z}^2, G) , there is precisely one point lying between the center points of $g^{-1}(z_1)$ and $g^{-1}(z_2)$ - it is the only point adjacent to each of the two center points in the square-diagonal graph). Since D is a simple closed curve in (\mathbb{Z}^2, G) , we have $\text{card}(\{(x + i, y + j); i, j \in \{-2, 2\}\} \cap C) = 2$ for every point $(x, y) \in C$ with $x = 4k + 2$ and $y = 4l + 2$ for some $k, l \in \mathbb{Z}$. Further, we have $\text{card } C > 8$ because D has more than four points. The fact that every pair of different points $z_1, z_2 \in g(C)$ with both coordinates even satisfies $A_4(z_1) \cap A_4(z_2) \subseteq g(C)$ implies that $(4k, 4l + 2) \in C$ whenever $(4k, 4l), (4k, 4l + 4) \in C$ ($k, l \in \mathbb{Z}$) and $(4k + 2, 4l) \in C$ whenever $(4k, 4l), (4k + 4, 4l) \in C$ ($k, l \in \mathbb{Z}$).

Let C_1, C_2 be the two components of the induced subgraph $\mathbb{Z}^2 - C$ of (\mathbb{Z}^2, E) and put $C'_i = C_i - g^{-1}(D)$ for $i = 1, 2$. Since C has more than four points, we have $C'_i \neq \emptyset$ for $i = 1, 2$. Let $(x, y) \in D$ be a point and write $g^{-1}(x, y)$ briefly instead of $g^{-1}(\{(x, y)\})$. Clearly, there exists a point $(x, y) \in D$ with x or y even.

(1) Suppose that x is even and y is odd. Then $g^{-1}(x, y) = \{(2x + k, 2y); k \in \{-1, 0, 1\}\}$ where $(2x, 2y) \in C$ and there is $i \in \{1, 2\}$ such that $(2x - 1, 2y) \in C_i$ and $(2x + 1, 2y) \in C_{3-i}$. We also have $\{(x, y - 1), (x, y + 1)\} \subseteq D$ and,

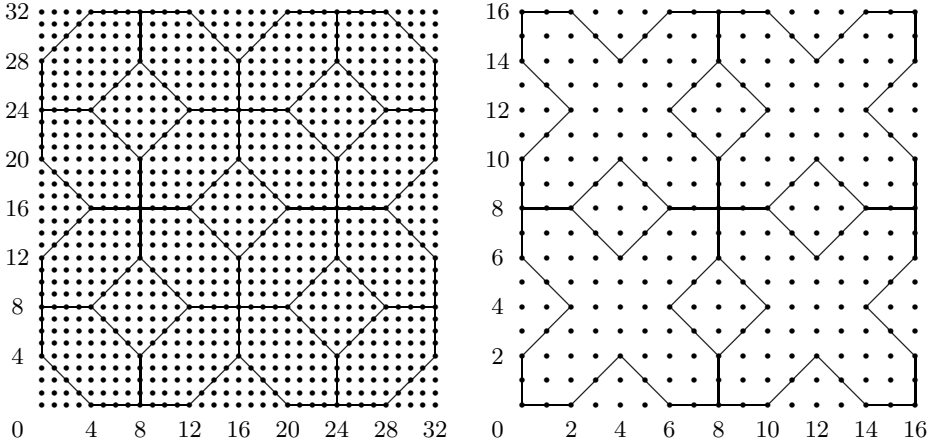


Fig. 7. Portions of two subgraphs of the adjacency graph (\mathbb{Z}^2, G)

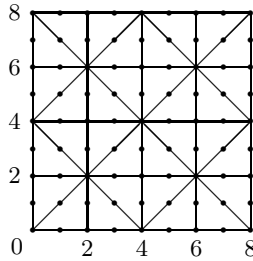


Fig. 8. A portion of a subgraph of the adjacency graph (\mathbb{Z}^2, H)

consequently, $\{(2x - 1, 2y - 1), (2x - 1, 2y + 1)\} \subseteq C_i$ and $\{(2x + 1, 2y - 1), (2x + 1, 2y + 1)\} \subseteq C_{3-i}$. Suppose that D contains exactly one of the points $(x - 2, y - 1)$ and $(x - 2, y + 1)$, say $(x - 2, y - 1)$. Then $(x - 2, y + 1) \notin D$ implies $g^{-1}(x - 2, y + 1) = A_8(2x - 4, 2y + 2) \cup \{(2x - 4, 2y + 2)\} \subseteq C'_i$. It may easily be seen that every point of C'_i may be joined with $(2x - 4, 2y + 2)$ by a path in (\mathbb{Z}^2, E) contained in C'_i . It follows that C'_i is connected. Further, we have $(x + 2, y - 1) \notin D$ because, otherwise, $\{(x + 1, y - 1)\} = A_4(x, y - 1) \cap A_4(x + 2, y - 1) \subseteq D$, so that D would contain three points adjacent to $(x, y - 1)$ in (\mathbb{Z}^2, G) , namely $(x - 1, y - 1)$, (x, y) and $(x + 1, y - 1)$, which is impossible. Thus, we have $g^{-1}(x + 2, y - 1) = A_8(2x + 4, 2y - 2) \cup \{(2x + 4, 2y - 2)\} \subseteq C'_{3-i}$. It may easily be seen that every point of C'_{3-i} may be joined with $(2x + 4, 2y - 2)$ by a path in (\mathbb{Z}^2, E) contained in C'_{3-i} . It follows that C'_{3-i} is connected.

(2) If x is odd and y is even, then the situation is analogous to the previous one.

(3) Suppose that both x and y are even. Then, by the assumptions of the statement, $D \cap A_4(x, y) = \emptyset$ and the set $A_8(x, y) - (A_4(x, y) \cup D)$ has precisely

two points $z = (p, q)$ and $z' = (p', q')$ such that $p \neq p'$ and $q \neq q'$. It follows that there is $i \in \{1, 2\}$ such that $g^{-1}(p, q) = A_8(2p, 2q) \cup \{(2p, 2q)\} \subseteq C'_i$ and $g^{-1}(p', q') = A_8(2p', 2q') \cup \{(2p', 2q')\} \subseteq C'_{3-i}$. Using arguments similar to those used in (1), we may show that C'_i and C'_{3-i} are connected.

By (1)-(3), C'_i is connected for $i = 1, 2$. We clearly have $g(C_1) \cap g(C_2) = \emptyset$ (because, otherwise, there is a point $y \in g(C_1) \cap g(C_2)$, which means that $g^{-1}(\{z\}) \cap C_1 \neq \emptyset \neq g^{-1}(\{z\}) \cap C_2$ - this is a contradiction because $g^{-1}(\{z\})$ is connected). Therefore, $g(C'_1) \cap g(C'_2) = \emptyset$. This yields $C'_i = g^{-1}(g(C'_i))$ for $i = 1, 2$, hence $g(C'_i)$ is connected for $i = 1, 2$ by Lemma 1 and Theorem 2. Suppose that $\mathbb{Z}^2 - D$ is connected. Then $g^{-1}(\mathbb{Z}^2 - D) = C'_1 \cup C'_2$ is connected by Lemma 1 and Theorem 2. This is a contradiction because $\emptyset \neq C'_i \subseteq C_i$ for $i = 1, 2$, C_1 and C_2 are disjoint and $C_1 \cup C_2$ is not connected. Therefore, $\mathbb{Z}^2 - D = g(C'_1) \cup g(C'_2)$ is not connected and, consequently, $g(C'_1)$ and $g(C'_2)$ are components of the induced subgraph $\mathbb{Z}^2 - D$ of (\mathbb{Z}^2, G) . We have shown that D is a Jordan curve in (\mathbb{Z}^2, v) .

Example 1. By Theorem 4, every cycle in any of the two graphs sections of which are shown in Fig. 7 is a Jordan curve in (\mathbb{Z}^2, G) .

The following statement may be proved similarly to Theorem 4. But it may also be obtained as a consequence of Theorem 8 from [13]:

Theorem 5. *Every simple closed curve D in (\mathbb{Z}^2, H) which is a cycle in the graph a section of which is shown in Fig. 8 is a Jordan curve in (\mathbb{Z}^2, H) .*

5 Conclusion

The *sd*-adjacencies introduced and studied proved to provide graphs on the vertex set \mathbb{Z}^2 having certain natural cycles as Jordan curves, namely arbitrary cycles in the square-diagonal graph. An advantage of these Jordan curves over the Jordan curves with respect to the Khalimsky topology on \mathbb{Z}^2 is that they may turn, at some points, to form the acute angle $\frac{\pi}{4}$. From this point of view, the *sd*-adjacencies provide a more convenient structure on \mathbb{Z}^2 than the Khalimsky topology for the study of digital images. Another advantage of the *sd*-adjacencies is that some of their quotient adjacencies may also be used in digital topology to structure the digital plane because they possess rich enough varieties of Jordan curves (a Jordan curve theorem for each of the quotient adjacencies may be obtained by using the Jordan curve theorem proved for the *sd*-adjacencies). A rich variety of Jordan curves is one of the most important criteria of convenience of a structure on \mathbb{Z}^2 for the study of digital pictures because Jordan curves represent borders of picture regions (imaged objects). Therefore, the results presented in this note may be useful for solving the problems of digital image processing that are closely related to borders such as contour filling, thinning, pattern recognition, etc.

Acknowledgement. The author acknowledges partial support from the Specific Research Grant Agency, Brno University of Technology, project no. FSI-S-11-3.

References

1. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer (2008)
2. Khalimsky, E.D., Kopperman, R., Meyer, P.R.: Computer graphics and connected topologies on finite ordered sets. *Topology Appl.* 36, 1–17 (1990)
3. Khalimsky, E.D., Kopperman, R., Meyer, P.R.: Boundaries in digital planes. *Journ. Appl. Math. Stoch. Anal.* 3, 27–55 (1990)
4. Kiselman, C.O.: Digital Jordan Curve Theorems. In: Nyström, I., Sanniti di Baja, G., Borgefors, G. (eds.) DGCI 2000. LNCS, vol. 1953, pp. 46–56. Springer, Heidelberg (2000)
5. Kong, T.Y., Kopperman, R., Meyer, P.R.: A topological approach to digital topology. *Amer. Math. Monthly* 98, 902–917 (1991)
6. Kopperman, R., Meyer, P.R., Wilson, R.G.: A Jordan surface theorem for three-dimensional digital spaces. *Discr. Comput. Geom.* 6, 155–161 (1991)
7. Rosenfeld, A.: Digital topology. *Amer. Math. Monthly* 86, 621–630 (1979)
8. Rosenfeld, A.: *Picture Languages*. Academic Press, New York (1979)
9. Šlapal, J.: Closure operations for digital topology. *Theor. Comp. Sci.* 305, 457–471 (2003)
10. Šlapal, J.: A digital analogue of the Jordan curve theorem. *Discr. Appl. Math.* 139, 231–251 (2004)
11. Šlapal, J.: Digital Jordan curves. *Topology Appl.* 153, 3255–3264 (2006)
12. Šlapal, J.: A quotient universal digital topology. *Theor. Comp. Sci.* 405, 164–175 (2008)
13. Šlapal, J.: Jordan Curve Theorems with Respect to Certain Pretopologies on \mathbb{Z}^2 . In: Brlek, S., Reutenauer, C., Provençal, X. (eds.) DGCI 2009. LNCS, vol. 5810, pp. 252–262. Springer, Heidelberg (2009)
14. Šlapal, J.: Convenient Closure Operators on \mathbb{Z}^2 . In: Wiederhold, P., Barneva, R.P. (eds.) IWCIA 2009. LNCS, vol. 5852, pp. 425–436. Springer, Heidelberg (2009)
15. Šlapal, J.: A Jordan Curve Theorem in the Digital Plane. In: Aggarwal, J.K., Barneva, R.P., Brimkov, V.E., Koroutchev, K.N., Korutcheva, E.R. (eds.) IWCIA 2011. LNCS, vol. 6636, pp. 120–131. Springer, Heidelberg (2011)
16. Šlapal, J.: A digital pretopology and one of its quotients. *Topology Proc.* 39, 13–25 (2012)

On Topology Preservation for Triangular Thinning Algorithms

Péter Kardos and Kálmán Palágyi

Department of Image Processing and Computer Graphics,
University of Szeged, Hungary
{pkardos,palagyi}@inf.u-szeged.hu

Abstract. Thinning is a frequently used strategy to produce skeleton-like shape features of binary objects. One of the main problems of parallel thinning is to ensure topology preservation. Solutions to this problem have been already given for the case of orthogonal and hexagonal grids. This work introduces some characterizations of simple pixels and some sufficient conditions for parallel thinning algorithms working on triangular grids (or hexagonal lattices) to preserve topology.

Keywords: Triangular grids, Topology preservation, Thinning.

1 Introduction

The concept of skeletonization serves as a useful shape descriptor in various areas of image processing and pattern recognition [13]. Thinning is a widely used iterative technique for producing the skeletons of binary objects [8,16]. Thinning algorithms are composed of reductions (i.e., some object points having value of “1” in a binary picture that satisfy certain topological and geometric constraints are changed to “0” ones simultaneously).

A fundamental requirement of these algorithms is topology preservation [7]. At first, sufficient conditions for the topological correctness of reduction operators working on orthogonal grids were proposed by Ronse and Kong [6,12].

2D digital pictures on hexagonal and triangular grids have been studied by a number of authors [7,10]. There were also various thinning algorithms working on hexagonal and triangular grids proposed in [1,2,5,14,15,17]. For the hexagonal case, Kardos and Palágyi established some sufficient conditions for topology preserving reductions [5]. A triangular grid, which is formed by a tessellation of regular triangles, corresponds, by duality, to the hexagonal lattice, where the points are the centers of that triangles, see Fig. 1. The geometry of triangular grids has been already investigated (see for example [3,11]), however, their topological properties have been poorly dealt with.

In this paper we study reductions on triangular grids in the view of topology preservation. For this purpose, first we discuss some characterizations of so-called simple pixels which play a key role in this field, and we also present some sufficient conditions for topology preserving reductions. Our result can be applied to construct topologically correct triangular parallel thinning algorithms.

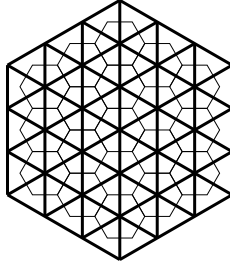


Fig. 1. A triangular grid and the hexagonal lattice dual to it. Triangular pixels are bounded by thick line segments. Pixel centers are joined with thin line segments.

The rest of this paper is organized as follows. Section 2 briefly introduces some basic notions of digital topology. Then, in Section 3 we introduce two kinds of characterizations of simple pixels. Section 4 discusses our sufficient conditions mentioned above. To illustrate the usefulness of these conditions, we define a pair of reductions in Section 5 and validate its topological correctness.

2 Basic Notions and Results

We use the fundamental concepts of digital topology as reviewed by Kong and Rosenfeld [7].

Let us consider the *digital space* V , and let us call the elements of V *pixels*. We refer with the notation $N_k(p)$ to the set of pixels that are k -adjacent to pixel $p \in V$ and let $N_k^*(p) = N_k(p) \setminus \{p\}$. Note that reflexive and symmetric adjacency relations are generally considered (i.e., $p \in N_k(p)$ and if $q \in N_k(p)$, then $p \in N_k(q)$). The sequence of distinct pixels $\langle x_0, x_1, \dots, x_s \rangle$ is called a k -path of length s from pixel x_0 to pixel x_s in a non-empty set of pixels $X \subseteq V$ if each pixel of the sequence is in X and x_i is k -adjacent to x_{i-1} ($i = 1, \dots, s$). Note that a single pixel is a k -path of length 0. In the special case when $x_0 = x_n$, we call the k -path an k -cycle. Two pixels $p, q \in V$ are said to be k -connected in the set $X \subseteq V$ if there is a k -path from p to q in X . A set of pixels $X \subseteq V$ is k -connected in the set of pixels $Y \supseteq X$ if any two pixels in X are k -connected in Y . A set of pixels $X \subseteq V$ is a k -component in the set of pixels $Y \supseteq X$ if X is k -connected in Y , but the set $X \cup \{y\}$ is not k -connected in Y for any $y \in Y \setminus X$ (i.e., X is a maximal k -connected set of pixels in Y).

An (m, n) *digital picture* is a quadruple $\mathcal{P} = (V, m, n, B)$. Each pixel in $B \subseteq V$ is called a *black pixel* and has a value of 1 assigned to it. Picture \mathcal{P} is finite if it contains finitely many black pixels. Each pixel in $V \setminus B$ is called a *white pixel* and has a value of 0 assigned to it. Adjacency relation m is assigned to black pixels, and a *black component* or an *object* is an m -connected set of pixels in B . Adjacency relation n is assigned to white pixels, and a *white component* is an n -connected set of pixels in $V \setminus B$. In a finite picture there is a unique white component that is called the *background*. A finite white component is called a *cavity*.

A *reduction* transforms a binary picture only by changing some black pixels to white ones (which is referred to as the deletion of 1's). A 2D reduction does *not* preserve topology [6] if any black component is split or is completely deleted, any white component is merged with another white component, or a new white component is created.

A black pixel is called a *border pixel* in an (m, n) picture if it is m -adjacent to at least one white pixel. A *simple pixel* is a black pixel whose deletion is a topology preserving reduction [7]. Let \mathcal{P} be an (m, n) picture. The set of black pixels $D = \{d_1, \dots, d_k\}$ is called a *simple set* of \mathcal{P} if D can be arranged in a sequence $\langle d_{i_1}, \dots, d_{i_k} \rangle$ in which d_{i_1} is simple and each d_{i_j} is simple after $\{d_{i_1}, \dots, d_{i_{j-1}}\}$ is deleted from \mathcal{P} , for $j = 2, \dots, k$. (By definition, let the empty set be simple.)

In this paper our attention is focussed on pictures sampled on the triangular grid denoted by T (see Fig. 1). Each pixel in T is a regular triangle and a point in the hexagonal lattice is associated to it [10]. Two kinds of adjacency relations have been considered in the triangular grid T : two triangles (pixels) are *3-adjacent* if they share an edge, and two triangles are *12-adjacent* if they share at least one vertex. It is easy to see that $N_3(p) \subset N_{12}(p)$ for any $p \in T$. The considered adjacencies on the hexagonal lattice is shown in Fig. 2. The set composed by six pairwise 12-adjacent pixels of T is called a *unit hexagon* (see Fig. 2c).

In order to avoid connectivity paradoxes [7] and verify the discrete Jordan's theorem [10], $(m, n) = (12, 3)$ and $(m, n) = (3, 12)$ pictures are considered on the triangular grid T (where $m \neq n$). That is why we intruduce characterizations of simple points and give sufficient conditions for topology preserving reductions for $(12, 3)$ and $(3, 12)$ pictures in the next two sections.

3 Characterizations of Simple Pixels

If we want to verify whether a reduction preserves topology or not, first we must be able to determine, which pixels in an object are simple. For this aim, we present some characterizations of simple pixels in both $(12, 3)$ and $(3, 12)$ pictures.

Theorem 1. *Let p a black pixel in a picture (T, m, n, B) ($(m, n) = (12, 3), (3, 12)$). Pixel p is simple if and only if all the following conditions are satisfied:*

1. *Pixel p is m -adjacent to exactly one m -component of $N_{12}^*(p) \cap B$.*
2. *Pixel p is n -adjacent to exactly one n -component of $N_{12}(p) \setminus B$.*

Proof. First, we prove that if p is simple in picture (T, m, n, B) , then Conditions 1 and 2 are fulfilled. Let us denote $C(p)$ and $\overline{C}(p)$ the number of m -components of $N_{12}^*(p) \cap B$ and the number of n -components of $N_{12}(p) \setminus B$, respectively. Thus we want to prove that $C(p) = \overline{C}(p) = 1$. We give an indirect proof for this, therefore, let us suppose that p is simple but at least one of Conditions 1 and 2 does not hold.

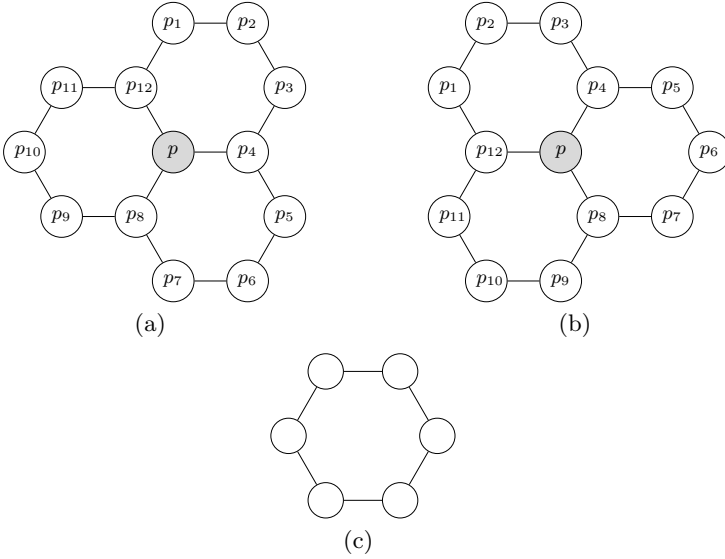


Fig. 2. Indexing schemes for $N_3(p) = \{p_4, p_8, p_{12}\}$ and $N_{12}(p) = \{p_1, \dots, p_{12}\}$ (a)-(b), and a unit hexagon (c)

First, we will see that in this case none of Conditions 1 and 2 is fulfilled, i.e., there exist some pixels $p_{i_1}, p_{i_2} \in N_{12}^*(p) \cap B$ and $p_{j_1}, p_{j_2} \in N_{12}(p) \setminus B$ such that p_{i_1} and p_{i_2} are not m -connected in $N_{12}^*(p) \cap B$, and the pixels p_{j_1}, p_{j_2} are not n -connected in $N_{12}(p) \setminus B$:

- The assumption $C(p) = 0$ implies that p is an isolated pixel, which contradicts the simplicity of p since an object is completely deleted if an isolated point is deleted. Hence, $C(p) \neq 0$.
 - The relationship $\bar{C}(p) = 0$ means that p is not a border pixel, thus again p could not be simple since deletion of a non-border pixel creates a new cavity. Therefore, $\bar{C}(p) \neq 0$.
 - Let us examine the case when $C(p) \geq 2$. This directly implies that there exist two pixels $p_{i_1}, p_{i_2} \in B$ such that p_{i_1} and p_{i_2} are not m -connected in $N_{12}^*(p) \cap B$.
 - If $(m, n) = (12, 3)$, then p must have two white 3-neighbors, or else any pixel in $N_{12}^*(p) \cap B$ would be 12-adjacent to at least one black 3-neighbor of p , and as any two 3-neighbors of p are 12-adjacent, as well, p_{i_1} and p_{i_2} would be 12-connected in $N_{12}^*(p) \cap B$. Let us denote the mentioned two white 3-neighbors p_{j_1} and p_{j_2} . For the sake of clarity, let for example $p_{j_1} = p_4, p_{j_2} = p_8$. (Any other possible case could be examined similarly because of the symmetrical structure of $N_{12}(p)$.)
- If p_{12} is a black pixel, then at least one of the pixels p_5, p_6 , and p_7 must be black, and if p_{12} is white, then at least two of the sets $\{p_1, p_2, p_3\}$, $\{p_5, p_6, p_7\}$, and $\{p_9, p_{10}, p_{11}\}$ must contain a black pixel, or else p would

be 12-adjacent to exactly one black component in $N_{12}^*(p) \cap B$, which is a contradiction with our assumption. Hence, p_4 and p_8 are not 3-connected in $N_{12}^*(p) \setminus B$. Consequently, there exist two black pixels $q, r \in N_{12}^*(p) \cap B$ such that q and r are not 12-adjacent in $N_{12}^*(p) \cap B$ and they are not contained in the same 3-path between p_{j_1} and p_{j_2} in $N_{12}^*(p)$, and thus, p_{j_1} and p_{j_2} are not contained in the same 3-path between q and r in $N_{12}^*(p)$. Without loss of generality, let $p_{i_1} = q$ and $p_{i_2} = r$.

- If $(m, n) = (3, 12)$, then p_{i_1} and p_{i_2} are 3-adjacent to p . Let, for example, $p_{i_1} = p_4$ and $p_{i_2} = p_8$. Each of the sets $\{p_3, p_5\}$ and $\{p_7, p_9\}$ contains at least one black pixel, or else a black component would be split by the removal of p . If p_{12} is a white pixel, then at least one of the pixels p_5, p_6 , and p_7 must be white, and if p_{12} is black, then at least two sets of $\{p_1, p_2, p_3\}$, $\{p_5, p_6, p_7\}$, and $\{p_9, p_{10}, p_{11}\}$ must contain a white pixel, or else p would be 3-adjacent to exactly one 3-component of $N_{12}^*(p) \cap B$. By the latter four relationships it can be shown that p is 12-adjacent to at least two white 12-components of $N_{12}^*(p) \setminus B$. This implies that there exist two white pixels $q, r \in N_{12}^*(p) \setminus B$ such that q and r are not 12-adjacent in $N_{12}^*(p) \setminus B$ and they are not contained in the same 3-path between p_{i_1} and p_{i_2} in $N_{12}^*(p)$. It is also easy to see that q and r are not 12-connected in $N_{12}^*(p) \setminus B$. Without loss of generality, let $p_{j_1} = q$ and $p_{j_2} = r$.
- Now, let us consider the case when $\overline{C}(p) \geq 2$, i.e., there exist two pixels $p_{j_1}, p_{j_2} \in T \setminus B$ such that p_{j_1} is not n -connected to p_{j_2} in $N_{12} \setminus B$. We can get a similar result to the one derived in the previous case, if we follow the same train of thoughts for the situations $(m, n) = (3, 12)$ and $(m, n) = (12, 3)$, as in the previous case for $(m, n) = (12, 3)$ and $(m, n) = (3, 12)$, respectively. To do this, we only need to interchange the adjectives “white” and “black”, the set notations $N_{12}^*(p) \cap B$ and $N_{12}(p) \setminus B$, and the indices i_1, i_2 and j_1, j_2 , respectively.

Let us take such pixels $p_{i_1}, p_{i_2} \in N_{12}^*(p) \cap B$ and $p_{j_1}, p_{j_2} \in N_{12}(p) \setminus B$ that have the property shown above. Obviously, p_{i_1} and p_{i_2} are m -connected in $B \setminus \{p\}$, furthermore, p_{j_1} and p_{j_2} are n -connected in $T \setminus B$, or else a black component would be split by the removal of p , or two white components would be merged, which contradicts the simplicity of p .

Let P_1 be the set of the pixels of an m -cycle that we get if we extend an m -path between p_{i_1} and p_{i_2} with p , furthermore, let P_2 be the set of the pixels of an n -cycle that we get if we extend an n -path between p_{j_1} and p_{j_2} in $T \setminus B$ with p .

It is easy to see that picture $(T, n, m, T \setminus P_1)$ contains two objects. Due to the assumptions on p_{j_1} and p_{j_2} , it follows that p_{j_1} and p_{j_2} are not n -connected in $T \setminus P_1$ (i.e., p_{j_1} and p_{j_2} are contained in distinct objects in picture $(T, n, m, T \setminus P_1)$), or else we came to a contradiction with our assumption that p_{j_1} and p_{j_2} are not contained in the same 3-path between p_{i_1} and p_{i_2} in $N_{12}^*(p)$.

An object of picture $(T, n, m, T \setminus P_1)$ must contain all elements of $P_2 \setminus \{p\}$, or else p_{j_1} and p_{j_2} would not be n -connected in $T \setminus B$, thus the number of white

components would be reduced by the removal of p . However, this leads to a contradiction with the above stated fact that p_{j_1} and p_{j_2} are not contained in the same object in $(T, n, m, T \setminus P_1)$.

Hence from this follows that if p is simple, then both of Conditions 1 and 2 hold.

Now, let us suppose that Conditions 1 and 2 are satisfied. We will see that the removal of p preserves topology:

- Let q_1 and q_2 be two black pixels in (T, m, n, B) such that q_1 and q_2 are both m -connected to p , i.e., these pixels are contained by the same black component. It is easy to verify that in this case, there exists an m -path in B between q_1 and q_2 that contains a subpath $\langle p_{i_1}, p, p_{i_2} \rangle$ ($i_1, i_2 \in \{1, 2, \dots, 12\}, i_1 \neq i_2$). However, by Condition 1, p_{i_1} and p_{i_2} are also m -connected in $N_{12}^*(p) \cap B$, which means that there is also an m -path between q_1 and q_2 that does not contain p . This implies that after the removal of p , q_1 and q_2 will still fall into the same black component, i.e., no black component is split by the removal of p .
- Now, let r_1 and r_2 be two white pixels belonging to distinct white components in (T, m, n, B) . If r_1 is n -connected to a pixel $p_j \in N_3(p) \setminus B$, then r_2 cannot be n -connected to any pixel in $N_{12}(p) \setminus B$, or else r_1 and r_2 would fall into the same white component, as, by Condition 2, p_j is n -connected to all pixels of $N_3(p) \setminus B$. Therefore, after the removal of p , r_1 and r_2 will still fall into two distinct white components, which means that no cavity is merged with the background nor with another cavity by the removal of p .
- Furthermore, no object is completely removed by the deletion of p , or else this would mean that p is an isolated object pixel, but this would contradict Condition 1. Even new cavities cannot be created by the removal of p , because this could only arise, if $N_m^*(p)$ contained only black pixels, which contradicts Condition 2.

By the above observations we can state that if Conditions 1 and 2 hold, then p is simple. □

Some configurations of simple and non-simple pixels in (12,3) pictures are shown in Fig. 3.

We remark that Theorem 4.1 in [7] states a similar relationship as above for (8,4) and (4,8) pictures sampled on the orthogonal grid \mathbb{Z}^2 .

The following lemma points out to a kind of a duality between (12,3) and (3,12) pictures concerning simple pixels.

Lemma 1. *Pixel p is simple in picture (T, m, n, B) ($(m, n) = (12, 3), (3, 12)$), if and only if p is simple in picture $(T, n, m, (T \setminus B) \cup \{p\})$.*

Proof. Let us suppose that p is simple in picture (T, m, n, B) . Note that we get picture $(T, n, m, T \setminus B)$ from (T, m, n, B) by recoloring all its pixels and switching the applied types of adjacency relation between black and white pixels. From this follows, that each white/black component of $(T, y, x, T \setminus B)$ coincides with a black/white component of (T, x, y, B) .

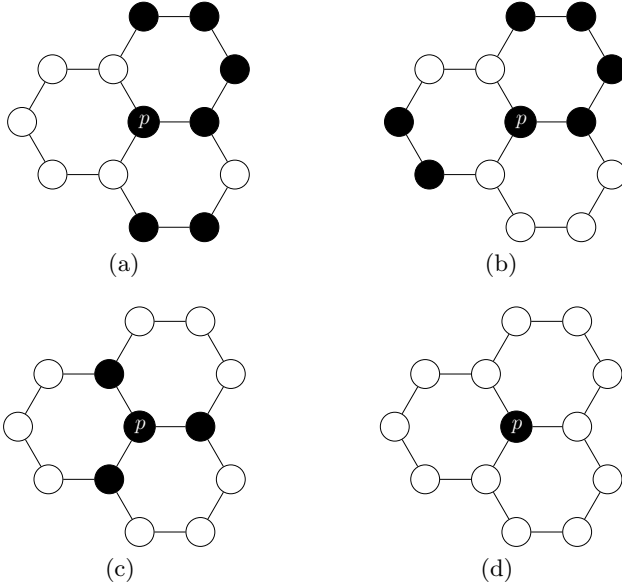


Fig. 3. Examples for a simple (a) and three non-simple (b-d) pixels in (12,3) pictures. Pixel p is not simple in (b) since its deletion may split an object (Condition 1 of Theorem 1 is violated). A white 3-component (singleton cavity) is created by deletion of p in (c) (Condition 2 of Theorem 1 is violated). Pixel p is also not simple in (d), since its deletion completely deletes a (singleton) object (Condition 1 of Theorem 1 is violated).

By Theorem 1, p is m -adjacent to exactly one m -component of $N_{12}^*(p) \cap B$ in (T, m, n, B) , which we will hereafter denote by C_1 . Furthermore, p is n -adjacent to exactly one n -component of $N_{12}(p) \setminus B$, which will be denoted by C_2 .

According to the above observations, it is easy to show that in picture $(T, n, m, T \setminus B)$ and thus in picture $(T, n, m, (T \setminus B) \cup \{p\})$ the white component coinciding with C_1 is the only m -component of $N_{12}(p) \setminus (T \setminus B)$ being m -adjacent to p , and the black component coinciding with C_2 is the only n -component of $N_{12}^*(p) \cap (T \setminus B)$ being n -adjacent to p . Hence, p is simple in picture $(T, n, m, (T \setminus B) \cup \{p\})$, as well.

The proof in the opposite direction can be carried out similarly. □

For a better understanding of the concept of Lemma 1, let us again examine the configurations in Fig. 3. It can be easily verified that if we “invert” those configurations by switching the black and white pixels in $N_{12}(p)$, then we get some other configurations such that the first example (“inverted” version of Fig. 3a) represents a simple pixel in a (3,12) picture, and the remaining three examples (“inverted” versions of Figs. 3b-d) refer to non-simple pixels in (3,12) pictures.

Using the relationship formulated in Theorem 1 to check simple pixels may not be convenient for implementational purposes. Here we give some configurations

(so-called matching templates) to decide whether an object pixel p is simple or not, which make possible an efficient implementation of the verification of simplicity. We define a *matching template* as a pair (τ_b, τ_w) , where τ_b and τ_w are both subsets of the set $\{1, 2, \dots, 12\}$ such that $\tau_b \cap \tau_w = \emptyset$. We say that an object pixel p in a picture (T, m, n, B) $((m, n) = (12, 3), (3, 12))$ *matches* a matching template $\tau = (\tau_b, \tau_w)$, if the following two conditions hold:

- if $k \in \tau_b$ ($k \in \{1, 2, \dots, 12\}$), then $p_k \in N_{12}^*(p) \cap B$, and
- if $k \in \tau_w$ ($k \in \{1, 2, \dots, 12\}$), then $p_k \in N_{12}^*(p) \setminus B$.

In the lattice representation of the matching template $\tau = (\tau_b, \tau_w)$ with central pixel p , p_k is depicted in black (white) if $k \in \tau_b$ ($k \in \tau_w$), furthermore, p_k is denoted by “.” if $k \notin \tau_b \cup \tau_w$.

Let $\mathcal{T}_i^{(m,n)}$ denote the set of matching templates composed by the matching template $T_i^{(m,n)}$ in Figs. 4-5 and its rotations by $k \cdot 60$ degrees $((m, n) = (12, 3), (3, 12); i = 1, 2, 3; k = 1, \dots, 5)$, and let $\mathcal{T}^{(m,n)} = \mathcal{T}_1^{(m,n)} \cup \mathcal{T}_2^{(m,n)} \cup \mathcal{T}_3^{(m,n)}$.

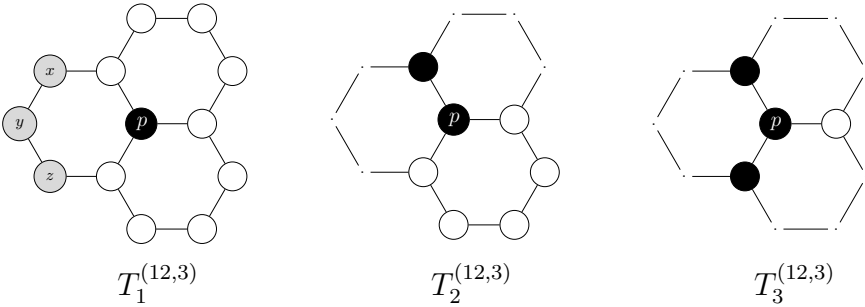


Fig. 4. Matching templates for characterizing the simplicity of a pixel p in $(12,3)$ pictures. Notations: each position marked \bullet matches a black pixel; \circ matches a white point; at least one of the pixels x, y , and z is black; positions denoted by “.” refer to pixels of arbitrary color.

Theorem 2. *A black pixel p is simple in an (m, n) picture, if and only if it matches an element of the set of matching templates $\mathcal{T}^{(m,n)}$ $((m, n) = (12, 3), (3, 12))$.*

Proof. It can be easily verified that we can get the templates of $\mathcal{T}_{4-i}^{(3,12)}$ ($i = 1, 2, 3$) from the elements of $\mathcal{T}_i^{(12,3)}$ by recoloring their pixels and by switching the types of adjacency relations applied on their black and white pixels. Hence, by Lemma 1 it is sufficient to carry out the proof for $(m, n) = (12, 3)$.

By observing the possible configurations we can state that if p has more than one white 3-neighbors, then there exists a white 3-path between them, and any pixel coinciding with a position denoted by “.” is 12-connected with any black pixel. From these follows that p represents a simple pixel in each of the mentioned templates.

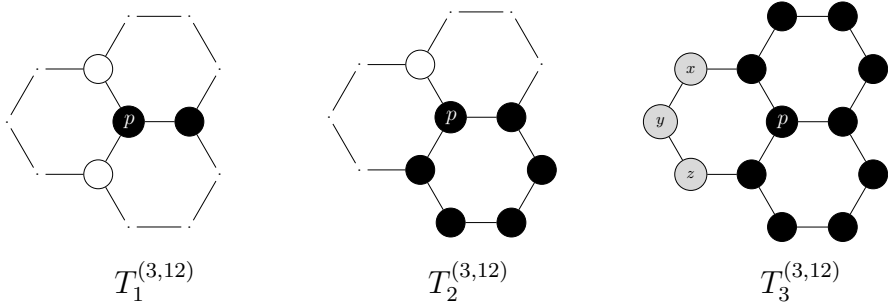


Fig. 5. Matching templates for characterizing the simplicity of a pixel p in $(3,12)$ pictures. Notations: each position marked \bullet matches a black pixel; \circ matches a white point; at least one of the pixels $x, y,$ and z is white; positions denoted by “.” refer to pixels of arbitrary color.

Let us suppose that p is simple but it does not match any template of $\mathcal{T}_i^{(12,3)}$ ($i = 1, 2, 3$). Even in this case, p must be 3-adjacent to at least one white pixel, or else p would not be simple. Hence, the neighborhood of p differs in some pixels from the ones shown in $\mathcal{T}_i^{(12,3)}$. If we changed the color of a white pixel in $N_{12}^*(p) \setminus N_3^*(p)$ to black, then this would obviously break the 3-connectedness of two white 3-neighbors of p , which, by Theorem [II](#), would cause that p is not simple after the recoloring. Furthermore, if we changed every black pixels in $\{x, y, z\}$ to white in the templates of $\mathcal{T}_1^{(12,3)}$, then p would be an isolated object pixel, i.e., a non-simple pixel. Hence, if p does not match any of the templates of $\mathcal{T}^{(12,3)}$, then p is not simple. \square

4 Sufficient Conditions for Topology Preserving Reductions

Based on Theorems 1 and 2 we present some sufficient conditions for topology preservation on both $(12,3)$ and $(3,12)$ pictures.

Before we formulate the main theorem of this paper, first we provide two important relationships.

Lemma 2. *Let \mathcal{O} be a reduction, and let S be the set of black pixels removed by \mathcal{O} from picture $\mathcal{P} = (T, m, n, B)$ ($(m, n) = (12, 3), (3, 12)$). \mathcal{O} is topology-preserving, if for any pixel $p \in S$ and for any set $Q \subseteq S \cap N_{12}^*(p)$, p is simple in picture $(T, m, n, B \setminus Q)$.*

Proof. We note that the proof of the alternative version of this lemma for $(6,6)$ pictures given in Lemma [2](#) of [5](#) does not rely on any special property of the hexagonal grids, therefore the proof for the triangular case can be done exactly the same way, only some notations will change. \square

Lemma 3. *Let us suppose that p and q are simple pixels in picture (T, m, n, B) , where $(m, n) = (12, 3), (3, 12)$. Then the following statements are equivalent:*

1. *Pixel q is simple in picture $(T, m, n, B \setminus \{p\})$.*
2. *Pixel p is simple in picture $(T, m, n, B \setminus \{q\})$.*

Proof. Let $x, y \in \{p, q\}$, where $x \neq y$. Let us suppose that x is simple in picture $(T, m, n, B \setminus \{y\})$. From this follows that $\{p, q\}$ is a simple set. If y would not be a simple pixel in $(T, m, n, B \setminus \{x\})$, then this would imply that the removal of y is not a topology-preserving reduction in $(T, m, n, B \setminus \{x\})$, thus even the removal of $\{x, y\} = \{p, q\}$ is not topology-preserving in (T, m, n, B) . However, this would contradict the simplicity of $\{p, q\}$. \square

Now we are ready to discuss our sufficient conditions for topology preservation.

Theorem 3. *A reduction \mathcal{O} is topology-preserving in picture $\mathcal{P} = (T, m, n, B)$ $((m, n) = (12, 3), (3, 12))$, if all of the following conditions hold:*

1. *Only simple pixels are deleted by \mathcal{O} .*
2. *If \mathcal{O} removes two n -adjacent pixels p and q , then p is simple in $(T, m, n, B \setminus \{q\})$, or q is simple in $(T, m, n, B \setminus \{p\})$.*
3. *\mathcal{O} does not delete completely any object contained in a unit hexagon.*

Proof. Let us suppose that \mathcal{O} fulfills Conditions 1-3, and let us denote by S the set of black pixels removed by \mathcal{O} . By Condition 1, each member of S is a simple pixel. By Lemma 2 it is sufficient to show that for any pixel $p \in S$ and for any set $Q \subseteq S \cap N_{12}^*(p)$, p is simple in picture $(T, m, n, B \setminus Q)$.

Let $p \in S$ and $Q \subseteq S \cap N_{12}^*(p)$. As p is simple, it matches a template $X \in \mathcal{T}^{(m,n)}$ in (T, m, n, B) by Theorem 2. If $X \neq \mathcal{T}_3^{(3,12)}$, then let $Q' \subseteq Q$ the set of black pixels coinciding with a position denoted by “.” in X , else let Q' be the set of black pixels coinciding with one of the positions denoted by x, y , and z . Obviously, if $Q' = Q$, then p will match X even in picture $(T, m, n, B \setminus Q)$.

Let us suppose that $X \in \mathcal{T}_1^{(m,n)}$. If $(m, n) = (3, 12)$, then $Q' = Q$, as if we removed any black 3-neighbor of p , then p would not match any template of $\mathcal{T}^{(m,n)}$, which, by Theorem 2, would contradict Condition 2.

Hence, in this case the property to be proved holds by the above observation on Q' . Let us examine the case $(m, n) = (12, 3)$ and let us introduce the set $R = \{x, y, z\} \cap B$. It is obvious that $Q \subseteq R$. If each element of R is simple, then it is easy to see by Condition 2 and Theorem 1 that this can only occur if p and the elements of R constitute an object that can be covered by a unit hexagon. This and Condition 3 implies that p is not removed by \mathcal{O} , i.e., $p \notin S$. If any element of R is simple, then p will obviously match X in picture $(T, 12, 3, B \setminus Q)$, hence by Theorem 2, it remains simple after the removal of Q .

Let us suppose that $X \in \mathcal{T}_2^{(m,n)} \cup \mathcal{T}_3^{(m,n)}$. If $(m, n) = (3, 12)$, then p would not match X (nor any other possible templates), hence, p would not be simple by Theorem 2. However, this would contradict Condition 2 (see Lemma 3), therefore

\mathcal{O} does not remove any pixel in $N_{12}^*(p) \setminus (N_3^*(p) \cup Q')$. Consequently, if \mathcal{O} does not delete any black 3-neighbor of p , then $Q = Q'$ must hold, thus the property to be proved holds by the above observation on Q' .

Let $q, r \in N_3^*(p) \cap B$. The following statements can be formulated on q and r , depending on the possible values of X :

- If $X \in \mathcal{T}_2^{(12,3)}$, then $q = r$, as p has only one black 3-neighbor.
- If $X \in \mathcal{T}_3^{(12,3)}$ or $X \in \mathcal{T}_2^{(3,12)}$, then $N_3^*(p) \cap B = \{q, r\}$. As both q and r have at least two black 3-neighbors, none of them matches any template of $\mathcal{T}_1^{(m,n)}$. By careful examination of the templates of $\mathcal{T}_2^{(m,n)}$ and $\mathcal{T}_3^{(m,n)}$, we can state that r may even not match these templates after the removal of q , and the same goes for q after the deletion of r , i.e., $\{q, r\}$ is not a simple set. Hence, by Condition 2, \mathcal{O} may only remove at most one black 3-neighbor of p .
- If $X \in \mathcal{T}_3^{(3,12)}$, then \mathcal{O} may not remove the black 3-neighbor of p being not 3-adjacent to x nor to z in X , or else p would not remain simple by Theorem [1](#), hence Condition 2 would fail (see Lemma [3](#)). Therefore, \mathcal{O} may remove at most two black 3-neighbors of p , namely the common black 3-neighbors of pixels p, x and p, z . Let these pixels be q and r , respectively. By applying Theorems 1 and 2, we can easily show that if \mathcal{O} removes q , then $x \in T \setminus B$, and if \mathcal{O} deletes r , then $z \in T \setminus B$ must be satisfied. Hence, if $\{q, r\}$ is a simple set, then, according to the above observations on $N_{12}^*(p) \setminus (N_3^*(p) \cup Q')$, $Q = \{q, r\}$ or $Q = \{q, r, y\}$ holds, furthermore, after the removal of $\{q, r\}$, p will match a template of $\mathcal{T}_1^{(3,12)}$. If $y \in Q$, then we can easily verify by Theorem [2](#) that the colors of q, r does not influence the simplicity of y , and in the latter template, the color of y does not influence the simplicity of p . From this we can conclude that p is simple even in picture $(T, 3, 12, B \setminus Q)$. If set $\{q, r\}$ is not simple, then \mathcal{O} can only remove at most one black 3-neighbor of p .

Hereafter, it is sufficient to examine the case when \mathcal{O} removes exactly one black 3-neighbor of p , as all the other cases are previously discussed. For the sake of simplicity, let $q \in N_3^*(p) \cap B$ be this removed pixel, i.e., let $q \in Q$. By Condition 2 and Lemma [3](#), p is simple after the removal of q , hence, p matches a template $Y \in \mathcal{T}_1^{(m,n)} \cup \mathcal{T}_2^{(m,n)}$ after q is deleted.

Let us examine the situation after the removal of q depending on the value of Y . We will prove that for any possible value of Y all the pixels of $Q \setminus \{q\}$ are simple in $(T, m, n, B \setminus \{q\})$, hereafter we can reduce the remaining part of the proof to one of the previously discussed situations.

- If $Y \in \mathcal{T}_1^{(12,3)}$, then obviously $Q \subseteq \{q, x, y, z\}$. It is easy to verify that the color of p does not influence the properties of pixels x, y, z formulated in Theorem [1](#), i.e., the pixels of $\{x, y, z\} \cap Q$ remain simple after the removal of q . From this point, the proof can be reduced to the case when $X \in \mathcal{T}_1^{(12,3)}$.

- If $Y \in \mathcal{T}_1^{(3,12)}$, then, by Condition 2, Lemma 3, and Theorem 1, each element of the sets $Q \cap N_{12}^*(q)$ and $Q \setminus N_{12}^*(q)$ remains simple after q is deleted. From this point, the proof can be reduced to the case when $X \in \mathcal{T}_1^{(3,12)}$.
- If $Y \in \mathcal{T}_2^{(12,3)}$, then, by Theorem 1, the elements of $Q \setminus N_{12}(q)$ remain simple after the removal of q . Obviously, the pixels of $Q \cap (N_{12}(q) \setminus N_3(q))$ fulfill Condition 2 of Theorem 1, furthermore, as they with q are elements of a unit hexagon that contains a non-deletable black pixel, and as the elements of a unit hexagon are pairwise 12-adjacent, pixels belonging to $Q \cap (N_{12}(q) \setminus N_3(q))$ also satisfy Condition 1 of Theorem 1, independently of the color of q . Consequently, by Theorem 1, the elements of $Q \cap (N_{12}(q) \setminus N_3(q))$ are simple in $(T, m, n, B \setminus \{q\})$, as well. In Y we can note that the white element on the position corresponding to the removed pixel q may have only one black 3-neighbor. Let $Q \cap N_3(q) = \{s\}$. Pixel s is simple even after q is removed, because of Condition 2 and Lemma 3. Hence, each pixel of $Q \setminus \{q\}$ remains simple after the removal of q . Hereafter, the proof can be reduced to the case when $X \in \mathcal{T}_2^{(12,3)}$ and the remaining black 3-neighbor of p is not deleted.
- If $Y \in \mathcal{T}_2^{(3,12)}$, then, by Condition 2 and Lemma 3, each element of $Q \cap N_{12}^*(q)$ remains simple after q is deleted, and the pixels of $Q \setminus N_{12}^*(q)$ were not even deletable in the initial picture by the previous observations. From this point, we can reduce the proof to the situation where $X \in \mathcal{T}_2^{(3,12)}$ and none of the remaining black 3-neighbors of p are deleted.

Herewith, we have examined all possible cases. □

We note that the above result is similar to Ronse’s sufficient conditions for topology preserving reduction on (8,4) and (4,8) pictures on the 2D orthogonal grid \mathbb{Z}^2 [12].

Figure 6 illustrates two examples for reductions on (12,3) pictures. The first one (see Fig. 6b) satisfies all the conditions of Theorem 3, therefore it is topology preserving. The second one (see Fig. 6c) violates Conditions 2 and 3 of Theorem 3, and it is topologically incorrect.

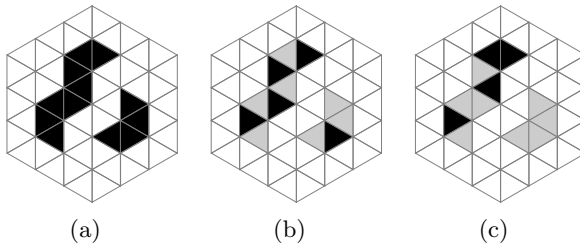


Fig. 6. The original picture (a) and the results produced by two reductions (b-c) on it. Deleted pixels are depicted in gray. The first reduction satisfies all conditions of Theorem 3, while the second one violates Conditions 2 and 3 of Theorem 3.

5 Validating Topology Preservation

Here we introduce a pair of reductions relying on the so-called subfield-based thinning strategy, which rests on the decomposition of the digital space into several subfields [4]. During an iteration step, the subfields are alternatively activated, and only pixels in the active subfield may be deleted. We propose the partitioning of the triangular grid T into two subfields, $SF(0)$ and $SF(1)$ (see Fig. 7).

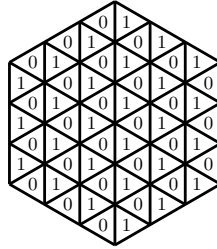


Fig. 7. Partition of T into two subfields. The pixels marked i are in $SF(i)$ ($i = 0, 1$).

By carefully examining the pattern in Fig. 7 we can observe the following property of this partitioning:

Proposition 1. *If $p \in SF(i)$ ($i = 0, 1$), then $N_3^*(p) \cap SF(i) = \emptyset$.*

Thinning algorithms must care about not only preserving the topology but also the shape of the original object. The latter requirement is usually fulfilled by adding some geometric constraints to the deleting conditions of the algorithms. For this purpose, here we give a definition of so-called end pixels.

Definition 1. *A black pixel p in a $(12, 3)$ picture is called an end pixel if there are at most two black pixels 12-adjacent to p , and at most one of them is 3-adjacent to p .*

We define our reductions on $(12, 3)$ pictures as follows.

Definition 2. *Let $R-SF-i$ be the reduction that deletes a black pixel $p \in SF(i)$ from a $(12, 3)$ picture if p is simple and not an end pixel ($i = 0, 1$).*

Now, using our sufficient conditions introduced in the previous section, we prove that the above reductions are topology preserving.

Theorem 4. *Both reductions $R-SF-0$ and $R-SF-1$ are topology-preserving.*

Proof. As $R-SF-i$ ($i = 0, 1$) deletes only simple pixels, Condition 1 of Theorem 3 is satisfied. Furthermore, Proposition 1 implies that if $R-SF-i$ deletes p , then p does not remove any $q \in N_3(p)$, which means that $R-SF-i$ fulfills Condition 2 of Theorem 3.

Let S be a set of the black pixels of an object contained in a unit hexagon, and let us denote by $|S|$ the number of elements in S . If there is a black pixel $p \in S$ such that $p \notin SF(i)$, then it is obvious that p will not be removed. Let us suppose that $S \subset SF(i)$. It can be readily seen that in this case $|S| \leq 3$. If $|S| = 1$, then the only object pixel in S is a non-simple pixel, which will not be deleted by $R-SF-i$. If $|S| = 2$ or $|S| = 3$, then the pixels of S are all end pixels, which are also retained by $R-SF-i$. Having examined all the possible cases for the content of S , we can conclude that Condition 3 of Theorem 3 also holds.

Hence, $R-SF-i$ ($i = 0, 1$) is topology-preserving by Theorem 3. □

Let us perform these two reductions successively in a (12,3) picture shown in Figure 8a. The effects of operators $R-SF-0$ and $R-SF-1$ can be observed in Figs. 8b-c.

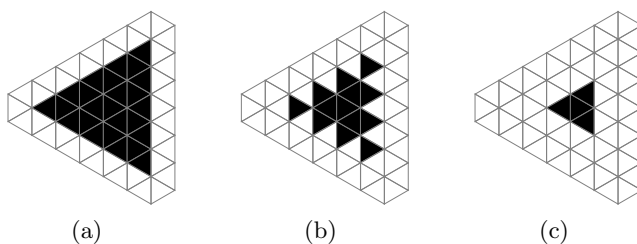


Fig. 8. A picture of a triangle (a) and the results produced by successively applying the reductions $R-SF-0$ (b) and $R-SF-1$ (c) in it

6 Conclusions

This paper has concerned itself with the topological properties of (12,3) and (3,12) pictures. We have given some characterizations of simple pixels in these types of pictures, and as the main novelty of our work, we have presented some sufficient conditions to ensure topology preservation for reductions on (12,3) pictures. As an illustration for the usefulness of these conditions, we have defined a pair of subfield-based reductions, and we have verified their topological correctness. As a future work we intend to give further conditions that are capable of constructing triangular thinning algorithms whose topological correctness is automatically guaranteed.

Acknowledgements. This research was supported by the European Union and the European Regional Development Fund under the grant agreements TÁMOP-4.2.1/B-09/1/KONV-2010-0005 and TÁMOP-4.2.2/B-10/1-2010-0012, and the grant CNK80370 of the National Office for Research and Technology (NKTH) & the Hungarian Scientific Research Fund (OTKA).

References

1. Deutsch, E.S.: On parallel operations on hexagonal arrays. *IEEE Transactions on Computers* C-19(10), 982–983 (1970)
2. Deutsch, E.S.: Thinning algorithms on rectangular, hexagonal, and triangular arrays. *Communications of the ACM* 15, 827–837 (1972)
3. Gaspar, F.J., Gracia, J.L., Lisboa, F.J., Rodrigo, C.: On geometric multigrid methods for triangular grids using three-coarsening strategy. *Applied Numerical Mathematics* 59(7), 1693–1708 (2009)
4. Hall, R.W.: Parallel connectivity-preserving thinning algorithms. In: Kong, T.Y., Rosenfeld, A. (eds.) *Topological Algorithms for Digital Image Processing*, pp. 145–179. Elsevier Science (1996)
5. Kardos, P., Palágyi, K.: On Topology Preservation for Hexagonal Parallel Thinning Algorithms. In: Aggarwal, J.K., Barneva, R.P., Brimkov, V.E., Koroutchev, K.N., Korutcheva, E.R. (eds.) *IWCIA 2011. LNCS*, vol. 6636, pp. 31–42. Springer, Heidelberg (2011)
6. Kong, T.Y.: On topology preservation in 2-d and 3-d thinning. *Int. Journal of Pattern Recognition and Artificial Intelligence* 9, 813–844 (1995)
7. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing* 48, 357–393 (1989)
8. Lam, L., Lee, S.-W., Suen, C.Y.: Thinning methodologies — A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence* 14, 869–885 (1992)
9. Ma, C.M.: On topology preservation in 3D thinning. *CVGIP: Image Understanding* 59, 328–339 (1994)
10. Marchand-Maillet, S., Sharaiha, Y.M.: *Binary Digital Image Processing – A Discrete Approach*. Academic Press (2000)
11. Nagy, B.: Characterization of digital circles in triangular grid. *Pattern Recognition Letters* 25(11), 1231–1242 (2004)
12. Ronse, C.: Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images. *Discrete Applied Mathematics* 21, 67–79 (1988)
13. Siddiqi, K., Pizer, S.: *Medial Representations: Mathematics, Algorithms and Applications*, 1st edn. Springer Publishing Company, Incorporated (2008)
14. Staunton, R.C.: An analysis of hexagonal thinning algorithms and skeletal shape representation. *Pattern Recognition* 29, 1131–1146 (1996)
15. Staunton, R.C.: One-pass parallel hexagonal thinning algorithm. In: *Proc. IEE 7th Int. Conf. Image Processing and its Applications*, pp. 841–845 (1999)
16. Suen, C.Y., Wang, P.S.P. (eds.): *Thinning methodologies for pattern recognition. Series in Machine Perception and Artificial Intelligence*, vol. 8. World Scientific (1994)
17. Kardos, P., Németh, G., Palágyi, K.: An Order-Independent Sequential Thinning Algorithm. In: Wiederhold, P., Barneva, R.P. (eds.) *IWCIA 2009. LNCS*, vol. 5852, pp. 162–175. Springer, Heidelberg (2009)

Cellular Topology on the Triangular Grid

Benedek Nagy

Department of Computer Science, Faculty of Informatics, University of Debrecen,
P.O. Box 12, 4010 Debrecen, Hungary
`nbenedek@inf.unideb.hu`

Abstract. In this paper we use the triangular grid and present a coordinate system that is appropriate to address elements (cells) of cell complexes. Coordinate triplets are used to address the triangle pixels of both orientations, the edges between them and the points at the corners of the triangles. To illustrate the utility of this system some topological algorithms, namely collapses and cuts are presented.

Keywords: Coordinate system, Triangular grid, Topology, Digital geometry.

1 Introduction

In digital geometry, i.e., in grids (tessellations of the plane) there are some phenomena which do not occur in the Euclidean plane. For instance, there are neighbor points (pixels), etc. there can be two 8-connected lines that cross on each other without sharing a pixel (as the two diagonals of a chessboard; this phenomena is connected to the Jordan curve theorem and its digital variations). There are several important examples for such a non-correspondence of concepts [9] and several algorithms and methods of digital image processing are based on these non-correspondences. In digital image processing and in computer graphics discrete space is used. In digital geometry spaces are digital, i.e., they consist of points that can be addressed by integer coordinate values. The square and cubic grids are well-known and frequently used in applications, since they use a Cartesian coordinate system. To use other grids, one needs a good coordinationization method. The points of the hexagonal grid can be addressed by two integers [8]. There is a more elegant solution using three coordinate values with zero sum reflecting the symmetry of the grid [6]. Similarly the triangular grid can be described with three values [14–17] (see Fig. 1).

Topology is an important part of geometry and so, of image processing theory [5, 10]. One of the main concepts is the cell complex. For digital images geometry of locally finite spaces is given [13]. Two and three dimensional images are usually stored in two and three dimensional arrays. Each element of the array contain a color, a density, etc. value. For topological calculations we need a modification: a two dimensional image, as a cell complex, contain not only two-dimensional pixels, but one dimensional edges and zero dimensional points, as well. Usually we do not care about these lower dimensional parts since only pixels are shown

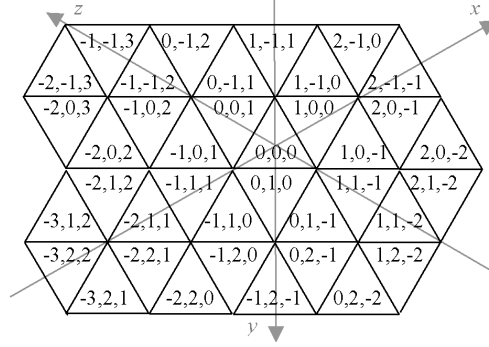


Fig. 1. A symmetric coordinate frame for triangular pixels

in a screen, only they are printed out etc. However for image processing feature these parts play also importance. In this paper we extend the coordinate system addressing not only the triangle pixels, but also the edges and the points on their boundaries with coordinate triplets analogously to the method given by Kovalevsky in [12, 13]. In this way the lower dimensional parts of the picture can also be shown in the screen and corresponding image processing algorithms can also be described more elegantly (see e.g. [11]).

We note here that there is another system for triangles that uses three coordinate values: the barycentric coordinates. They were discovered by Möbius (see, e.g., [4]). They can effectively be used to address points inside the triangle with triplets. Actually, the values can be seen as masses located in the three corners of the triangle resulting the center of mass at the addressed point. Our aim in this paper is not to address the points of a triangle but to address the various objects of the (digital) triangular grid.

The structure of the paper is as follows. In the next section we recall some earlier results concentrating on the square grid. In Section 3 we present the new, topological coordinate system for the triangular grid, further cell complexes and collapses and cuts are described using the new coordinate system to present its usability. Finally some conclusions and future thoughts close the paper.

2 Preliminary Results: The Square Grid

In this section we recall and adapt some definitions from [12] to our case. We also present some details about the case of the square grid in Subsection 2.1.

Definition 1. An **abstract cell complex** $C = (E, B, d)$ is a set E of abstract elements (cells) provided with an antisymmetric, irreflexive, and transitive binary relation $B \subset E \times E$ called the bounding relation, and with a dimension function $d : E \rightarrow \{0, 1, 2\}$ such that $d(e') < \dim(e'')$ for all pairs $(e', e'') \in B$.

The maximum dimension d of the cells of an abstract cell complex is called its dimension. We consider complexes of dimensions 2 in the triangular grid. Their cells with dimension 0 (0-cells) are called points (or nodes), cells of dimension 1 (1-cells) are called edges, cells of dimension 2 (2-cells) are called pixels (or triangles). If $(e', e'') \in B$, then it is usual to write $e' < e''$ or to say that the cell e' bounds (or it is on the boundary of) the cell e'' . Two cells e' and e'' of an abstract cell complex C are called incident to each other in C if and only if either $e' = e''$, or e' bounds e'' , or e'' bounds e' .

It is very crucial that in abstract cell complexes no cell is a subset of another cell, as it is the case in simplicial and Euclidean (or geometric) complexes [5, 7]. Exactly this property of abstract cell complexes makes it possible to define a topology on the set of abstract cells independently from any Hausdorff space. The topology of abstract cell complexes with some applications to computer imagery has been described in [13].

Definition 2. A *subcomplex* $S = (E', B', d')$ of a given abstract cell complex $C = (E, B, d)$ is a complex such that $E' \subset E$ and the relation B' is a restriction of B to E' , i.e., it is the intersection of B with $E' \times E'$. The dimension d' is equal to d for all cells of E' .

Since a subcomplex is uniquely defined by the subset E' it is possible to apply set operations as union, intersection and complement to abstract cell complexes. We could say the term subset in the meaning subcomplex. The connectivity in abstract cell complexes is the transitive hull of the incidence relation. It can be shown that the connectivity thus defined corresponds to classical connectivity.

2.1 The Square Grid

Here we recall the topological raster and the extended (uniform) coordinate system used by Kovalevsky to address two-, one-, and zero-dimensional cells, i.e., square pixels, edges and points (corners) on the square grid (Fig. 2). The pixels are addressed by two odd values, the points of the grid are addressed by two even values, the horizontal edges of the grid is addressed by a vector with even first and odd second component, while vertical edges of the grid has vector with odd first and even second component. In this way each cell has different coordinates, and the number of odd values gives the dimension of the cell.

This topological (or combinatorial) coordinate system is very helpful to use various dimensional cells in the same framework in topological image processing algorithms.

In the next section we will show similar uniform coordinate system for the triangular grid.

3 The Extended Coordinate System for the Triangular Grid

In this section we present an extended coordinate system by modifying the system defined and used in [14–17] (Fig. 1) to include addressing the zero and one

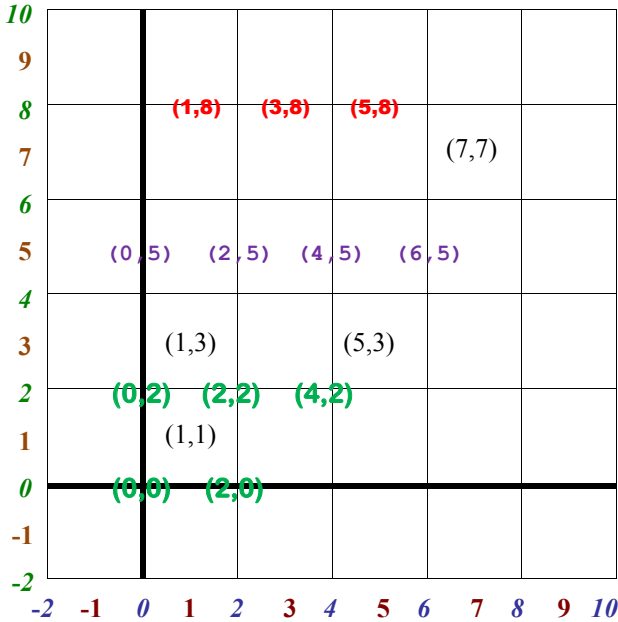


Fig. 2. Cartesian cells and some of their coordinate values

dimensional cells: to use not only the triangle pixels, but the edges between two neighbor triangles and their endpoints, i.e., the corners of the triangles (the nodes of the grid) we need to address them also. Since we want to use only integers, first we modify the original coordinate system as follows. For any triangle that had coordinate triplet (x, y, z) we use the triplet $(2x, 2y, 2z)$ in the new system.

We note here that the square grid is selfdual: the (neighbor) structure of the square pixels is the same as the (neighborhood) structure of the zero-dimensional points of the grid. Therefore we have a choice that points or squares should have vector with even coordinate values. We presented Kovalevsky's choice in the previous section. (Actually everything goes in an analogous way with the other choice.) However the graph theoretical dual of the triangular grid is the hexagonal grid: while the (neighborhood) structure of the pixels form the triangular grid, i.e., we use the triangles; the zero-dimensional points (nodes) of the grid forms a hexagonal structure. For this reason we use even coordinate values for the triangle pixels.

There are two types of triangles: shape \triangle and shape ∇ . In the original coordinate system they are addressed by zero sum coordinate triplets and by one-sum coordinate triplets, respectively. According to this fact the names even- and odd- triangles are used. By the structure of the grid a triangle pixel has three

neighbor pixels, and all of them have the opposite shape than the considered triangle. In our new coordinate system the even triangles (\triangle) have zero-sum coordinate values, while odd triangles (∇) have two-sum coordinate triplets. Then the neighbor triangles (triangles sharing an edge) have two coordinate values in which they do not differ, and the difference in the third coordinate value is ± 2 . Therefore the edge between them can be addressed by two even coordinates (the same as for the two triangles) and an odd value that is the average of the values of the two triangles. Actually, having two neighbor triangles with coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) , the edge between them is addressed by the triplet $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}, \frac{z_1+z_2}{2})$ (a triplet with two even and an odd value). The sum of the coordinate values of every edge is one. The edges of the grid lies on gridlines that are orthogonal to one of the coordinate axes. The odd coordinate is the first element of the vector for edges orthogonal to axis x . Edges orthogonal to axis y has odd value at their second coordinate. Finally, edges orthogonal to axis z are described by vectors having odd third value. Accordingly a node (zero-dimensional cell, point) is described by a triplet with odd values. The sum of these coordinate values is one for each point. Actually, these points form a hexagonal-grid structure, and therefore they can be described by triplets with a fixed coordinate sum (based on [6, 14, 15, 18, 19]). See Fig. 3 also, where a small segment of the grid is shown with the coordinate values. Consequently, the

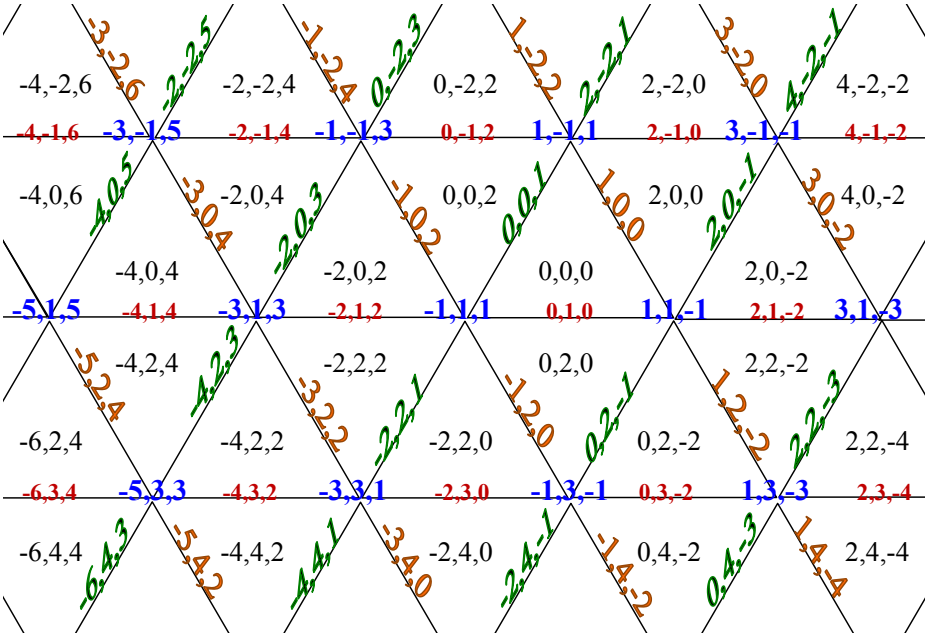


Fig. 3. The topological coordinate system for a segment of the triangular grid

dimension and also the orientation (it can be defined for the triangles and the edges) of any cell can be described from its topological coordinate values.

Considering the topological neighborhood structure of the grid, we have the following properties: stepping from a triangle to one of its sides (edges of the grid) two coordinate values do not change, while the third one changes by ± 1 with respect to the direction of the step (and resulting a one-sum triplet). When we step from an edge to one of its endpoints (corners of the triangles, nodes of the grid) a coordinate value does not change (our step is orthogonal to one of the coordinate axes), the two other values are changing by ± 1 depending on the direction of the step. When we step from a triangle to one its corner, all the three values are changing by ± 1 such that the sum of the values become one. We will also describe these facts in a formal way. In our description the parity of the coordinate values and the parity of the coordinate-sum together can be used to show the dimension of the cell. So let n be the number of even coordinate values of a cell plus one if the coordinate sum is also even (else, i.e., when the coordinate sum is odd, we do not add this one). Then the dimension of the cell is $\frac{n}{2}$.

Now we present some technical definitions: By fixing a coordinate value one can obtain *lines* and *lanes*:

Definition 3. *Let us fix a coordinate value. If it is even, then all the cells having this fixed coordinate value form a lane.*

If the fixed coordinate value is odd, then all the cells having this coordinate value form a line.

A lane consists of triangles and edges alternately, moreover in the following order: odd triangle, edge, even triangle, edge, etc., where the corresponding edges are between the two consecutive triangles. Any line consists of edges and points alternately, where the point between two edges is their common endpoint.

3.1 Cell Complexes on the Triangular Grid

In this subsection we present some details about cell complexes using the new coordinate system.

Let us see how the bound relation $(e', e'') \in B$ (i.e., $e' < e''$) and thus the incidence relation can be described by our coordinate system. First the boundary of an edge is described, therefore let e'' be an edge (one-dimensional cell) and let (x_1, y_1, z_1) be its coordinate triplet. Then e'' bounds e' if e' is a point and its coordinate triplet differ from (x_1, y_1, z_1) by $+1$, -1 and $+0$, respectively. In details, there are three possibilities up to direction (orientation) of the edge:

- edges orthogonal to axis x (i.e., x_1 is odd, y_1 and z_1 are even): the points e' with $(x_1, y_1 + 1, z_1 - 1)$ and $(x_1, y_1 - 1, z_1 + 1)$ are the ones for which $e' < e''$ is fulfilled;

- edges orthogonal to axis y (i.e., y_1 is odd, x_1 and z_1 are even): the points e' with $(x_1 + 1, y_1, z_1 - 1)$ and $(x_1 - 1, y_1, z_1 + 1)$ are the ones for which $e' < e''$ is fulfilled;
- edges orthogonal to axis z (i.e., z_1 is odd, x_1 and y_1 are even): the points e' with $(x_1 + 1, y_1 - 1, z_1)$ and $(x_1 - 1, y_1 + 1, z_1)$ are the ones for which $e' < e''$ is fulfilled.

Let us see the cells that bound a triangle. Let e'' be a triangle with coordinate values (x_1, y_1, z_1) . Then all the cells (but e'') bound e'' with coordinate values from $X \times Y \times Z$ where $X = \{x_1 - 1, x_1, x_1 + 1\}$, $Y = \{y_1 - 1, y_1, y_1 + 1\}$ and $Z = \{z_1 - 1, z_1, z_1 + 1\}$. Actually, depending on the orientation (parity) of the triangle the coordinate values of its bounding edges can be computed as follows: change any of the values by ± 1 to obtain a coordinate triplet with one-sum. Changing all the three coordinate values by ± 1 to obtain one-sum triplets the coordinate triplets of the bounding points are obtained. For more precisely, there are the following cases:

Let e'' be a triangle with coordinate values (x_1, y_1, z_1) .

- Let e'' be an even triangle, i.e., $x_1 + y_1 + z_1 = 0$, then
 - the edges e'' with $(x_1 + 1, y_1, z_1)$, $(x_1, y_1 + 1, z_1)$ and $(x_1, y_1, z_1 + 1)$ are the ones for which $e'' < e'$ is fulfilled; (they are orthogonal to the axes x , y and z , respectively.)
 - the points e'' with $(x_1 + 1, y_1 + 1, z_1 - 1)$, $(x_1 + 1, y_1 - 1, z_1 + 1)$ and $(x_1 - 1, y_1 + 1, z_1 + 1)$ are the ones for which $e'' < e'$ is fulfilled;
- Let e'' be an odd triangle ($x_1 + y_1 + z_1 = 2$), then
 - the edges e'' with $(x_1 - 1, y_1, z_1)$, $(x_1, y_1 - 1, z_1)$ and $(x_1, y_1, z_1 - 1)$ are the ones for which $e'' < e'$ is fulfilled; (they are orthogonal to the axes x , y and z , respectively.)
 - the points e'' with $(x_1 + 1, y_1 - 1, z_1 - 1)$, $(x_1 - 1, y_1 + 1, z_1 - 1)$ and $(x_1 - 1, y_1 - 1, z_1 + 1)$ are the ones for which $e'' < e'$ is fulfilled.

In the next section we present algorithm for collapses and cuts ([2]) based on our new coordinate system.

4 Collapses and Cuts

In this section we present an algorithm for collapses on the triangular grid. Let $C = (E, B, d)$ be an abstract cell complex on the triangular grid. Let e be an edge (1-cell) in C . If there exists a unique triangle (2-cell) e' in C such that $e < e'$, then $C_1 = C \setminus \{e, e'\}$ is an *elementary 2-collapse* of C . Let e be a point (0-cell) in C . If there exists a unique edge (1-cell) e' in C such that $e < e'$, then $C_1 = C \setminus \{e, e'\}$ is an *elementary 1-collapse* of C . If there is a d -collapse sequence

($d = 1, 2$) $C = C_0, C_1, \dots, C_k = C'$ (for $k \geq 0$) such that C_i is a d -collapse of C_{i-1} for every $1 \leq i \leq k$, then C' is a d -collapse of C . If C' is a d -collapse of C and there is no $C'' \neq C'$ such that C'' is a d -collapse of C' , then C' is an *ultimate d -collapse* of C . Below algorithms are shown that compute a collapses and an ultimate collapses of the input abstract cell complex C .

Algorithm 1. For an elementary 2-collapse

Input: the cell complex C .

Find an edge (x, y, z) in C (with two even an one odd values)

such that exactly one of the elements of the set

$S = \{(x+1, y, z), (x-1, y, z), (x, y+1, z), (x, y-1, z), (x, y, z+1), (x, y, z-1)\}$ is in C .

if there is no such edge,

then C is an ultimate 2-collapse, STOP.

else

let (x', y', z') be the triangle in $S \cap C$;

let $C' = C \setminus \{(x, y, z), (x', y', z')\}$;

the output is C' as a 2-collapse of C ;

endif.

The next algorithm obtains an elementary 1-collapse of the input C . It is usually required that the input C is an ultimate 2-collapse (of itself).

Algorithm 2. For an elementary 1-collapse

Input: the cell complex C .

Find a point (x, y, z) in C (with x, y, z all odd values)

such that exactly one of the elements of the set $S = \{(x+1, y-1, z),$

$(x-1, y+1, z), (x, y+1, z-1), (x, y-1, z+1), (x-1, y, z+1), (x+1, y, z-1)\}$ is in C .

if there is no such point,

then C is an ultimate 1-collapse, STOP.

else

let (x', y', z') be the edge in $S \cap C$;

let $C' = C \setminus \{(x, y, z), (x', y', z')\}$;

the output is C' as a 1-collapse of C ;

endif.

By iterative use of the above steps one can obtain an ultimate d -collapses.

Algorithm 3. For ultimate d -collapse

Input: the cell complex C and the parameter $d \in \{1, 2\}$.**if** $d = 2$ **then** **while** (there is an edge (x, y, z) in C)

such that exactly one of the elements of the set

 $S = \{(x+1, y, z), (x-1, y, z), (x, y+1, z), (x, y-1, z), (x, y, z+1), (x, y, z-1)\}$
 is in C) **let** (x', y', z') be the unique triangle in $S \cap C$; **let** $C = C \setminus \{(x, y, z), (x', y', z')\}$; **endwhile** the output C is an ultimate 2-collapse, STOP.**else** **while** (there is a point (x, y, z) in C)

such that exactly one of the elements of the set

 $S = \{(x+1, y-1, z), (x-1, y+1, z), (x, y+1, z-1),$
 $(x, y-1, z+1), (x-1, y, z+1), (x+1, y, z-1)\}$
 is in C) **let** (x', y', z') be the unique edge in $S \cap C$; **let** $C = C \setminus \{(x, y, z), (x', y', z')\}$; **endwhile** the output C is an ultimate 1-collapse, STOP.**endif.**

Note that computing collapses need only local tests and changes.

Segmentation is an image processing task of finding objects of interest. Usually the result of such a process is a set of incident cells which constitutes the separation between regions. A separation which cannot be reduced without connecting some regions is a cut.

Actually a cut of an abstract cell complex C can be computed from any of its ultimate 1-collapses by eliminating cells that are not on the boundary of any regions. There could be a cell e' in an ultimate 1-collapse C' that all the cells e'' for which $e' < e''$ are in the same region, i.e., any two of them can be connected by a sequence of incident cells such that none of the elements of the sequence are elements of C' . To eliminate such cells and obtain a cut we need a global computation: this can be done by a bread-first-like search resulting a global labelling. A similar linear time technique is shown in ([2]).

5 Conclusions and Future Work

Non-traditional grids are used more frequently in image processing and computer graphics. Some results on the hexagonal and triangular grids can be found in [1, 20, 21]. In this paper the topological coordinate system is introduced for the triangular grid. We have defined a coordinate system that is appropriate to

describe abstract cell complexes on the triangular grid. We have detailed some important properties of the description and give also algorithm for collapses and cuts using our coordinate frame. We use coordinate triplets reflecting the symmetry of the grid. In the proposed coordinate system the natural directions of the grid (parallel and orthogonal to the edges/lines of the grid) have the same (symmetric) role. One may think that two values (i.e., coordinate pairs) are enough to address various cells, since the triangular grid is a two dimensional grid. Unfortunately, the number and the directions of the neighbor cells, i.e., the structure of the grid, do not allow to use a simple coordinate system with two coordinates. With coordinate triplets, however, the elements of the cell complexes can be addressed in a uniform way as we detailed. The main advantages of the topological description is that it contains more information about the image than the usual description: storing not only the pixels of the two dimensional image, but storing also the lower dimensional segments, the edges and the points separating the pixels. Therefore it is reasonable to use the topological system for research purposes. The best way to use the topological coordinates consists in the following by the advice of Kovalevsky ([12]). When thinking about the problem to be solved one should use the topological coordinates. Also a program which must solve a topological or geometrical problem in which notions like the connectivity, boundary, incidence etc. are involved should work with topological coordinates. Only the storage of values assigned to the pixels, should be performed to save memory. This is possible since the cells of lower dimension do not carry values like color, gray value or density. They only carry indices of subsets to which they belong, i.e., foreground or background.

In this paper we made the first step to use cellular topology on the triangular grid, further extension, such as, description of open and smallest open neighborhood, closure, frontier and open frontier will come. Other algorithms (e.g., boundary tracing, segmentation, thinnings, watersheds) can also be investigated using this framework ([3]).

Acknowledgements. The author thank the anonymous referees for their useful remarks and suggestions.

The research is supported by by the TÁMOP 4.2.1/B-09/1/KONV-2010-0007 project. The project is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund.

References

1. Brimkov, V.E., Barneva, R.: “Honeycomb” vs square and cubic models. *Electronic Notes in Theoretical Computer Science* 46 (2001)
2. Cousty, J., Bertrand, G., Couprie, M., Najman, L.: Collapses and Watersheds in Pseudomanifolds. In: Wiederhold, P., Barneva, R.P. (eds.) *IWCIA 2009*. LNCS, vol. 5852, pp. 397–410. Springer, Heidelberg (2009)
3. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest path forests, and topological watersheds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(5), 925–939 (2010)

4. Coxeter, H.S.M.: Introduction to Geometry, 2nd edn. Wiley, New York (1969)
5. Hatcher, A.: Algebraic Topology. Cambridge University Press (2002)
6. Her, I.: Geometric transformations on the hexagonal grid. IEEE Transactions on Image Processing 4(9), 1213–1222 (1995)
7. Ito, K. (ed.): Encyclopedic Dictionary of Mathematics, vol. 2. MIT Press (1993, 2000) (original Japanese edition, 1954)
8. Luczak, E., Rosenfeld, A.: Distance on a hexagonal grid. Transactions on Computers C-25(5), 532–533 (1976)
9. Klette, R., Rosenfeld, A.: Digital geometry. Geometric methods for digital picture analysis. Morgan Kaufmann Publishers, San Francisco; Elsevier Science B.V., Amsterdam (2004)
10. Kong, T., Rosenfeld, A.: Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing 48(3), 357–393 (1989)
11. Kovalevsky, V.: Finite topology as applied to image analysis. Computer Vision, Graphics and Image Processing 45, 141–161 (1989)
12. Kovalevsky, V.: Algorithms in Digital Geometry Based on Cellular Topology. In: Klette, R., Žunić, J. (eds.) IWICIA 2004. LNCS, vol. 3322, pp. 366–393. Springer, Heidelberg (2004)
13. Kovalevsky, V.A.: Geometry of Locally Finite Spaces (Computer Agreeable Topology and Algorithms for Computer Imagery). Editing House Dr. Bärbel Kovalevski, Berlin (2008)
14. Nagy, B.: A family of triangular grids in digital geometry. In: ISPA 2003, 3rd International Symposium on Image and Signal Processing and Analysis, Rome, Italy, pp. 101–106 (2003)
15. Nagy, B.: Generalized triangular grids in digital geometry. Acta Mathematica Academiae Paedagogicae Nyíregyháziensis 20, 63–78 (2004)
16. Nagy, B.: A symmetric coordinate frame for hexagonal networks. In: Theoretical Computer Science - Information Society 2004, Ljubljana, Slovenia, pp. 193–196 (2004)
17. Nagy, B.: Characterization of digital circles in triangular grid. Pattern Recognition Letters 25(11), 1231–1242 (2004)
18. Nagy, B., Strand, R.: A Connection between \mathbb{Z}^n and Generalized Triangular Grids. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part II. LNCS, vol. 5359, pp. 1157–1166. Springer, Heidelberg (2008)
19. Nagy, B., Strand, R.: Non-traditional grids embedded in \mathbb{Z}^n . International Journal of Shape Modeling - IJSM 14(2), 209–228 (2008)
20. Yong-Kui, L.: The generation of straight lines on hexagonal grids. Comput. Graph. Forum 12(1), 21–25 (1993)
21. Wuthrich, C.A., Stucki, P.: An algorithm comparison between square- and hexagonal-based grids. Graph. Model Im. Proc. 53(4), 324–339 (1991)

A P System Model for Contextual Array Languages

K.G. Subramanian¹, Ibrahim Venkat¹, and Petra Wiederhold²

¹ School of Computer Sciences, Universiti Sains Malaysia,
11800 Penang, Malaysia

² Department of Automatic Control,
Centro de Investigacion y de Estudios Avanzados (CINVESTAV-IPN),
Av. I.P.N. 2508, Col. San Pedro Zacatenco, Mexico 07000 D.F., Mexico

Abstract. A P system model, called Contextual array P system, that makes use of array objects and contextual array rules, is introduced and its generative power in the description of picture arrays is examined, by comparing it with certain other array generating devices.

Keywords: Two-dimensional arrays, Array grammars, Contextual Array rules, P system.

1 Introduction

Motivated by different problems in the framework of image analysis and picture processing, the classical grammar models of formal string languages [20,21] have been extended by many researchers and several array grammar models have been proposed (see for example [10,17,19,24,26] and references therein) for generation of picture arrays in the two-dimensional plane. Most of these array grammars extend the rewriting feature in Chomsky's string grammars and make use of string rewriting rules or array rewriting rules. On the other hand the contextual array grammar introduced and investigated by Freund et al [9] is an interesting array counterpart of string contextual grammars introduced by Marcus [14] and intensively investigated subsequently (see for example [6,15]). In the string case, contextual grammars are motivated from certain fundamental linguistic phenomenon [15]. There is a basic feature in the generation of strings of symbols in contextual grammars. Unlike the Chomsky grammars [20] or Lindenmayer systems [20], in contextual grammars rewriting of symbols is not done, modifying the symbols. Instead, strings of symbols are adjoined to the current string and the symbols once introduced remain in the finally generated string .

In the area of membrane computing, P systems which were introduced by Paun [16] have turned out to be a rich framework for dealing with several types of computing related problems. Among different kinds of P systems, rewriting P systems, in which objects are given as finite strings over an alphabet and the evolution rules are given as context-free rewriting rules, have been investigated extensively [1,8,16,28]. Contextual way of handling string-objects in P

systems has been considered in [13] and contextual P systems are found to be more powerful than string contextual grammars. Ceterchi et al [2] introduced and investigated array P systems of the isometric variety, extending the string-rewriting P systems to arrays using context-free type of rules. Subsequently, several P system models for array generation, both isometric and non-isometric variety, have been considered in the literature (see for example [23]). It is natural to introduce and examine the power of using contextual type of rules in array P systems. Based on the contextual style of array generation considered in [9], we introduce a P system model with array objects and array contextual rules and bring out its generative power by comparing it with other array generative devices. In particular, we note that the P system model considered here and called as Contextual Array P System, properly includes a class of contextual array languages considered in [9], thus bringing out the power of handling array objects in P systems in the contextual way.

2 Preliminaries

For notions related to formal language theory we refer to [20,21] and for array grammars and two-dimensional languages we refer to [9,10]. We briefly recall, as found in [9], the mathematical formulation of needed notions on arrays and array contextual grammars.

We consider the two-dimensional digital plane \mathbb{Z}^2 , where \mathbb{Z} is the set of integers, together with a graph theoretical connectivity concept. Any point of \mathbb{Z}^2 is called a pixel, and each pixel p is identified with the unit closed square whose center is p . So, it is common to name a pixel also a square or a cell. Two pixels $p = (p_1, p_2), q = (q_1, q_2) \in \mathbb{Z}^2$ are called neighbors (more precisely, 8-neighbors [18]) if $\|p - q\|_{max} = \max\{|p_1 - q_1|, |p_2 - q_2|\} = 1$. The neighborhood relation defines an undirected graph on \mathbb{Z}^2 . A subset $M \subseteq \mathbb{Z}^2$ is called connected if for any two points $p, q \in M$, there is a sequence (a path in the graph) $p = m_1, m_2, \dots, m_k = q$ of points of M where any two consecutive points are neighbors. Interpreting the digital plane as set of squares, two squares p, q are neighbors if they intersect. The connectivity used here coincides with the well-known 8-connectivity on the two-dimensional digital plane \mathbb{Z}^2 [18], and with the 1-connectivity on \mathbb{Z}^2 considered in [9].

Now let V be a finite alphabet, and $\#$ a symbol not belonging to V . A two-dimensional array is a set of pixels labelled by the symbols of V or the so-called *blank symbol* $\#$, where $\#$ indicates that the pixel is empty or unlabelled. Formally, a (*two-dimensional*) *array* is a function $\alpha : \mathbb{Z}^2 \rightarrow V \cup \{\#\}$ with finite and connected support $supp(\alpha)$, given by $supp(\alpha) = \{v \in \mathbb{Z}^2 | \alpha(v) \neq \#\}$. Although an array could be given as a function defined on a proper subset of \mathbb{Z}^2 , it can be extended to an array defined on the whole plane \mathbb{Z}^2 , assigning the blank symbol $\#$ to any pixel non-labelled by some symbol of V . So, for providing an array, it is sufficient to specify some finite set $supp(\alpha)$ and to give the label (element of V) which the function α assigns to each pixel $p \in supp(\alpha)$. The restriction of an array to its support, which is a connected finite set of

pixels labelled by elements of V , is named a *pattern*. Due to the suppositions of connectivity and finiteness of its support, an array is named (*two-dimensional connected finite array* in [9].

Note that the supposition of finiteness of the support implies that $supp(\alpha)$ is always completely surrounded by pixels having the blank symbol.

We denote by V^{+2} the set of all non-empty, (connected finite) arrays over V , where the empty array is supposed as the function which assigns the blank symbol to all pixels of \mathbb{Z}^2 . An *array language* is a subset of V^{+2} . For example, Fig. 1 shows an array (which is a pattern) \mathcal{H} describing the letter H that has two of its pixels labelled by c with the other pixels labelled by a . Supposing that the pixel having label c in the left vertical arm of the letter H has coordinates $(0,0) \in \mathbb{Z}^2$, the array in Fig. 1 can be described in a formal manner by giving the label $\alpha(p) \in V = \{a, c\}$ for each pixel p belonging to the pattern as follows:

$$\begin{aligned} \mathcal{H} = \{ & ((0, 0), c), ((1, 0), a), ((2, 0), a), ((3, 0), a), ((4, 0), a), ((5, 0), a), ((6, 0), c), \\ & ((0, 1), a), ((0, 2), a), ((0, 3), a), ((0, -1), a), ((0, -2), a), ((0, -3), a), \\ & ((6, 1), a), ((6, 2), a), ((6, 3), a), ((6, -1), a), ((6, -2), a), ((6, -3), a) \} . \end{aligned}$$

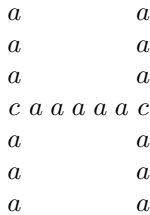


Fig. 1. An array (pattern) representing the letter H

For any $v \in \mathbb{Z}^2$, the translation $\tau_v : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$, given by $\tau_v(w) = w + v$ for all $v \in \mathbb{Z}^2$, provides a bijection between $supp(\alpha)$ and $\tau(supp(\alpha))$ for any array α . Defining $\alpha'(\tau_v(p)) = \alpha(p)$ for any $p \in supp(\alpha)$, we obtain an array α' which is the translation of the array α . On the set of all arrays, the binary relation defined for any two arrays by the fact that one array is a translation of the other one, clearly is an equivalence relation. Arrays can therefore be regarded as equivalence classes of arrays with respect to translations on \mathbb{Z}^2 . Hence, only relative positions of the symbols different from the blank symbol are essential for describing an array. In order to keep the description simple, we refer to a member of such an equivalence class as the array itself. We will use the usual method to denote an array by a figure indicating only the non-blank labels of the pixels belonging to its support, but without mentioning the coordinates of the pixels themselves. For example, the array in Fig. 1 is shown in this manner.

Given two arrays α, β , array β is called a *sub-array* of α if there exists a vector $v \in \mathbb{Z}^2$ such that for the translation τ_v it follows that $\tau_v(supp(\beta)) \subseteq supp(\alpha)$. In other words, all labelled pixels of β coincide with the corresponding labelled pixels of α when β is placed on α after a suitable translation of the pattern $supp(\beta)$.

We now recall the definition of a contextual array grammar [9], restricting ourselves to the two-dimensional case.

Definition 1. A contextual array grammar (CAG) is a construct $G = (V, \#, P, A)$ where V is an alphabet not containing the blank symbol $\#$, A is a finite set of two-dimensional arrays in V^{+2} , named axioms, and P is a finite set of rules of the form (α, β) , called array contextual rules, where

- (i) α is a function defined on $U_\alpha \subset \mathbb{Z}^2$ with values in $V \cup \{\#\}$ such that $\text{supp}(\alpha) \neq \emptyset$;
- (ii) β is a function defined on $U_\beta \subset \mathbb{Z}^2$ with values in V ;
- (iii) $U_\alpha \cap U_\beta = \emptyset$, U_α, U_β are finite and at least U_α is non-empty.

(U_α, α) is called the selector and (U_β, β) the context of the production (α, β) ; U_α is called the selector area, and U_β is the context area.

For arrays $\mathcal{C}_1, \mathcal{C}_2 \in V^{+2}$, intuitively, if in \mathcal{C}_1 we find a sub-array that corresponds to the selector (U_α, α) , and if the places corresponding to (U_β, β) are labelled only by the blank symbol, then we can add the context (U_β, β) , thus obtaining the derivation \mathcal{C}_2 . Formally, \mathcal{C}_2 is called directly derivable from \mathcal{C}_1 by the contextual array production $p = (\alpha, \beta) \in P$ (we write $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$), if there exists a vector $v \in \mathbb{Z}^2$ such that, denoting again by τ_v the translation by v , the following is true:

- $\mathcal{C}_1(w) = \mathcal{C}_2(w) = \alpha(\tau_{-v}(w))$ for all $w \in \tau_v(U_\alpha)$,
- $\mathcal{C}_1(w) = \#$ for all $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_2(w) = \beta(\tau_{-v}(w))$ for all $w \in \tau_v(U_\beta)$,
- $\mathcal{C}_1(w) = \mathcal{C}_2(w)$ for all $w \in \mathbb{Z}^2 \setminus \tau_v(U_\alpha \cup U_\beta)$.

If there exists a contextual array production $p \in P$ such that $\mathcal{C}_1 \Longrightarrow_p \mathcal{C}_2$, then \mathcal{C}_2 is called derivable from \mathcal{C}_1 and we write $\mathcal{C}_1 \Longrightarrow_G \mathcal{C}_2$. By \Longrightarrow_G^* we denote the reflexive transitive closure of \Longrightarrow_G and by \Longrightarrow_G^t we denote the relation which, for arbitrary arrays $\mathcal{A}, \mathcal{B} \in V^{+2}$ is defined by $\mathcal{A} \Longrightarrow_G^t \mathcal{B}$ if and only if $\mathcal{A} \Longrightarrow_G^* \mathcal{B}$ and there is no $\mathcal{C} \in V^{+2}$ such that $\mathcal{B} \Longrightarrow_G \mathcal{C}$.

For a given CAG grammar G , the relation \Longrightarrow_G^t corresponds to collecting only the arrays produced by blocked derivations, namely, derivations which cannot be continued. This derivation mode is known as the maximal mode or t -mode (“termination-mode”) [4]. In contrast, the collection of all possible derivations is commonly named the $*$ -mode of derivation. In consequence, due to these two modes, the following two kinds of array languages can be associated with a CAG grammar G :

$$L_*(G) = \{\mathcal{B} \in V^{+2} \mid \mathcal{A} \Longrightarrow_G^* \mathcal{B} \text{ for some } \mathcal{A} \in A\},$$

$$L_t(G) = \{\mathcal{B} \in V^{+2} \mid \mathcal{A} \Longrightarrow_G^t \mathcal{B} \text{ for some } \mathcal{A} \in A\}.$$

In general, the family of two-dimensional array languages generated by contextual array grammars of the form $G = (V, \#, P, A)$ with G in the f -mode with $f \in \{*, t\}$ will be denoted by $L(\text{cont}, f)$. We illustrate with an example of a contextual array grammar working in t -mode of derivation, the generation of solid rectangles of a 's.

Example 1. Consider the contextual array grammar $G = (\{a\}, \#, P, A)$ with A containing the axiom array $[{\{(0, 0), a\}, \{(0, 1), a\}, \{(1, 0), a\}}]$, which is given in the figure $\begin{matrix} a \\ a & a \end{matrix}$, and P consisting of the rules p_i of the form (α_i, β_i) , $i = 1, 2, \dots, 8$. Only two of the rules are listed and the rest of the rules are given only in pictorial form.

$$p_1 = (\{(0, 0), a\}, \{(0, 1), a\}, \{(1, 0), a\}), \{((1, 1), a), ((1, 2), a), ((2, 1), a)\}),$$

$$p_2 = (\{(0, 0), a\}, \{(0, 1), a\}, \{(1, 0), a\}), \{((1, 1), a)\}).$$

The selector area U_α and the context area U_β are disjoint in a contextual array production. Hence the rules p_1, p_2, \dots, p_8 can be represented by the following patterns where the pixels and symbols of the selector are indicated by enclosing these in boxes:

$$p_1 = \begin{matrix} & a & & \\ \boxed{a} & a & a & \\ \boxed{a} & \boxed{a} & & \end{matrix}, \quad p_2 = \begin{matrix} \boxed{a} & a & & \\ \boxed{a} & \boxed{a} & & \end{matrix}, \quad p_3 = \begin{matrix} & a & & \\ & \boxed{a} & \boxed{a} & \\ & \boxed{a} & \boxed{a} & \end{matrix},$$

$$p_4 = \begin{matrix} \boxed{a} & \boxed{a} & & \\ \boxed{a} & a & & \end{matrix}, \quad p_5 = \begin{matrix} \boxed{a} & a & a & \\ \boxed{a} & \boxed{a} & a & \end{matrix}, \quad p_6 = \begin{matrix} \boxed{a} & a & a & \\ \boxed{a} & \boxed{a} & a & a \end{matrix},$$

$$p_7 = \begin{matrix} & a & a & \\ \boxed{a} & a & & \\ \boxed{a} & \boxed{a} & & \end{matrix}, \quad p_8 = \begin{matrix} & a & & \\ & \boxed{a} & a & \\ & \boxed{a} & \boxed{a} & \end{matrix}$$

A maximal (that is, t -mode) derivation in G generating a 3×4 solid rectangle of a 's, is shown below.

$$\begin{matrix} & a & & a & a & a & & a & a & a & a & & a & a & a & a & & a & a & a & a \\ a & \implies_G & a & a & a & \implies_G & a & a & a & a & \implies_G & a & a & a & a & \implies_G & a & a & a & a & a & a \\ a & a & & a & a & & a & a & & a & a & & a & a & a & & a & a & a & a & a \end{matrix}$$

We note that in this maximal derivation, starting with the axiom array $[{\{(0, 0), a\}, \{(0, 1), a\}, \{(1, 0), a\}}]$, the rules p_1, p_5, p_3, p_4 and again p_4 are applied in this order. After the last rule p_4 is applied, no other rule can be applied and thus the axiom array yields in maximal mode the 3×4 solid rectangle of a 's. In general, in a derivation in the maximal mode, starting with the axiom array, the rule p_1 can be applied any number of times (which will decide the number of rows of the rectangle) until the rule p_2 or p_5 or p_6 is used. While using p_2 will allow only p_3 and p_4 , to be applied a suitable number of times, thus giving a solid square array of a 's, application of p_5 or p_6 (along with p_3 and p_4 applied a suitable number of times) will enable to add more columns of a 's to the right of the rectangle and application of p_7 or p_8 (again along with p_3 and p_4 applied a suitable number of times) will enable to add more rows of a 's to the top of the rectangle. In fact the rules p_1, p_2, p_3, p_4 are given in (Example 4, page 120 of [9]), while illustrating maximal derivation in a contextual array grammar for generating solid squares of a 's. Our additional rules p_5, p_6, p_7, p_8 make possible to generate arbitrary rectangles.

3 A P System Generating Contextual Array Languages

In a string-rewriting P system [16] in each step, each string which can be rewritten by a rule in its region is rewritten but the rewriting of each string is done sequentially, giving rise to transitions between the configurations of the system. A sequence of transitions constitutes a computation. A halting computation is one that reaches a configuration where no rule can be applied. The result of a halting computation is the set of strings sent out of the system during the computation, thus, generating a language. Ceterchi et al [2] introduced an array P system linking the areas of membrane computing and picture grammars and investigated its power in generating array languages. The array P system is a rewriting kind of P system with array objects and array-rewriting rules in its regions. It has an internal output in the sense that the result is obtained in a specified membrane and has halting computations to define correct computations. We now introduce a P system that has similar features but with a difference of having array contextual rules in its membranes as in a contextual array grammar [9].

Definition 2. *A contextual array P system of degree $m \geq 1$ is a construct*

$$\Pi = (V, \#, \mu, A_1, \dots, A_m, P_1, \dots, P_m, i_o),$$

where V is the alphabet, $\#$ is the blank symbol, μ is a membrane structure with m membranes or regions, labelled $1, 2, \dots, m$, in a one-to-one manner, A_1, \dots, A_m are finite sets of arrays over V respectively associated with the m regions of μ , P_1, \dots, P_m are finite sets of array contextual rules over V associated with the m regions of μ ; the rules have attached targets here, out, in (in general, here is omitted). Finally, i_o is the label of the so-called elementary membrane or output membrane which is the innermost membrane of μ and which serves to collect the computation results.

In general, the definition of a P system allows a membrane of μ to contain one or more regions. In the present work, we need only a linear structure of the m regions with the skin membrane being the outermost membrane. The skin membrane contains a membrane inside which in turn contains another membrane inside and so on with the innermost membrane being an elementary membrane. We represent the structure μ as follows: $[_1[_2 \cdot \dots [_m]_m \cdot \dots]_2]_1$, where the skin membrane has label 1 and the innermost elementary membrane has label m , so that the number of regions equals m .

A computation in a contextual array P system is done as follows, with the successful computations being the halting computations. For each array \mathcal{A} in each region of the system, if an array contextual rule p in the region can be applied to \mathcal{A} , then it should be applied which means that the application of a rule is sequential at the level of arrays. The resulting array, if any, is placed in the region indicated by the target associated with the rule used interpreting the attached target as follows: (*here* means that the array remains in the same region, *out* means that the array exits the current membrane and is placed in

the immediately outer membrane if one exists and so, if the application of the rule was in the skin membrane (the outer most membrane enclosing all other membranes), then the array exits the system and arrays leaving the system are considered “lost” in the environment), and *in* means that the array is immediately sent to one of the directly lower membranes, nondeterministically chosen if several exist; (if no internal membrane exists, then a rule with the target indication *in* cannot be used). A computation is successful only if it stops such that a configuration is reached where no rule can be applied to the existing arrays. The result of a halting computation consists of the arrays collected in the membrane with label i_o in the halting configuration. The set of all such arrays computed or *generated* by a system Π is denoted by $CAL(\Pi)$.

The family of all array languages $CAL(\Pi)$ generated by systems Π as mentioned above, with at most m membranes, is denoted by CAP_m .

In order to increase the generative power of a generative device like grammar, several regulating mechanisms have been introduced in formal language theory, both in the string and array cases. We now show that the family of array languages generated by contextual array P system of degree 2 properly includes the family of array languages generated in the maximal mode by the contextual array grammars. This result shows that contextual way of handling array objects in P systems do really increase the generative power of contextual way of rewriting arrays in the t mode.

Theorem 1. $L(cont, t) \subset CAP_2$.

Proof. The inclusion can be seen as follows: Let L be an array language in $L(cont, t)$ generated by a CAG G in maximal mode. We construct a contextual array P system Π with only one membrane which is also the output membrane, containing all the rules of G , each with attached target *here*, and the axiom array of G as its initial object. It is clear that Π generates all and only the arrays of L .

The proper inclusion is seen by considering the array language L_c consisting of all pixel sets forming solid squares of odd side length $2n + 1, n \geq 1$, with the label c in its “central” pixel (the pixel in row $n + 1$ and column $n + 1$) and all other pixels having label a . Fig. 2 shows such a 5×5 solid square with c in its central pixel.

$$\begin{array}{c} a a a a a \\ a a a a a \\ a a c a a \\ a a a a a \\ a a a a a \end{array}$$

Fig. 2. A solid square of a 's with c at the center

The language L_c cannot be generated by any CAG in maximal mode, since adding a context to a selector which has the pixel with label c in it cannot maintain the special symbol c in its central position while choosing a selector anywhere else in the square cannot ensure to make the symbol appear in its

central position. But the following contextual array P system Π_c with two membranes can generate all and only the arrays in L_c .

We construct $\Pi_c = (\{a, c\}, \#, \mu, A_1, A_2, P_1, P_2, 2)$ where $\mu = [{}_1[{}_2]_2]_1$ with membrane labelled 2 inside the membrane labelled 1, $A_1 = \begin{smallmatrix} a \\ c \ a \end{smallmatrix}$, $A_2 = \emptyset$. The set P_1 contains the rule $p_1(in)$, $p_2(in)$ and the set P_2 contains $p_3(out)$, $p_4(out)$, p_5, p_6, p_7, p_8, p_9 . Note that the rules p_5, p_6, p_7, p_8, p_9 all have the target *here*. The rules are given below in pictorial form.

$$\begin{aligned}
 p_1 &= a \begin{array}{c} \boxed{a} \\ \boxed{c} \end{array} a, & p_2 &= a \begin{array}{c} \boxed{a} \\ \boxed{a} \end{array} a, & p_3 &= \begin{array}{c} a \\ \boxed{a} \end{array} a \begin{array}{c} a \\ \boxed{c} \end{array} a, \\
 p_4 &= \begin{array}{c} a \\ \boxed{a} \end{array} a \begin{array}{c} a \\ \boxed{a} \end{array}, & p_5 &= \begin{array}{c} \boxed{a} \\ \boxed{a} \end{array} a \begin{array}{c} a \\ \boxed{a} \end{array}, & p_6 &= a \begin{array}{c} \boxed{a} \\ \boxed{a} \end{array}, & p_7 &= \begin{array}{c} \boxed{a} \\ \boxed{a} \end{array} \begin{array}{c} \boxed{a} \\ \boxed{a} \end{array}, \\
 p_8 &= \begin{array}{c} \boxed{c} \\ \boxed{a} \end{array} \begin{array}{c} \boxed{a} \\ a \end{array}, & p_9 &= \begin{array}{c} a \\ \boxed{a} \end{array} \begin{array}{c} \boxed{a} \\ \boxed{c} \end{array}.
 \end{aligned}$$

Starting from the initial array $\begin{smallmatrix} a \\ c \ a \end{smallmatrix}$ in region 1, the rule $p_1(in)$ is applied (only once) adjoining the context $\begin{smallmatrix} a \\ a \ a \end{smallmatrix}$ to the selector $\begin{smallmatrix} a \\ c \ a \end{smallmatrix}$ and then the resulting array

$\begin{smallmatrix} a \\ a \ c \ a \end{smallmatrix}$ is sent to region 2 due to the target indication *in* in p_1 . Note that there is no initial array in region 2. If the rule $p_3(out)$ is applied (only once) in region

2, then the context $\begin{smallmatrix} a \\ a \ a \end{smallmatrix}$ will be adjoined resulting in the array $\begin{smallmatrix} a \ a \ a \\ a \ c \ a \end{smallmatrix}$ which is

sent back to region 1, due to the target indication *out* in rule p_3 . The rule $p_2(in)$ in region 1, can now be applied and the resulting array sent again to region 2. The process can be repeated by applying the rule $p_4(out)$. If rule p_5 is applied in region 2, the process of “filling” the north-east corner with a is done and the array remains in region 2 itself due to the target indication *here* understood in rule p_5 . This is then followed by application of rules p_6, p_7 suitable number of times “filling” the remaining pixels to the left and right of the diagonal joining the south-west corner to north-east corner. The rules p_8, p_9 are needed only once for starting the filling of the north-west part and of the south-east part of the square to be generated. At the end, the process comes to a halt and thus in the halting configuration, a solid square array of odd side length with c in its central position and a 's for all other pixels is generated. This proves the result. \square

Generation of solid and hollow rectangles and squares is one of the problems of interest and investigation in the study of two-dimensional picture grammars.

In [27], a regular array grammar [3] generating solid squares of a 's, is given while it is shown that hollow rectangles made of a 's cannot be generated by even a context-free array grammar [27]. On the other hand hollow rectangles, two of whose arms (left and bottom arms) are made of a 's and the other two (right and top arms) are made of b 's are shown to be generated by a contextual array grammar in [7,9]. Here we give a contextual array P system with one membrane generating hollow rectangles made of only one symbol a . This is an improvement over the contextual array grammar for hollow rectangles given in [7,9]. It is interesting to note that an array P system with array rewriting rules is given in (Example 3, page 10 [2]) generating hollow rectangles of a 's, which motivates our construction.

Theorem 2. *The array language L_H consisting of hollow rectangles of a 's belongs to CAP_1 .*

Proof. The array language L_H is generated by a contextual array P system Π_H with just one membrane. $\Pi_H = (\{a\}, \#, [1]_1, A, P, 1)$ where A contains the initial array a , and P has the rules p_1, p_2, p_3, p_4, p_5 given below in pictorial form.

$$\begin{aligned}
 p_1 &= \begin{array}{|c|} \hline \# \\ \hline \end{array} \begin{array}{|c|} \hline a \\ \hline \end{array} \begin{array}{|c|} \hline \# \\ \hline \end{array}, & p_2 &= \begin{array}{|c|} \hline \# \\ \hline a \\ \hline \# \\ \hline \end{array} a, & p_3 &= \begin{array}{|c|} \hline \# \\ \hline \# \\ \hline \end{array} \begin{array}{|c|} \hline a \\ \hline \end{array} a, \\
 p_4 &= \begin{array}{|c|} \hline \# \\ \hline a \\ \hline \end{array} \#, & p_5 &= \begin{array}{|c|} \hline \# \\ \hline \end{array} \begin{array}{|c|} \hline a \\ \hline \end{array} \begin{array}{|c|} \hline \# \\ \hline \end{array}.
 \end{aligned}$$

Starting with the initial array in the only region 1, the rule p_1 allows the left vertical arm of the hollow rectangle of a 's to be generated, to grow up and the rule p_2 allows the bottom (as well as top) horizontal arms to grow right, while the rule p_3 allows the left vertical arm to turn to the right and the rule p_4 allows the top horizontal arm to turn down so as to grow the right vertical arm by using rule p_5 , until the bottom horizontal arm and right vertical arm meet at their ends, thereby reaching a halting configuration and the computation stops generating a hollow rectangle of a 's. Any other incorrect computation will not result in a halting configuration, and will not contribute anything to the language. Note that the halting condition is crucial for a successful computation. \square

Another class of two-dimensional array grammars, called parallel contextual array grammars, that extend the contextual operations on strings to arrays and generate languages of rectangular arrays has been introduced in [11] and subsequently, properties of several variants of these parallel contextual grammars have been obtained in [25]. Here we compare the class CAP of array languages considered here with a class called EPCA in [25] of array languages generated by external parallel contextual array grammars (EPCAG). This kind of EPCAG

[25] generates rectangular arrays by adjoining arrays to the left and right or to the top and bottom of the array, decided by certain conditions, analogous to the external contextual grammars in the string case [15] that adjoin strings to the left and right of a current string during a generation process.

Theorem 3. $CAP_1 - EPCA \neq \emptyset$.

Proof. It is known [25] that the array language $M = M' \cup M''$ where

$$M' = \left\{ \begin{pmatrix} a \\ a \\ a \end{pmatrix}^{2n+1} \mid n = 0, 1, \dots \right\}, M'' = \left\{ \begin{matrix} a & \begin{pmatrix} a \\ a \\ a \end{pmatrix}^{2n+1} & b \\ b & & a \\ a & & b \end{matrix} \mid n = 0, 1, \dots \right\}$$

cannot be generated by any external parallel contextual array grammar and hence does not belong to EPCA. But it belongs to CAP_1 since it is generated by the contextual array P system $\Pi = (\{a\}, \#, [1]_1, A, P, 1)$ where A contains

the two initial arrays $\begin{matrix} a \\ a \end{matrix}$, $\begin{matrix} a \\ a \end{matrix}$, and P has the rules p_1, p_2, p_3, p_4 given below in pictorial form.

$$p_1 = \begin{matrix} a \\ \boxed{a} \\ \boxed{a} \\ \boxed{\#} \end{matrix}, \quad p_2 = \begin{matrix} a & a \\ \boxed{a} & a \ a \\ \boxed{a} & a \ a \\ \boxed{\#} \end{matrix}, \quad p_3 = \begin{matrix} a & b \\ a & a \\ \boxed{a} & b \\ \boxed{\#} \end{matrix}, \quad p_4 = \begin{matrix} a & a \\ a & a \\ \boxed{a} & a \ a \\ \boxed{\#} \end{matrix}.$$

The rules p_1, p_2 generate the arrays in M' from the initial array $\begin{matrix} a \\ a \end{matrix}$ while the

rules p_3, p_4 generate the arrays in M'' from the initial array $\begin{matrix} a \\ b \\ a \end{matrix}$. □

4 Conclusion

We have introduced a P system with array objects and array contextual rules of the kind considered in [9] and we have shown that the P system model has more generative power than the contextual array grammars [9] working in maximal mode in generating two-dimensional arrays. We can also consider other modes of derivation and examine the power of contextual array P systems. Also the contextual style of generating strings has been extended to arrays in different ways [11,12,25]. These models belong to the category of non-isometric variety and generate rectangular arrays. We have made a comparison with a class in [11,25]. Comparison of the contextual array P system considered here with other classes in [25] could be studied. The model in [12] is more in line with the array grammars considered in [22] and based on this work a P system generating rectangular arrays is considered in [5]. A comparison of the P system model introduced here with the models in [12,5] could also be studied.

Acknowledgements. The authors thank the referees for their valuable comments which enabled to improve the presentation of the paper. The first and second authors gratefully acknowledge support for this research from a FRGS grant No. 203/PKOMP/6711267 of the Ministry of Higher education, Malaysia.

References

1. Bottoni, P., Labella, A., Martín-Vide, C., Păun, G.: Rewriting P Systems with Conditional Communication. In: Brauer, W., Ehrig, H., Karhumäki, J., Salomaa, A. (eds.) *Formal and Natural Computing*. LNCS, vol. 2300, pp. 325–353. Springer, Heidelberg (2002)
2. Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G.: Array - rewriting P systems. *Natural Computing* 2, 229–249 (2003)
3. Cook, C.R., Wang, P.S.-P.: A Chomsky hierarchy of isotonic array grammars and languages. *Computer Graphics and Image Processing* 8, 144–152 (1978)
4. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G.: *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Yverdon. Topics in Computer Mathematics 5. Gordon and Breach Science Publishers (1994)
5. Dersanambika, K.S., Krithivasan, K.: Contextual array P systems. *International Journal of Computer Mathematics* 81(8), 955–969 (2004)
6. Ehrenfeucht, A., Păun, G., Rozenberg, G.: Contextual grammars and formal languages. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, vol. 2, pp. 237–293. Springer, Berlin (1997)
7. Fernau, H., Freund, R., Holzer, M.: Representations of recursively enumerable array languages by contextual array grammars. *Fundamenta Informaticae* 64, 159–170 (2005)
8. Ferretti, C., Mauri, G., Păun, G., Zandron, C.: On three variants of rewriting P systems. *Theoretical Computer Science* 301, 201–215 (2003)
9. Freund, R., Păun, G., Rozenberg, G.: Contextual array grammars. In: Subramanian, K.G., Rangarajan, K., Mukund, M. (eds.) *Formal Models, Languages and Applications Series in Machine Perception and Artificial Intelligence*, vol. 66, pp. 112–136. World Scientific (2007)
10. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, vol. 3, pp. 215–267. Springer (1997)
11. Helen Chandra, P., Subramanian, K.G., Thomas, D.G.: Parallel contextual array grammars and languages. *Electronic Notes in Discrete Math.* 12, 106–117 (2003)
12. Krithivasan, K., Balan, M.S., Rama, R.: Array contextual grammars. In: Martín-Vide, C., Păun, G. (eds.) *Recent Topics in Mathematical and Computational Linguistics*, pp. 154–168 (2000)
13. Krithivasan, K., Mutyam, M.: Contextual P Systems. *Fundam. Inform.* 49, 179–189 (2002)
14. Marcus, S.: Contextual grammars. *Rev. Roum. Math. Pures Appl.* 14, 1525–1534 (1969)
15. Păun, G.: *Marcus Contextual Grammars*. Studies in Linguistics and Philosophy. Kluwer Academic Publishers, Dordrecht (1997)
16. Păun, G.: Computing with membranes. *J. Comp. System Sci.* 61, 108–143 (2000)
17. Rosenfeld, A.: *Picture Languages*. Academic Press, Reading (1979)
18. Rosenfeld, A., Kak, A.: *Digital Picture Processing*, 2nd edn. Academic Press (1982)

19. Rosenfeld, A., Siromoney, R.: Picture languages – a survey. *Languages of Design* 1, 229–245 (1993)
20. Rozenberg, G., Salomaa, A. (eds.): *Handbook of Formal Languages*, 3 volumes. Springer, Berlin (1997)
21. Salomaa, A.: *Formal Languages*. Academic Press, Reading (1973)
22. Siromoney, G., Siromoney, R., Krithivasan, K.: Picture languages with array rewriting rules. *Information and Control* 22, 447–470 (1973)
23. Subramanian, K.G.: P Systems and Picture Languages. In: Durand-Lose, J., Margenstern, M. (eds.) *MCU 2007. LNCS*, vol. 4664, pp. 99–109. Springer, Heidelberg (2007)
24. Subramanian, K.G., Ali, R.M., Geethalakshmi, M., Nagar, A.K.: Pure 2D picture grammars and languages. *Discrete Appl. Math.* 157, 3401–3411 (2009)
25. Subramanian, K.G., Van, D.L., Helen Chandra, P., Quyen, N.D.: Array grammars with contextual operations. *Fundam. Inform.* 83, 411–428 (2008)
26. Wang, P.S.-P. (ed.): *Array Grammars, Patterns and Recognizers*. Series in Computer Science, vol. 18. World Scientific (1989)
27. Yamamoto, Y., Morita, K., Sugata, K.: Context-sensitivity of two-dimensional regular array grammars. In: Wang, P.S.-P. (ed.) *Array Grammars, Patterns and Recognizers*. WSP Series in Computer Science, vol. 18, pp. 17–41. World Scientific Publ., Singapore (1989)
28. Zandron, C., Ferretti, C., Mauri, G.: Two Normal Forms for Rewriting P Systems. In: Margenstern, M., Rogozhin, Y. (eds.) *MCU 2001. LNCS*, vol. 2055, pp. 153–164. Springer, Heidelberg (2001)

Rectangular Arrays and Petri Nets

D. Lalitha¹, K. Rangarajan², and Durairaj Gnanaraj Thomas³

¹ Department of Mathematics, Sathyabama University
Chennai - 600 119, India

lalkrish_24@yahoo.co.in

² Department of Mathematics, Barath University, Chennai - 600 073, India

kr2210@hotmail.com

³ Department of Mathematics, Madras Christian College
Tambaram, Chennai - 600 059, India

dgthomasccc@yahoo.com

Abstract. Array Token Petri Net Structure (ATPNS) to generate rectangular arrays has been defined in [6]. We prove that this model generate the regular array languages. By introducing a control on the firing sequence, we have shown that, ATPNS with inhibitor arcs generate the context-free and context-sensitive array languages. Comparisons with other classes of array languages have been made.

Keywords: Array token, Catenation, Inhibitor arcs, Petri net, Picture languages.

1 Introduction

Picture languages generated by grammars or recognized by automata have been advocated since the seventies for problems arising in the framework of pattern recognition and image analysis [2,7,9]. In syntactic approaches to generation of picture patterns, several two-dimensional grammars have been proposed. Array rewriting grammars [11], controlled tabled L array grammars [10] and pure 2D context-free grammars [13] are some of the picture generating devices. Applications of these models to generation of “kolam” patterns [12] and in clustering analysis [14] are found in the literature.

On the other hand, a Petri net is an abstract formal model of information flow [4]. Petri nets have been used for analyzing systems that are concurrent, asynchronous, distributed, parallel, non deterministic and/or stochastic. Tokens are used in Petri nets to simulate dynamic and concurrent activities of the system. A language can be associated with the execution of a Petri net. By defining a labeling function for transitions over an alphabet, the set of all firing sequences, starting from a specific initial marking leading to a finite set of terminal markings, generates a language over the alphabet.

Petri net structure to generate rectangular arrays are found in [5,6]. The two models have different firing rules and catenation rules. In [5] column row catenation petri net structure (CRCPNS) has been defined. A transition with

several input places having different arrays is associated with a catenation rule as label. The label of the transition decides the order in which the arrays are joined (columnwise or rowwise) provided the condition for catenation is satisfied. In CRCPNS a transition with a catenation rule as label and different arrays in the input places is enabled to fire.

In ATPNS [6] the catenation rule involves an array language. All the input places of the transition with a catenation rule as label, should have the same array as token, for the transition to be enabled. The size of the array language to be joined to the array in the input place, depends on the size of the array in the input place.

In this paper we examine the generative capacity of ATPNS. We find that ATPNS is able to generate only the regular languages. To control the firing sequence inhibitor arcs are introduced. The introduction of inhibitor arcs increases the generative capacity. ATPNS with inhibitor arcs generate the context-free and context-sensitive languages.

The paper is organized as follows: Section 2 defines Array Token Petri Net Structure (ATPNS), language associated with it and gives an example. Section 3 compares ATPNS with three families of array grammars, TOL array grammar with regular control and pure 2D context-free grammar. Section 4 defines Array Token Petri Net Structure with inhibitor arcs, compares with the other six families of array grammars and TOL array grammar with context-free or context-sensitive control.

2 Array Token Petri Nets

In this section we give preliminary definitions of Petri Net and give the notations used. We define Array Token Petri Net Structure (ATPNS), language associated with it and give an example.

A Petri net is one of several mathematical models for the description of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e., events that may occur, signified by bars) and places (i.e., conditions, signified by circles). The directed arcs from places to a transition denote the pre-conditions and the directed arcs from the transition to places denote the post-conditions (signified by arrows). Graphically, places in a Petri net may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called a marking. In an abstract sense relating to a Petri net diagram, a transition of a Petri net may fire whenever there are sufficient tokens at the start of all input arcs; when it fires, it consumes these tokens, and places tokens at the end of all output arcs. Transitions can be labeled with elements of an alphabet so that the firing sequence corresponds to a string over the alphabet. A labeled Petri net generates a language. Hack [3] and Baker [1] have published a report on Petri net languages. Petri net to generate string languages is also found in [8].

We now recall the basic definitions of Petri net [4] and the basic notations pertaining to rectangular arrays [11].

Definition 1. A Petri Net structure is a four tuple $C = (P, T, I, O)$ where $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places, $n \geq 0$, $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transitions $m \geq 0$, $P \cap T = \phi$, $I : T \rightarrow P^\infty$ is the input function from transitions to bags of places and $O : T \rightarrow P^\infty$ is the output function from transitions to bags of places.

Definition 2. A Petri Net marking is an assignment of tokens to the places of a Petri Net. The tokens are used to define the execution of a Petri Net. The number and position of tokens may change during the execution of a Petri Net. In this paper arrays over an alphabet are used as tokens.

Definition 3. An inhibitor arc from a place p_i to a transition t_j has a small circle in the place of an arrow in regular arcs. This means the transition t_j is enabled only if p_i has no tokens. A transition is enabled only if all its regular inputs have tokens and all its inhibitor inputs have zero tokens.

Basic Notations: Σ^{**} denotes the arrays made up of elements of Σ . If A and B are two arrays having same number of rows then $A \oplus B$ is the column wise catenation of A and B . If two arrays have the same number of columns then $A \ominus B$ is the row wise catenation of A and B . $(x)^n$ denotes a horizontal sequence of n 'x' and $(x)_n$ denotes a vertical sequence of n 'x' where $x \in \Sigma^{**}$. $(x)^{n+1} = (x)^n \oplus x$ and $(x)_{n+1} = (x)_n \ominus x$.

The Petri net model defined here has places and transitions connected by directed arcs. Rectangular arrays over an alphabet are taken as tokens to be distributed in places. Variation in firing rules and labels of the transition are listed out below.

Firing Rules in ATPNS

We define three different types of enabled transition in ATPNS. The pre and post condition for firing the transition in all the three cases are given below:

1. When all the input places of t_1 (without label) have the same array as token.
 - Each input place should have at least the required number of arrays.
 - Firing t_1 removes arrays from all the input places and moves the array to all its output places.

The graph in Fig. 1 shows the position of the array before the transition fires and Fig. 2 shows the position of the array after transition t_1 fires.

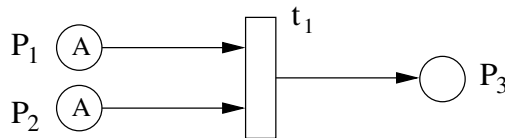


Fig. 1. Position of arrays before firing

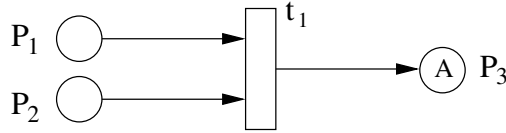


Fig. 2. Position of array after firing

2. When all the input places of t_1 have different arrays as token
 - The label of t_1 designates one of its input places.
 - The designated input place has the same array as tokens.
 - The designated input place has sufficient number of tokens.
 - Firing t_1 removes arrays from all the input places and moves the array from the designated input place to all its output places.

The graph in Fig. 3 shows the position of the array before the transition fires and Fig. 4 shows the position of the array after transition t_1 fires. Since the designated place is p_1 the array in p_1 is moved to the output place.

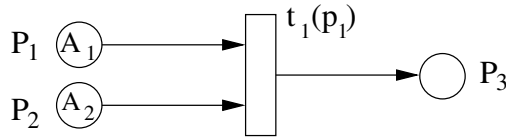


Fig. 3. Transition with label before firing

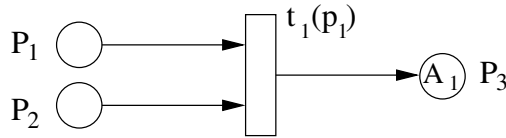


Fig. 4. Transition with label after firing

3. When all the input places of t_1 (with catenation rule as label) have the same array as token
 - Each input place should have at least the required number of arrays.
 - The condition for catenation should be satisfied.
 - The designated input place has sufficient number of tokens.
 - Firing t_1 removes arrays from all the input places p and the catenation is carried out in all its output places.

Catenation Rule as Label for Transitions: Column catenation rule is in the form $A \oplus B$. Here the array A denotes the $m \times n$ array in the input place of the transition. B is an array language whose number of rows will depend on ‘ m ’ the number of rows of A . The number of columns of B is fixed. For example $A \oplus (x \ x)_m$ adds two columns of x after the last column of the array A which is in the input place. But $(x \ x)_m \oplus A$ would add two columns of x before the first column of A . ‘ m ’ always denotes the number of rows of the input array A . Row catenation rule is in the form $A \ominus B$. Here again the array A denotes the $m \times n$ array in the input place of the transition. B is an array language whose number of columns will depend on ‘ n ’ the number of columns of A . The number of rows of B is always fixed. For example $A \ominus [x]^n$ adds two rows of x after the last row of the array A which is in the input place. But $[x]^n \ominus A$ would add two rows of x before the first row of the array A . ‘ n ’ always denotes the number of columns of the input array A .

An example to explain row catenation rule is given below. The position of the arrays before the transition fires is shown in Fig. 5 and Fig. 6 shows the position of the array after transition t_1 fires. Since the catenation rule is associated with the transition, catenation takes place in p_3 .

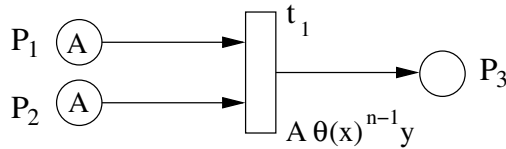


Fig. 5. Transition with catenation rule before firing

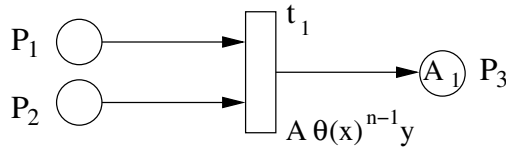


Fig. 6. Transition with catenation rule after firing

$a \ a \ a$

If $A = a \ a \ a$, the number of columns of A is 3, $n - 1$ is 2, firing t_1 adds the

$a \ a \ a$

row $x \ x \ y$ as the last row. Hence $A_1 = \begin{matrix} a & a & a \\ a & a & a \\ a & a & a \\ x & x & y \end{matrix}$.

Definition 4. If $C = (P, T, I, O)$ is a Petri net structure with arrays over of Σ^{**} as initial markings, $M_0 : P \rightarrow \Sigma^{**}$, label of at least one transition being catenation rule and a finite set of final places $F \subset P$, then the Petri net structure C is defined as Array Token Petri Net Structure (ATPNS).

Definition 5. If C is a ATPNS then the language generated by the Petri net C is defined as $L(C) = \{A \in \Sigma^{**} / A \text{ is in } p \text{ for some } p \text{ in } F\}$. With arrays of Σ^{**} in some places as initial marking all possible sequences of transitions are fired. The set of all arrays collected in the final places F is called the language generated by C .

Example 1. $\Sigma = \{x, \cdot\}$, $F = \{p_1\}$

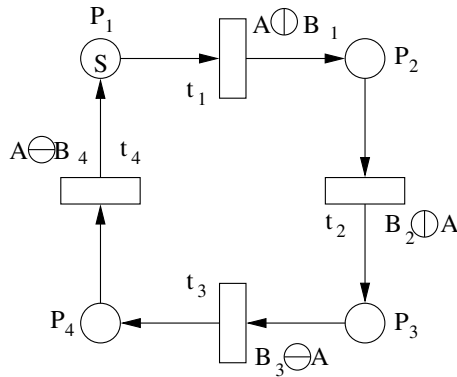


Fig. 7. ATPNS

where $S = \begin{matrix} x & x & x \\ x & \cdot & x \\ \cdot & \cdot & x \end{matrix}$, $B_1 = (\cdot \ x)_m$, $B_2 = (x \ \cdot)_m$, $B_3 = \begin{matrix} x & \binom{x}{x}^{n-2} & x \\ & x & \end{matrix}$ and $B_4 = \begin{matrix} \binom{x}{\cdot}^{n-2} & \cdot & x \\ & \cdot & x \end{matrix}$

Firing t_1 puts an array in p_2 making t_2 enabled. Firing t_2 puts an array in p_3 making t_3 enabled. Firing t_3 puts an array in p_4 making t_4 enabled. Firing t_4 puts an array in p_1 . The firing sequence $(t_1 t_2 t_3 t_4)^k$, $k \geq 0$ puts a square spiral of size $4k + 3$ in p_1 . The language generated by this ATPNS is a set of square spirals. When the transitions t_1, t_2, t_3 and t_4 fire the array that reaches the output place is shown below

$$\begin{array}{rcccc}
 & & & & x & x & x & x & x & x \\
 S & \xrightarrow{t_1} & \begin{matrix} x & x & x & \cdot & x \\ x & \cdot & x & x & x \\ \cdot & \cdot & x & x & x \end{matrix} & \xrightarrow{t_2} & \begin{matrix} x & \cdot & x & x & x & \cdot & x \\ x & \cdot & x & \cdot & x & x & x \\ \cdot & \cdot & x & \cdot & x & x & x \end{matrix} & \xrightarrow{t_3} & \begin{matrix} x & \cdot & x & x & x & \cdot & x \\ x & \cdot & x & \cdot & x & x & x \\ \cdot & \cdot & x & \cdot & x & x & x \end{matrix} & & \begin{matrix} x & \cdot & x & x & x & \cdot & x \\ x & \cdot & x & \cdot & x & x & x \\ \cdot & \cdot & x & \cdot & x & x & x \end{matrix}
 \end{array}$$

$$\begin{array}{c}
 x x x x x x x \\
 x x \\
 x . x x x . x \\
 \xRightarrow{t_A} x . x . x . x \\
 x . . . x . x \\
 x x x x x . x \\
 x
 \end{array}$$

The language generated by this ATPNS is square spirals of size $4n + 3, n \geq 0$.

3 Comparative Results

In this section we recall the definitions of Array rewriting Grammar [11], Extended Controlled Table L -array Grammar [10], pure 2D context-free grammar with regular control [13] and compare it with ATPNS.

Definition 6. $G = (V, I, P, S)$ is an array rewriting grammar (AG), where $V = V_1 \cup V_2$, V_1 a finite set of nonterminals, V_2 a finite set of intermediates, I a finite set of terminals, $P = P_1 \cup P_2 \cup P_3$, P_1 is the finite set of nonterminal rules, P_2 is the finite set of intermediate rules, P_3 is the finite set of terminal rules. $S \in V_1$ is the start symbol. P_1 is a finite set of ordered pairs (u, v) , u and v in $(V_1 \cup V_2)^+$ or u and v in $(V_1 \cup V_2)_+$.

P_1 is context-sensitive if there is a (u, v) in P_1 such that $u = u_1 S_1 v_1$ and $v = u_1 \alpha v_1$ where $S_1 \in V_1$, u_1, α, v_1 are all in $(V_1 \cup V_2)^+$ or all in $(V_1 \cup V_2)_+$. P_1 is context-free if every (u, v) in P_1 is such that $u \in V_1$ and v in $(V_1 \cup V_2)^+$ or $(V_1 \cup V_2)_+$. P_1 is regular if $u \in V_1$ and v of the form $U \oplus V$, U in V_1 and V in V_2 or U in V_2 and V in V_1 .

P_2 is a set of ordered pairs (u, v) , u and v in $(V_2 \cup \{x_1, \dots, x_p\})^+$ or u and v in $(V_2 \cup \{x_1, \dots, x_p\})_+$; x_1, \dots, x_p in I^{++} have same number of rows in the first case and same number of columns in the second case. P_2 is called CS, CF or R as the intermediate matrix languages generated are CS, CF or R.

P_3 is a finite set of terminal rules are ordered pairs (u, v) , u in $(V_1 \cup V_2)$ and v in I^{++} .

An Array Grammar is called $(CS : CS)AG$ if the nonterminal rules are CS and at least one intermediate language is CS. An Array Grammar is called $(CS : CF)AG$ if the nonterminal rules are CS and none of the intermediate language is CS. An Array Grammar is called $(CS : R)AG$ if the nonterminal rules are CS and all the intermediate languages are regular. Similarly all the other six families $(CF : CS)AG$, $(CF : CF)AG$, $(CF : R)AG$, $(R : CS)AG$, $(R : CF)AG$ and $(R : R)AG$ are defined. $(X : Y)AL$ refers to the language generated by the $(X : Y)AG$, where $X, Y \in \{R, CF, CS\}$.

Definition 7. An extended, controlled $\langle k_l, k_r, k_u, k_d \rangle$ table L -array grammar is a 5-tuple $G = (V, T, \mathcal{P}, C, S, \#)$ where V is a finite nonempty set; $T \subseteq V$ is the terminal alphabet of G ; \mathcal{P} is a finite set of tables $\{P_1, P_2, \dots, P_k\}$, and each $P_i, i = 1, 2, \dots, k$, is a left, right, up or down rules only. The rules within a table

are all of the same type: either string rules with neighborhood context determined by $k_l, k_r, k_u, k_d \in \{0, 1\}$, or matrix rules. In either case, all right-hand sides of rules within the same table are of the same length; C is a control language over \mathcal{P} ; and $S \notin V$ is the start matrix; $\#$ is an element not in V .

In particular if $V = T$ and S is a matrix, G is a controlled table L -array grammar; if $C = \mathcal{P}^*$, then there is no control and the order of applications of the tables is arbitrary; G is then an extended table L -array grammar.

If $k_l = k_r = k_u = k_d = 0$, then the rules are all context-free (0L) table array grammar. If at least one of k_l, k_r, k_u, k_d equals 1 then we get a context dependent (1L) table array grammar.

$(\gamma)TXLAL$ refers to the language generated by table XL array grammar with γ control; X may be 0 or 1 and γ may be R , CF or CS .

Definition 8. A pure 2D context-free grammar (P2DCFG) is a 4-tuple $G = (\Sigma, P_c, P_r, \mathcal{M}_0)$, where

- Σ is a finite set of symbols.
- $P_c = \{t_{c_i} | 1 \leq i \leq m\}$, $P_r = \{t_{r_j} | 1 \leq j \leq n\}$.
Each t_{c_i} , ($1 \leq i \leq m$), called a column table, is a set of context-free rules of the form $a \rightarrow \alpha$, $a \in \Sigma$, $\alpha \in \Sigma^*$ such that for any two rules $a \rightarrow \alpha$, $b \rightarrow \beta$ in t_{c_i} , we have $|\alpha| = |\beta|$, where $|\alpha|$ denotes the length of α .

Each t_{r_j} , ($1 \leq j \leq n$), called a row table, is a set of context-free rules of the form $c \rightarrow \gamma^T$, $x \in \Sigma$, $\gamma \in \Sigma^*$ such that for any two rules $c \rightarrow \gamma^T$, $d \rightarrow \delta^T$ in t_{r_j} , we have $|\gamma| = |\delta|$.

- $\mathcal{M}_0 \subseteq \Sigma^{**} - \{\lambda\}$ is a finite set of axiom arrays.

Derivations are defined as follows. For any two arrays M_1, M_2 , we write $M_1 \Rightarrow M_2$ if M_2 is obtained from M_1 by either rewriting a column of M_1 by rules of some column table t_{c_i} in P_c or a row of M_1 by rules of some row table t_{r_j} in P_r . \Rightarrow^* is the reflexive transitive closure of \Rightarrow .

The picture array language $L(G)$ generated by G is the set of rectangular picture arrays $\{M | M_0 \Rightarrow^* M \in \Sigma^{**}, \text{ for some } M_0 \in \mathcal{M}_0\}$. The family of picture array languages generated by pure 2D context-free grammars is denoted by P2DCFL.

Definition 9. A pure 2D context-free grammar with a regular control is $G_c = (G, \text{Lab}(G), \mathcal{C})$ where G is a pure 2D context-free grammar, $\text{Lab}(G)$ is a set of labels of the tables of G and $\mathcal{C} \subseteq \text{Lab}(G^*)$ is a regular (string) language. The words in $\text{Lab}(G)^*$ are called control words of G . Derivations $M_1 \Rightarrow_w M_2$ in G_c are done as in G , except that if $w \in \text{Lab}(G^*)$ and $w = l_1 l_2 \dots l_m$, then the tables of rules with labels l_1, l_2, \dots, l_m are successively applied starting with M_1 to yield M_2 . The picture array language generated by G_c consists of all picture arrays obtained from the axiom array of G with the derivations controlled as described above. We denote by $(R)P2DCFL$ the family of picture array languages generated by pure 2D context-free grammars with a regular control.

Theorem 1. The class of table 0L array languages without control or with regular control can be generated by ATPNS.

Proof. Let $G = (V, \mathcal{P}, C, S)$ be a table 0L array grammar; where V is a finite set of terminals, \mathcal{P} is a finite set of tables $\{P_1, P_2, \dots, P_k\}$, and each P_i , $i = 1, 2, \dots, k$, is a left, right, up or down rules only. S is the start array.

The rules within a table are all of the same type. The left hand side of each production is a single terminal. The right hand side of all the rules within the same table is of the same length. Each table will have at least one rule for each symbol on the boundary. If say, P_1 has left (right) rules then applying the rules of P_1 will amount to column catenation. Similarly applying the table containing up(down) rules will amount to row catenation.

Let us construct an array token Petri net structure when there is no control on the application of the tables. Let p_1 be the place with the start array S as a token. For every table $P_i \in \mathcal{P}$ have a transition t_i with the corresponding row or column catenation rule as label. Have k transitions one each for the k tables in \mathcal{P} with p_1 as both input place and output place of all the transitions. $F = \{p_1\}$. Every time a table production is required to be used the corresponding transition is fired. Since there is no control the tables can be applied in any order to generate the language. In the net p_1 is the output place of every transition. Hence after the firing of any transition the array reaches p_1 , so at any given time all the k transitions are enabled. Thus the Petri net constructed will generate the language generated by the grammar G .

Let us construct an array token Petri net structure when a regular control $C = (P_1 P_2 \dots P_k)^*$ is defined on the application of tables. Have k transitions and k places. Let S the start array be a token in place p_1 . Let t_1 be a transition with the catenation rule, which corresponds to the table P_1 , as label; p_1 being the input place and p_2 as its output place. Let t_2 be a transition with the catenation rule, which corresponds to the table P_2 , as label; p_2 being the input place and p_3 as its output place. Continuing like this have a transition t_k with the catenation rule, which corresponds to the table P_k , as label; p_k being the input place and p_1 as its output place. $F = \{p_1\}$. Firing t_k puts a token in p_1 so that t_1 is enabled again. The firing sequence $t_1 t_2 \dots t_k$ will have the same effect as applying the production rules $P_1 P_2 \dots P_k$ in that order once. The effect of the regular control is got by placing the transitions with those labels in the same order forming a loop in the net so that the sequence of transitions can be fired any number of times. Thus the Petri net constructed will generate the language generated by the grammar G . \square

Theorem 2. $(R)T0LAL$ is properly contained in the family generated by ATPNS.

Proof. Let us give an example to prove this theorem. The language of square spirals given in Example [11](#) is a $(R)T1LAL$ [\[10\]](#). Thus ATPNS can generate certain languages that do not belong to $(R)T0LAL$, which proves a proper containment. \square

Theorem 3. The families of $(R : Y)AL$, where $Y \in \{R, CF, CS\}$, can be generated by ATPNS.

Proof. Let $G = (V, I, P, S)$ be an array grammar, where $V = V_1 \cup V_2$, V_1 a finite set of nonterminals; V_2 a finite set of intermediates; I a finite set of terminals,

$P = P_1 \cup P_2$, P_1 is the finite set of regular nonterminal rules; P_2 is the finite set of terminal rules. $S \in V_1$ is the start symbol. For each A in V_2 , M_A is an intermediate language.

In array grammars the derivation is as follows. Starting with S the nonterminal rules are applied without any restriction, as in string grammar, till all the nonterminals are replaced. Then replace each intermediate A by M_A subject to the conditions imposed by the row and column catenation operator. Let the regular non-terminal rules of G generate the infinite sequence of matrices $\{M_n/n \geq 1\}$ where M_n is in any one of the following forms. For all $n > 1$, $M_n = (X \ominus M_{n-1}) \oplus Y$ or $M_n = Y \oplus (X \ominus M_{n-1})$ or $M_n = Y \oplus (M_{n-1} \ominus X)$ or $M_n = (M_{n-1} \ominus X) \oplus Y$, where X and Y are chosen from intermediate matrix languages L_X and L_Y (subject to conditions imposed by row and column catenation). The recursive definition of M_n is assumed to be unique. $S \rightarrow M_1$ is the terminal rule.

Construction of ATPNS, for the case when $M_n = (X \ominus M_{n-1}) \oplus Y$ where X and Y are intermediates, is given below. For the other cases the construction is similar. Define the arrays B_X and B_Y corresponding to the intermediate language X and Y . Put M_1 in the start place p_1 as a token. Have a transition t_1 with the row catenation rule $B_X \ominus A$ as a label. Let p_1 be the input place of t_1 . The number of rows of B_X is fixed but the number of columns of B_X is dependent on ‘ n ’ the number of columns of A . A is the array that reaches the input place p_1 of the transition t_1 during the course of the execution of the net. Let p_2 be the output place for the transition t_1 . The array B_Y is defined similar to the intermediate language generated by Y . Have a transition t_2 with the column catenation rule $A \oplus B_Y$ as a label. Let p_2 be the input place of t_2 . The number of columns of B_Y is fixed but the number of rows of B_X is dependent on ‘ m ’ the number of rows of A . A is the array that reaches the input place p_2 of the transition t_2 during the course of the execution of the net. Let p_1 be the output place for the transition t_2 . First time the sequence $t_1 t_2$ is executed, the matrix M_2 is put in p_1 . Let $F = \{p_1\}$ be the final set of places. The firing sequence $(t_1 t_2)^k$ puts the matrix M_{k+1} , $k \geq 0$ in p_1 . Thus $\{M_n/n \geq 1\}$ of matrices is the language generated. \square

Theorem 4. *The families of $(R : Y)AL$, where $Y \in \{R, CF, CS\}$, are properly contained in the family generated by ATPNS.*

Proof. Let us give an example to prove this theorem. Kirsch’s right triangles is a $(CF : R)AL$ [11]. But ATPNS can generate Kirsch’s right triangles. Thus ATPNS generates certain languages which do not belong to $(R : Y)AL$, which proves proper containment. \square

Theorem 5. *Any $(R)P2DCFL$ can be generated by ATPNS.*

Proof. Let the language be generated by a P2DCFG with a regular control, $G_c = (G, Lab(G), \mathcal{C})$ where G is a P2DCFG, $Lab(G)$ is the set of all labels and \mathcal{C} is the control language.

Application of a column table is equivalent to a column catenation. Hence for every t_{c_j} , we can define an equivalent column catenation rule. Similarly for every row table t_{r_j} , an equivalent row catenation rule can be defined.

Let M_1 be an array derived from the axiom array M_0 using the control words $w = l_1l_2 \dots l_m$. We give the steps for constructing the ATPNS to generate the language, assuming that all the tables are used on the boundary of M_0 .

Let p_0 be a place with the array M_0 as token. Let t_1 be a transition with the catenation rule corresponding to l_1 as a label, p_0 as input place and p_1 as output place. Let t_2 be a transition with the catenation rule corresponding to l_2 as label, p_1 as input place and p_2 as output place. Proceeding like this let t_m be a transition with the catenation rule corresponding to l_m as label, p_{m-1} as input place and p_0 as output place. The firing sequence $t_1t_2 \dots t_m$ has the same effect as of applying the tables l_1, l_2, \dots, l_m to the array M_0 . This Petri net structure generates all the arrays that can be generated by the control words of $(l_1l_2 \dots l_m)^*$.

If all the tables are not applied on the boundary of M_0 , then consider a subarray M_{01} of M_0 such that the table l_1 is applied to the boundary of M_{01} . Take M_{01} as a token in p_0 and construct the ATPNS as given above. Add a transition t_{m+1} with input place p_0 and output place p_m . The label of the transition should have the catenation rule, which joins the row/column that was removed from M_0 . Required number of transitions should be added to join all the rows/columns that were removed from M_0 . □

4 Array Token Petri Nets with Inhibitor Arcs

Since ATPNS is able to generate only T0L with regular control and $(R : Y)AL$, where $Y \in \{R, CF, CS\}$ we use inhibitor arcs to control the firing sequence. This section introduces Array Token Petri Net Structure with inhibitor arcs and compares it with the other array languages and tabled array languages.

Firing Rules in ATPNS with inhibitor arcs is similar to the firing rules of ATPNS with the extra condition that any transition with inhibitor input can fire only if the inhibitor input does not have any array.

Definition 10. *An Array Token Petri Net Structure with at least one inhibitor arc is defined as Array Token Petri Nets with inhibitor arcs.*

The language generated by the Petri net is the set of all arrays which reach the final place.

Example 2. $\Sigma = \{x \bullet\}, S = \begin{matrix} x & \bullet \\ & x \end{matrix}, B_1 = (\bullet)_m, B_2 = (x)_m, B_3 = \begin{pmatrix} x \\ x \end{pmatrix}^{\frac{n}{2}} \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix}^{\frac{n}{2}},$
 $B_4 = (x)^{\frac{n}{2}}(\bullet)^{\frac{n}{2}}, F = \{p_2\}.$

To start with both t_2 and t_5 are enabled. The sequence $t_2t_3t_4$ can be fired any number of times. Once t_5 is fired the inhibitor input p_6 makes sure that the sequences t_7t_8 is also fired the same number of times. When p_6 is empty the transition t_9 fires to put the final array into p_2 .

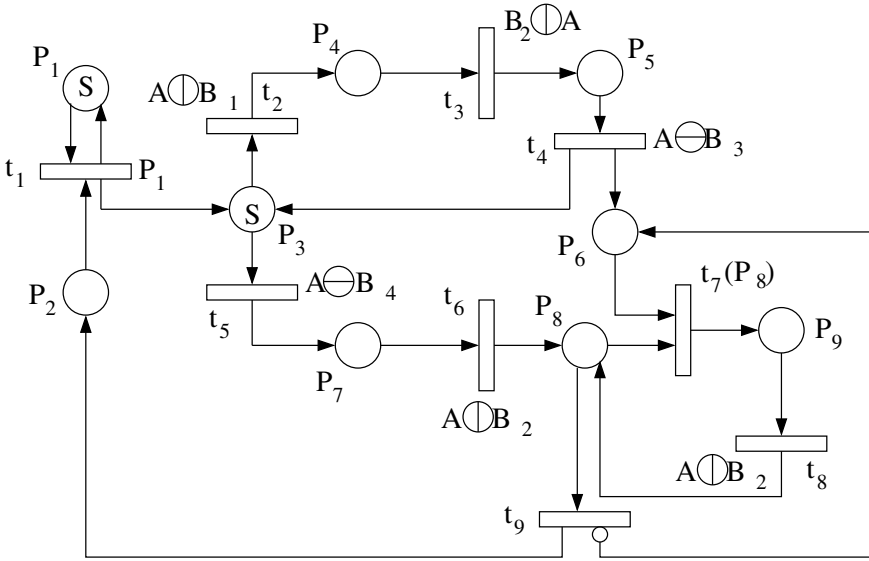


Fig. 8. ATPNS with inhibitor arc

The array generated by the firing sequence $t_1 \dots t_9$ is given below.

$$\begin{array}{c}
 x x \bullet \bullet x x \\
 x x \bullet \bullet x x \\
 S \xrightarrow{t_1 \dots t_9} x x \bullet \bullet x x \\
 x x \bullet \bullet x x \\
 x x \bullet \bullet x x \\
 x x \bullet \bullet x x \\
 x x \bullet \bullet x x
 \end{array}$$

The language generated is squares split into three equal columns.

Theorem 6. *The language generated by a table 0L array grammar with context-free or context-sensitive control can be generated by ATPNS with inhibitor arcs.*

Proof. Let $G = (V, \mathcal{P}, C, S)$ be a table 0L array grammar with context-free control, where V is the set of terminals, \mathcal{P} is a finite set of tables $\{P_1, P_2, \dots, P_k\}$, $C = (P_1 \dots P_i)^n (P_j \dots P_k)^n$, $1 \leq i, j \leq k$, be a context-free control and S is the start array.

Construct an ATPNS with two subnets C_1 and C_2 as in figure. Let p_1 belong to C_1 with the start array S as a token. Have transition t_1 with the catenation rule which corresponds to P_1 , p_1 being the input place and p_2 as its output place. Transition t_2 with the catenation rule which corresponds to P_2 , p_2 being the input place and p_3 as its output place and so on. Transition t_i with the catenation rule which corresponds to P_i , p_i being the input place and p_1, M_1 as its output places. The subnet C_1 can be executed any number of times. The sequence $(t_1 t_2 \dots t_i)^n$ would put n different arrays as tokens in M_1 . But the

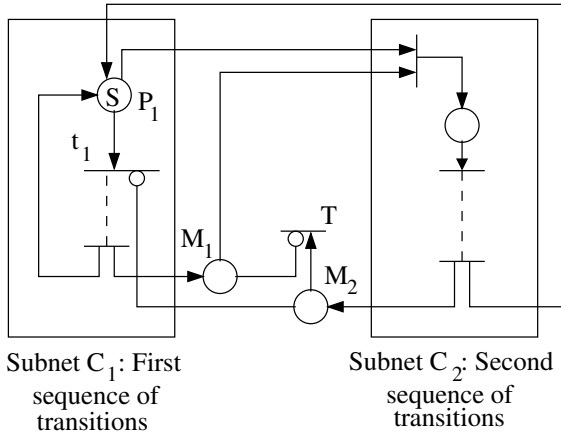


Fig. 9. Subnets of ATPNS with inhibitor arcs

place p_1 will have the array which is the array that would result in applying the tables $P_1 \dots P_i$ n times to S . Once t_j in C_2 is fired the second subnet starts its execution. Since M_1 is an input place for t_j , the subnet C_2 can be executed at the most n times (the number of times C_1 was executed). Similar to C_1 in C_2 there is a transition for every table P_j, \dots, P_k . Whenever C_2 is executed once an array is put in M_2 and p_1 . This array would be the array that results after applying the tables $(P_1 \dots P_i)^n (P_j \dots P_k)^m$ (m is the number of times C_2 was executed) to S . Once C_2 starts its execution C_1 cannot be executed again till M_2 is empty as M_2 is an inhibitor input for t_1 . After executing C_2 ' n ' times M_2 can be emptied by firing T ' n ' times. Since M_1 is an inhibitor input for T , T cannot be fired until M_1 is empty. In other words M_2 cannot be emptied until C_2 is executed exactly n times. Thus the subnets C_1 and C_2 are executed the same number of times. Hence the sequences $t_1 \dots t_i$ and $t_j \dots t_k$ can be fired exactly the same number of times. This is the effect of a context-free control.

Thus using the concepts of inhibitor arcs we are able to have a context-free control on the firing sequence. Similarly with three subnets and with proper usage of inhibitor inputs we can have a context-sensitive control on the firing sequence. □

Theorem 7. *The families of $(X : Y)AL$, where $X \in \{CF, CS\}$ and $Y \in \{R, CF, CS\}$, can be generated by ATPNS with inhibitor arcs.*

Proof. Let $G = (V, I, P, S)$ be an $(CF : Y)AG$. Then the nonterminal rules be of the form $(A)^n(B)^n$ or $\binom{A}{B}_n$ where A, B are intermediates. L_A and L_B the intermediate languages are regular, context-free or context-sensitive. Similar to the construction given in the proof of Theorem 6 have two subnets C_1 and C_2 . The subnet C_1 should generate the intermediate language L_A and the subnet C_2 should generate the intermediate language L_B . With the use inhibitor inputs we can make sure the subnets C_1 and C_2 are executed the same number of times. Thus with inhibitor arcs any $(CF : Y)AL$ can be generated.

For any $(CS : Y)AG$ the nonterminal rules are of the form $(A)^n(B)^n(C)^n$
 $(A)_n$
 or $(B)_n$, where A, B, C are intermediates. L_A, L_B and L_C the intermediate
 $(C)_n$
 languages are regular, context-free or context-sensitive. With three subnets and
 with proper usage of inhibitor inputs we can generate all $(X : Y)$ array languages,
 where $X \in \{CF, CS\}$ and $Y \in \{R, CF, CS\}$ can be generated by ATPNS with
 inhibitor arcs. \square

5 Conclusion

Array token Petri net structure generates rectangular arrays. This model is able to generate (R)P2DCFL, three of the nine families of array languages and the tabled 0L languages with regular control. Introducing inhibitor arcs to ATPNS the other six families of Array Languages and tabled 0L languages with context-free or context-sensitive control can also be generated. The languages generated by the nine families of array grammars and tabled 0L grammars with the three types of control can all be generated by ATPNS with inhibitor arcs.

References

1. Baker, H.G.: Petri net languages. Computation Structures Group Memo 124, Project MAC. MIT, Cambridge (1972)
2. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages 3, pp. 215–267. Springer (1997)
3. Hack, M.: Petri net languages. Computation Structures Group Memo 124, Project MAC. MIT (1975)
4. Peterson, J.L.: Petri Net Theory and Modeling of Systems. Prentice Hall, Inc., Englewood Cliffs (1981)
5. Lalitha, D., Rangarajan, K.: Column and row catenation petri net systems. In: Proceedings of the Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, pp. 1382–1387 (2010)
6. Lalitha, D., Rangarajan, K.: An application of array token Petri nets to clustering analysis for syntactic patterns. International Journal of Computer Applications 42(16), 21–25 (2012)
7. Rosenfeld, A., Siromoney, R.: Picture languages - A survey. Languages of Design 1(3), 229–245 (1993)
8. Shirley Gloria, D.K., Immanuel, B., Rangarajan, K.: Parallel context-free string-token Petri nets. International Journal of Pure and Applied Mathematics 59(3), 275–289 (2010)
9. Siromoney, R.: Advances in Array Languages. In: Ehrig, H., Nagl, M., Rosenfeld, A., Rozenberg, G. (eds.) Graph Grammars 1986. LNCS, vol. 291, pp. 549–563. Springer, Heidelberg (1987)
10. Siromoney, R., Siromoney, G.: Extended controlled table L -arrays. Information and Control 35, 119–138 (1977)

11. Siromoney, G., Siromoney, R., Kamala, K.: Picture languages with array rewriting rules. *Information and Control* 22, 447–470 (1973)
12. Siromoney, G., Siromoney, R., Kamala, K.: Array grammars and kolam. *Computer Graphics and Image Processing* 3(1), 63–82 (1974)
13. Subramanian, K.G., Rosihan, M., Ali, G.M., Nagar, A.K.: Pure 2D picture grammars and languages. *Discrete Applied Mathematics* 157(16), 3401–3411 (2009)
14. Wang, P.S.P.: An application of array grammars to clustering analysis for syntactic patterns. *Pattern Recognition* 17(4), 441–451 (1984)

Regional Hexagonal Tile Rewriting Grammars

Thangasamy Kamaraj¹ and Durairaj Gnanaraj Thomas²

¹ Department of Mathematics, Sathyabama University
Chennai - 119, India

kamaraj_mx@yahoo.co.in

² Department of Mathematics, Madras Christian College
Tambaram, Chennai - 600 059, India

dgthomasmcc@yahoo.com

Abstract. Regional tile rewriting grammars are recently introduced tiling based array rewriting grammar models on picture processing. They extend context free string grammars to rectangular array grammars with polynomial time complexity. We introduce Regional Hexagonal Tile Rewriting Grammar (RHTRG) and generate images. We have compared this model with Context Free Hexagonal Array Grammars (CFHAG) and Hexagonal Tiling Systems (HTS) and shown that RHTRG have higher generative capacity than CFHAG and incomparable with HTS and they are strictly included in the Hexagonal Tile Rewriting Grammars.

Keywords: Hexagonal picture languages, Regional tiling, Hexagonal array grammars, Hexagonal tiling systems.

1 Introduction

Picture languages generated by grammars have been advocated since seventies for problems arising in the framework of pattern recognition and image analysis. Regional tile rewriting grammars [1] are recently introduced tiling based array rewriting grammar models on picture processing. This model is resulting from the restriction of Tile Rewriting Grammars (TRG) [2]. This model extends the context free string grammars to two dimensions with polynomial-time complexity. The other kinds of 2D array rewriting grammars can be defined by restricting the tiles used in RTG. RTG's generative capacity is stronger than that of Siromoney's Array grammar [7] and context free Prussa Grammar [5].

Hexagonal patterns and grammars are known to occur in the literature on picture processing and scene analysis [4,6]. One of the classical formal models of generation of hexagonal arrays is Hexagonal Kolam Array Grammars (HKAG) [6], in which hexagonal arrays on triangular grids were considered as two dimensional representation of three dimensional blocks and several transformations related to scene analysis were studied. The notion of arrowhead catenation is a novel feature of that study. Recently, Hexagonal Tile Rewriting Grammars (HTRG) are introduced [9], which combined the features of Hexagonal Tiling System (HTS) [3] and Hexagonal Array Rewriting System, HAG [8].

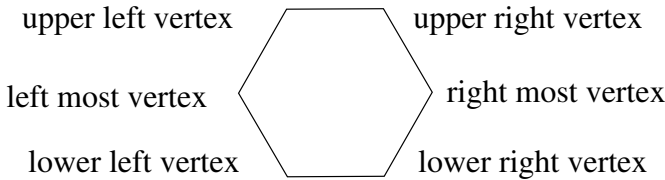
Motivated by the above studies, in this paper, we introduce Regional Hexagonal Tile Rewriting Grammars (RHTRG) and generate images. They are the restriction of HTRG with a specified set of simple type of tiling named regional. A typical rule of this grammar has a left and right part, both hexagonal pictures of unspecified but equal size (isometric). The left part is a C -homogeneous hexagonal picture (i.e.,) a hexagonal picture containing some non terminal symbol C for all its pixels. The right part is a picture of regional hexagonal local language over non-terminal symbols. Thus a grammar rule is a scheme defining a possibly unbounded number of isometric pairs of left and right hexagonal pictures. But we can also have the rules whose right part is single terminal.

We compare the class RHTRG with context free Hexagonal Array Grammars (CFHAG) [8], which are the extension of HKAG [6]. Then we compare it with HTS. We prove that RHTRG has more generative capacity than CFHAG and it is incomparable with HTS. We show that the family of languages generated by RHTRGs is included in the family of languages generated by HTRGs.

2 Preliminaries

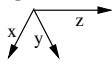
Let Σ be a finite alphabet of symbols. A hexagonal picture p over Σ is a hexagonal array of symbols of Σ .

We consider hexagons of the type



For example, a hexagonal picture over the alphabet $\{a\}$ is $\begin{matrix} & a & a \\ a & a & a \\ & a & a \end{matrix}$.

The set of all hexagonal arrays over of the alphabet Σ is denoted by Σ^{**H} . A hexagonal picture language L over Σ is a subset of Σ^{**H} .

With respect to a triad  of triangular axes x, y, z , the co-ordinates of each element of hexagonal picture can be fixed [3].

For $p \in \Sigma^{**H}$, let \hat{p} be the hexagonal array obtained by surrounding p with a special boundary symbol $\# \notin \Sigma$. For example,

$$\hat{p} = \begin{matrix} \# & \# & \# \\ \# & a & a & \# \\ \# & a & a & a & \# \\ \# & a & a & \# \\ \# & \# & \# \end{matrix} \quad \text{for } p = \begin{matrix} & a & a \\ a & a & a \\ & a & a \end{matrix}$$

Given a picture $p \in \Sigma^{**H}$, let ' ℓ ' denote the number of elements in the border of p from upper left vertex to left most vertex in the direction \swarrow called x

direction, ‘ m ’ denote the number of elements in the border of p from upper right vertex to right most vertex in the direction \searrow called y direction and ‘ n ’ denote the number of elements in the border of p from upper left vertex to upper right vertex in the direction \rightarrow called z direction.

The directions are fixed with origin of reference as the upper left vertex, having co-ordinates $(1, 1, 1)$. The triple (ℓ, m, n) is called the size of the picture p denoted by $|p| = (\ell, m, n)$. Let p_{ijk} denote the symbol in p with co-ordinates (i, j, k) where $1 \leq i \leq \ell, 1 \leq j \leq m, 1 \leq k \leq n$. Let $\Sigma^{(\ell, m, n)H}$ be the set of hexagonal pictures of size (ℓ, m, n) .

Given a hexagonal picture p of size (ℓ, m, n) , we denote by $B_{g,h,k}(p)$ the set of all hexagonal subpictures of p of size (g, h, k) , where $g \leq \ell, h \leq m$ and $k \leq n$. Each member of $B_{2,2,2}(p)$ is called a hexagonal tile. We denote the set of all hexagonal tiles contained in a picture \hat{p} by $[[\hat{p}]]$.

We now recall some basic definitions [3,6].

Definition 1. A p -hexagon is a six-sided convex polygon whose opposite sides are parallel. A b -hexagon is an equiangular p -hexagon whose opposite sides are equal.

Definition 2. A non-convex hexagon $ABCDEF$ as shown in Figure 1 is called an a -hexagon or an arrowhead if $|AB| = |FE|, |BC| = |ED|, AB$ is parallel to FE and BC is parallel to ED . It is noted that the opposite sides $|CD|$ and $|AF|$ are equal and parallel. $|CD|$ is the thickness of the arrowhead and B is called the vertex. ABC is the outermost edge and DEF is the innermost edge.

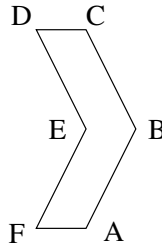


Fig. 1. Right arrowhead

Depending on the direction of a vertex an arrowhead is classified as right arrowhead (R), upper left arrowhead (UL), lower left arrowhead (LL) and their duals left (L), upper right (UR) and lower right (LR) arrowheads.

Definition 3. If $PQRSTU$ (Figure 2 (a)) is a hexagon and $ABCDEF$ is a right arrowhead with B as the vertex, then the arrowhead can be concatenated to the hexagon in the right direction \rightarrow if $|PQ| = |FE|$ and $|QR| = |ED|$ where PQ and QR are the outermost right edges of the hexagon $PQRSTU$.

The resultant is a hexagon $ABCSTU$ as shown in Figure 2 (b).

It is noted that $|SC| = |SR| + |RC| = |SR| + |DC|$ and $|UA| = |UP| + |PA| = |UP| + |FA|$.

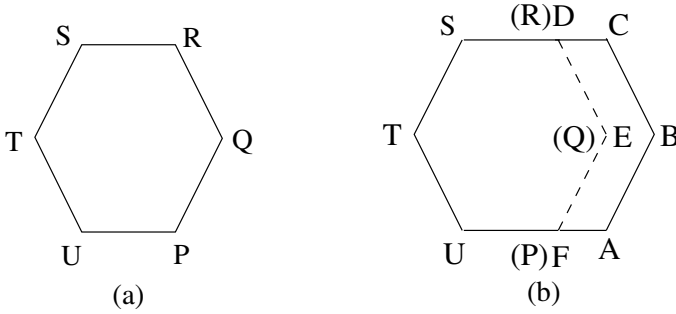


Fig. 2. Right arrowhead catenated to hexagon

The upper left and the lower left arrowheads and the corresponding catenations in the upper left direction \curvearrowleft and the lower left direction \curvearrowdown are illustrated in Figure 3.

Similarly the duals the left, upper right and lower right arrowheads and the corresponding catenations in the left direction \curvearrowright , the upper right direction \curvearrowup and the lower right direction \curvearrowdown can be defined.

Definition 4. Let Γ and Σ be two finite alphabets and $\pi : \Gamma \rightarrow \Sigma$ be a mapping which we call, a projection. Let $p \in \Gamma^{**H}$ be a hexagonal picture. The projection by mapping π of p is the picture $p' \in \Sigma^{**H}$ such that $p'(i, j, k) = \pi(p(i, j, k))$ for all $1 \leq i \leq \ell, 1 \leq j \leq m$ and $1 \leq k \leq n$, where (ℓ, m, n) is the size of the hexagonal picture.

Definition 5. Let $L \subset \Gamma^{**H}$ be a hexagonal picture language. The projection by mapping π of L is the language $L' = \{p'/p' = \pi(p), \forall p \in L\} \subseteq \Sigma^{**H}$.

Definition 6. Let Γ be a finite alphabet. A hexagonal picture language $L \subseteq \Gamma^{**H}$ is called local if there exists a finite set Δ of hexagonal tiles over $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**H} / B_{2,2,2}(\hat{p}) \subseteq \Delta\}$. In this case L is denoted by $L(\Delta)$. The family of hexagonal local picture languages will be denoted by $\mathcal{L}(HLOC)$.

Definition 7. Let Σ be a finite alphabet. A hexagonal picture language $L \subseteq \Sigma^{**H}$ is called recognizable if there exists a hexagonal local picture language L' (given by a set of hexagonal tiles) over an alphabet Γ and a mapping $\pi : \Gamma \rightarrow \Sigma$ such that $L = \pi(L')$.

The family of all recognizable hexagonal picture languages will be denoted by $\mathcal{L}(HREC)$.

Example 1. Let $\Sigma = \{a\}$. It is shown that the language of hexagonal pictures over Σ with all sides of equal length is not local, but recognizable [3].

Definition 8. A Hexagonal Tiling System (HTS) T is a 4-tuple $(\Sigma, \Gamma, \pi, \theta)$, where Σ and Γ are two finite sets of symbols, $\pi : \Gamma \rightarrow \Sigma$ is a projection and θ is a set of hexagonal tiles over the alphabet $\Gamma \cup \{\#\}$.

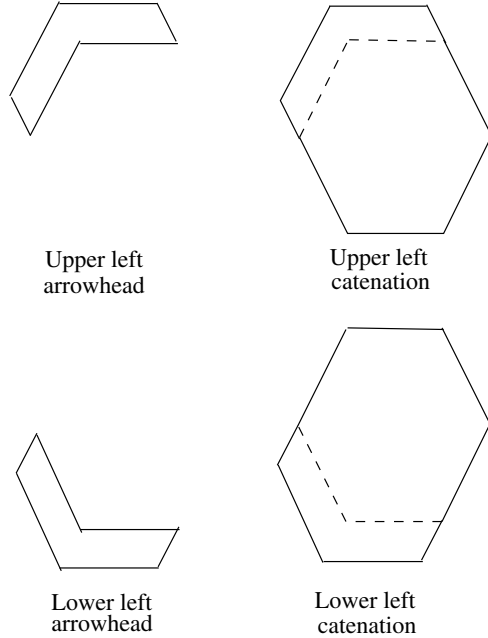


Fig. 3. Upper left and lower left arrowheads catenated to hexagons

Definition 9. The hexagonal picture language $L \subseteq \Sigma^{**H}$ is tiling recognizable if there exists a tiling system $T = (\Sigma, \Gamma, \pi, \theta)$ such that $L = \pi(L(\theta))$.

It is easy to see that $\mathcal{L}(HREC)$ is exactly the family of hexagonal picture languages recognized by HTS, i.e., $\mathcal{L}(HTS)$.

We now recall the notions of regular (R), context-free (CF) and context-sensitive (CS) hexagonal array grammars and languages [8].

Definition 10. A Hexagonal Array Grammar (HAG) is $G = (N, I, T, P, S, \mathcal{L})$ where N, I and T are finite non-empty sets of nonterminals, intermediates and terminals respectively. $S \in N$ is the start symbol. For each A in I , L_A is an intermediate language which is regular, CF or CS string language, written in the appropriate arrowhead form. An arrowhead is written in the form $\{\dots \langle v \rangle \dots\}$ where $\langle v \rangle$ denotes the vertex and the arrowhead is written in the clock-wise direction. $\mathcal{L} = \{L_A/A \in I\}$.

$P = P_1 \cup P_2$ is a finite nonempty set of productions where P_1 consists of initial rules of the following forms:

- (1) $S \rightarrow H \begin{matrix} \curvearrowright \\ \searrow \end{matrix} S'$ (2) $S \rightarrow H \begin{matrix} \curvearrowleft \\ \swarrow \end{matrix} S'$ (3) $S \rightarrow H \begin{matrix} \curvearrowup \\ \nearrow \end{matrix} S'$
 (4) $S \rightarrow H \begin{matrix} \curvearrowdown \\ \swarrow \end{matrix} S'$ (5) $S \rightarrow H \begin{matrix} \curvearrowright \\ \leftarrow \end{matrix} S'$ (6) $S \rightarrow H \begin{matrix} \curvearrowleft \\ \rightarrow \end{matrix} S'$

where $S' \in N$; $S' \neq S$ and H is a hexagonal array over T .

G is regular if the rules of P_2 are of the forms:

- (1) $S_1 \rightarrow A \begin{matrix} \curvearrowdown \\ \searrow \end{matrix} S_2$ (2) $S_1 \rightarrow A \begin{matrix} \curvearrowleft \\ \swarrow \end{matrix} S_2$ (3) $S_1 \rightarrow A \begin{matrix} \curvearrowup \\ \nearrow \end{matrix} S_2$

$$(4) S_1 \rightarrow A \curvearrowright S_2 \quad (5) S_1 \rightarrow A \curvearrowleft S_2 \quad (6) S_1 \rightarrow A \curvearrowright S_2$$

$$(7) S_1 \rightarrow A$$

where $S_1, S_2 \in N$; $S_1, S_2 \neq S$ and $A \in I$. Furthermore, if an initial rule in P_1 is of the form (r) , $r = 1, \dots, 6$, then P_2 does not contain any rule of the form $(r + 1)$ if r is odd and $(r - 1)$, if r is even. Also P_1 and P_2 do not contain rules of both the forms (r) and $(r + 1)$, $r = 1, 3, 5$.

G is CF if the rules of P_2 are of the form

$$S_1 \rightarrow \alpha_1 \circledast_1 \cdots \circledast_{k-1} \alpha_k (k \geq 1), \text{ where } S_1 \in N; S_1 \neq S \text{ and } \alpha_i \in (N - \{S\}) \cup I$$

$(1 \leq i \leq k)$ and \circledast_j denotes any one of the six arrowhead catenations, $(1 \leq j \leq k - 1)$.

G is CS if the rules of P_2 are of the form

$$\beta \circledast S_1 \circledast \delta \rightarrow \beta \circledast \gamma \circledast \delta \text{ or } \beta \circledast S_1 \rightarrow \beta \circledast \gamma \text{ or } S_1 \circledast \delta \rightarrow \gamma \circledast \delta \text{ or } S_1 \rightarrow \gamma, \text{ where}$$

$S_1 \in N; S_1 \neq S$ and β, γ, δ are of the form $\alpha_1 \circledast \cdots \circledast \alpha_k$, with $\alpha_i \in (N - \{S\}) \cup I$, $1 \leq i \leq k$.

In particular, G is called $(X : R)HAG$ or $(X : CF)HAG$ or $(X : CS)HAG$, for $X \in \{R, CF, CS\}$, according as all the intermediate languages are regular or at least one of them is CF or at least one of them is CS.

3 Regional Hexagonal Tile Rewriting Grammars

In this section we are going to introduce and study the grammar with a specified set of tiling named regional. The definitions given below are analogous to that of [1] for the case of hexagonal pictures. Here a picture p refers to a b -hexagonal (or) a -hexagonal (Arrowhead) array.

Definition 11. A pixel is an element $p(i, j, k)$ of a picture p . If all pixels are identical to $C \in \Sigma$ the picture is called C -homogeneous or C -picture. The domain of a picture p of size (ℓ, m, n) is set $d(p)$ consists of all positions of pixels in p . A subdomain $d_s(p)$ of $d(p)$ is a set of positions of pixels in p correspond to a sub picture s of size (g, h, k) if $1 \leq g \leq \ell, 1 \leq h \leq m, 1 \leq k \leq n$. A subdomain is called C -homogeneous when its associated sub picture is a C -picture. C is called the label of the subdomain.

Definition 12. A homogeneous partition of a picture p is any partition $P = \{d_{s_1}, d_{s_2}, \dots, d_{s_n}\}$ of $d(p)$ into homogeneous sub domains $d_{s_1}, d_{s_2}, \dots, d_{s_n}$. The unit partition of p , written $unit(p)$, is the homogeneous partition of $d(p)$ defined by single pixels. A homogeneous partition is called strong if adjacent subdomains have different labels. The unique partition given by strong homogeneous partition of a picture p is $\pi^*(p)$.

Definition 13. A Hexagonal Tile Rewriting Grammar [9] is a tuple (Σ, N, S, R) , where Σ is the terminal alphabet, N is a set of non terminal symbols, $S \in N$ is the starting symbol, R is a set of rules of two kinds of forms:

Fixed size : $A \rightarrow t$, where $A \in N, t \in \Sigma$.

Variable size : $A \rightarrow w$, where $A \in N, w$ is a set of hexagonal tiles over $N \cup \{\#\}$ such that $HLOC(w)$ admits a strong homogeneous partition for each $p \in HLOC(w)$.

Definition 14. A homogeneous partition is regional (RH) if and only if distinct subdomains have distinct labels. A picture p is regional if it admits a RH partition. A language is regional if all its pictures are regional.

Definition 15. A regional hexagonal tile rewriting grammar (RHTRG) is a tuple (Σ, N, S, R) , where Σ is the terminal alphabet, N is a set of non terminal symbols, $S \in N$ is the starting symbol, R is a set of rules of the form

Fixed size : $A \rightarrow t$, where $t \in \Sigma$.

Variable size : $A \rightarrow w$, where w is a set of hexagonal tiles over $N \cup \{\#\}$, $HLOC(w)$ is a regional language.

Picture derivation is defined as follows.

Consider a hexagonal grammar $G = \langle \Sigma, N, S, R \rangle$. Let $p, p' \in (\Sigma \cup N)^{(g,h,k)}$ be pictures of identical size. Let π, π' be the homogeneous partitions of $d(p)$ with $\pi = \{d_{s_1}, d_{s_2}, \dots, d_{s_n}\}$. We say that (p', π') is derived in one step from (p, π) , written as $(p, \pi) \xrightarrow[G]{\Rightarrow} (p', \pi')$ if and only if for some $A \in N$ and for some rule $r \in R$ with left part A , there exists a subdomain d_{s_i} in π called application area, such that

- (i) p' is obtained by substituting the sub picture s at d_s in p with a picture q of same size, defined as follows:
 - (a) if r is of type(1), then $q = t$
 - (b) if r is of type(2), then $q \in HLOC(w)$.
- (ii) π' is a homogeneous partition of $d(p)$ into the subdomains

$$(\pi \setminus \{d_{s_i}\}) \cup trans_{d_s} \pi^*(q)$$

where $trans_{d_s} \pi^*(q)$ denotes the displacement of strong homogeneous partition of q to the position of d_s .

We say that (p', π') is derived from (p, π) in n steps, written $(p, \pi) \xrightarrow[G]{\Rightarrow^n} (p', \pi')$ if and only if $p = p'$ and $\pi = \pi'$ when $n = 0$ or there is a picture q and a homogeneous partition π'' such that $(p, \pi) \xrightarrow[G]{\Rightarrow^{n-1}} (q, \pi'')$ and $(q, \pi'') \xrightarrow[G]{\Rightarrow} (p', \pi')$. $\xrightarrow[G]{\Rightarrow^*}$ denotes the transitive closure of $\xrightarrow[G]{\Rightarrow}$.

Definition 16. The hexagonal picture language $L(G)$ defined by a RHTRG G is the set of $p \in \Sigma^{++H}$ such that

$$\langle S|p|, d(p) \rangle_{\xrightarrow[G]{\Rightarrow^*}} (p, U)$$

where U denotes the partition of $d(p)$ defined by single pixels. Shortly we can also write $S \xrightarrow[G]{\Rightarrow^*} p$.

The family of hexagonal languages generated by RHTRGs is denoted by $\mathcal{L}(RHTRG)$.

4 Examples on Regional Hexagonal Tile Rewriting Grammar

In this section we give some examples of RHTRGs and their languages.

Example 2. The set of all b -hexagonal pictures whose diagonal elements are x 's and other elements are y 's is generated by the RHTRG grammar $G = (\Sigma, N, S, R)$ where $\Sigma = \{x, y\}$, $N = \{X, Y, Z, X', X'', Y', Y'', Z', A', A'', B', B'', A, B, Y, U_1, U_2, \dots, U_7\}$ and R consists of the following rules:

$$S \rightarrow \left[\left[\begin{array}{cccccccc} & & \# & \# & \# & \# & \# & \\ & & \# & X & Y & Y & Y & \# \\ & & \# & X & S & S & S & Y & \# \\ & & \# & X & S & S & S & S & Y & \# \\ \# & X & S & S & S & S & S & S & Z & \# \\ \# & X & S & S & S & S & S & Z & \# \\ & \# & X & S & S & S & Z & \# \\ & & \# & X & Z & Z & Z & \# \\ & & & \# & \# & \# & \# & \end{array} \right] \right]$$

$$/ \left[\left[\begin{array}{cccccccc} & & \# & \# & \# & \# & \\ & & \# & X & Y & Y & \# \\ & & \# & X & S & S & Y & \# \\ \# & X & S & S & S & Z & \# \\ \# & X & S & S & Z & \# \\ & \# & X & Z & Z & \# \\ & & \# & \# & \# & \# \end{array} \right] \right] / \left[\left[\begin{array}{cccc} \# & \# & \# & \\ \# & U_1 & U_2 & \# \\ \# & U_3 & U_7 & U_4 & \# \\ \# & U_5 & U_6 & \# \\ \# & \# & \# & \end{array} \right] \right]$$

$$X \rightarrow \left[\left[\begin{array}{cccc} & \# & \# & \# \\ & \# & X' & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & X'' & \# \\ \# & \# & \# & \# \end{array} \right] \right]; A \rightarrow \left[\left[\begin{array}{cccc} & \# & \# & \# \\ & \# & A' & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & A & \# \\ \# & \# & A'' & \# \\ \# & \# & \# & \# \end{array} \right] \right]$$

$$Y \rightarrow \left[\left[\begin{array}{cccccccc} & & \# & \# & \# & \# & \# & \\ & & \# & Y' & Y & Y & Y & \# \\ \# & \# & \# & \# & \# & \# & Y & \# \\ & & & & & & \# & Y & \# \\ & & & & & & \# & Y'' & \# \\ & & & & & & \# & \# & \\ & & & & & & \# & \# & \end{array} \right] \right]$$

$$\begin{array}{l}
 Z \rightarrow \left[\left[\begin{array}{cccccccc} & & & & & & & \# \\ & & & & & & \# & \# \\ & & & & & \# & Z' & \# \\ & & & & \# & B & \# \\ \# & \# & \# & \# & \# & B & \# \\ & \# & B & B & B & B & \# \\ & & \# & \# & \# & \# & \# \end{array} \right] \right] \\
 \\
 B \rightarrow \left[\left[\begin{array}{cccccccc} & & & & & & & \# \\ & & & & & & \# & \# \\ & & & & & \# & B' & \# \\ & & & & \# & B & \# \\ \# & \# & \# & \# & \# & B & \# \\ & \# & B'' & B & B & B & \# \\ & & \# & \# & \# & \# & \# \end{array} \right] \right]
 \end{array}$$

$X', X'', Y, Z', A, B, U_1, U_2, \dots, U_7 \rightarrow x$

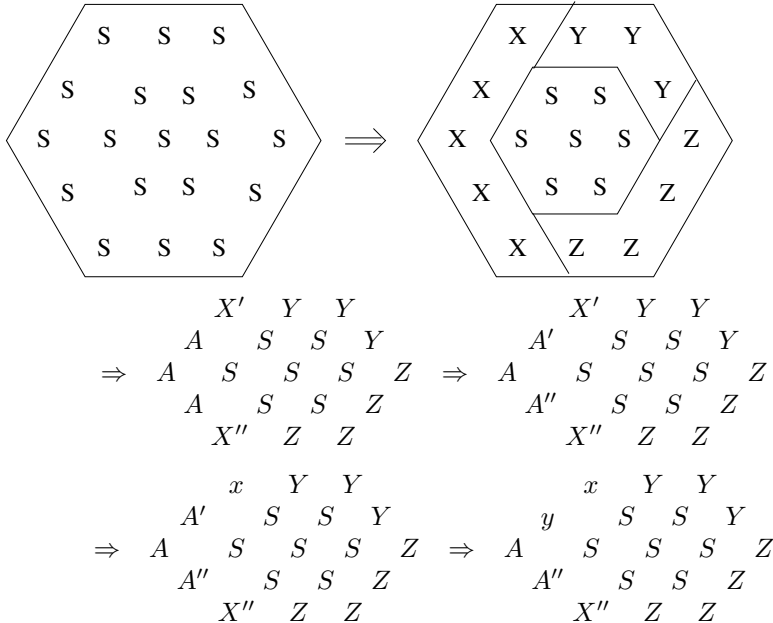
$A', A'', Y', Y'', B', B'' \rightarrow y$

First three pictures generated by this grammar are

```

          x y y x
        x y x   y x y x y
      x x   y x x y   y y x x y y
    x x x x x x x x x x x x x x
      x x   y x x y   y y x x y y
          x y x       y x y x y
                  x y y x
    
```

The derivation of second picture is given in the following figure.



$$\begin{array}{ccc}
 & x & Y & Y \\
 \Rightarrow & y & S & S & Y & Z \\
 & x & S & S & S & Z \\
 & A'' & S & S & Z \\
 & & X'' & Z & Z \\
 \\
 & x & Y & Y \\
 \Rightarrow & y & S & S & Y & Z \\
 & x & S & S & S & Z \\
 & y & S & S & Z \\
 & x & Z & Z \\
 \\
 & x & Y & Y \\
 \Rightarrow & y & U_1 & U_2 & y \\
 & x & U_3 & U_7 & U_4 & x \\
 & y & U_5 & U_6 & y \\
 & x & y & x \\
 \\
 & x & y & x \\
 \Rightarrow & y & x & x & y \\
 & x & x & x & x & x \\
 & y & x & x & y \\
 & x & y & x
 \end{array}$$

Example 3. The language

$$L = \left\{ \begin{array}{cccccccccccc} 1 & 1 & & 1 & 1 & 1 & & 1 & 1 & 1 & 1 & & 1 & 1 & 1 & 1 & 1 & \\ 1 & 2 & 1, & 1 & 2 & 2 & 1, & 1 & 2 & 2 & 2 & 1, & 1 & 2 & 2 & 2 & 2 & 1, & \dots \\ 1 & 1 & & 1 & 1 & 1 & & 1 & 1 & 1 & 1 & & 1 & 1 & 1 & 1 & 1 & \end{array} \right\}$$

is generated by the RHTRG grammar $G = \langle \Sigma, N, S, R \rangle$ where $\Sigma = \{1, 2\}$, $N = \{S, X, X', Y, Y', Z, Z', U, V\}$.

R consists of following rules

$$\begin{aligned}
 S &\rightarrow \left[\left[\begin{array}{cccccc} \# & \# & \# & \# & & \\ \# & X & X & X & \# & \\ \# & U & Z & Z & V & \# \\ \# & Y & Y & Y & \# & \\ \# & \# & \# & \# & & \end{array} \right] \right] / \left[\left[\begin{array}{cccccc} \# & \# & \# & & & \\ \# & X & X & \# & & \\ \# & U & Z & V & \# & \\ \# & Y & Y & \# & & \\ \# & \# & \# & & & \end{array} \right] \right] \\
 X &\rightarrow \left[\left[\begin{array}{cccccc} \# & \# & \# & \# & & \\ \# & X' & X & X & \# & \\ \# & \# & \# & \# & & \end{array} \right] \right] \\
 Y &\rightarrow \left[\left[\begin{array}{cccccc} \# & \# & \# & \# & & \\ \# & Y' & Y & Y & \# & \\ \# & \# & \# & \# & & \end{array} \right] \right] \\
 Z &\rightarrow \left[\left[\begin{array}{cccccc} \# & \# & \# & \# & & \\ \# & Z' & Z & Z & \# & \\ \# & \# & \# & \# & & \end{array} \right] \right]
 \end{aligned}$$

$$\begin{aligned}
 X' &\rightarrow 1, X \rightarrow 1, U \rightarrow 1, \\
 Y' &\rightarrow 1, Y \rightarrow 1, V \rightarrow 1, \\
 Z' &\rightarrow 2, Z \rightarrow 2.
 \end{aligned}$$

5 Comparison Results

In this section we give the main results of this paper.

Theorem 1. *The family of CF Hexagonal Array Grammar languages is strictly included in the family of RHTRG languages.*

Proof. Consider CFHAG grammar G in Chomsky Normal form. It may contain six types of rules $A \overset{\leftarrow}{\circlearrowright} B$, $A \overset{\rightarrow}{\circlearrowright} B$, $A \overset{\downarrow}{\circlearrowright} B$ and its duals $A \overset{\rightarrow}{\circlearrowleft} B$, $A \overset{\downarrow}{\circlearrowleft} B$, $A \overset{\leftarrow}{\circlearrowleft} B$. Then the rules of G , $C \rightarrow A \overset{\leftarrow}{\circlearrowright} B$ are equivalent to RHTRG rules of the form

$$C \rightarrow \left[\begin{array}{cccccc} \# & \# & \# & \# & \# & \\ \# & B & B & A & A & \# \\ \# & B & B & A & A & A \\ \# & B & B & A & A & \# \\ \# & \# & \# & \# & \# & \end{array} \right]$$

$C \rightarrow A \overset{\rightarrow}{\circlearrowright} B$ equivalent to RHTRG rules of the form

$$C \rightarrow \left[\begin{array}{cccccc} & & \# & \# & \# & \\ & & \# & B & B & \# \\ & \# & B & B & B & \# \\ \# & \# & A & A & A & B \\ \# & A & A & A & A & \# \\ \# & A & A & A & \# & \\ \# & \# & \# & \# & \# & \end{array} \right]$$

$C \rightarrow A \overset{\downarrow}{\circlearrowright} B$ equivalent to RHTRG rules of the form

$$C \rightarrow \left[\begin{array}{cccccc} & & \# & \# & \# & \\ & \# & A & A & \# & \\ \# & \# & A & A & A & \# \\ \# & \# & A & A & B & \# \\ & \# & B & B & B & \# \\ & & \# & B & B & \# \\ & & \# & \# & \# & \end{array} \right]$$

The rules $C \rightarrow t$, $t \in \Sigma$ are identical in both the grammars. The inclusion is strict because the language of Example 3 cannot be generated by any CFHAG.

Suppose there exists a CFHAG grammar $G = (N, I, T, P, S, L_{A_1})$ where $N = \{S, S_1\}$, $I = \{A_1\}$, $T = \{1, 2\}$, $P = P_1 \cup P_2$ where $P_1 = \{S \rightarrow H \overset{\leftarrow}{\circlearrowright} S_1\}$ and $P_2 = \{S_1 \rightarrow A_1 \overset{\leftarrow}{\circlearrowright} S_1, S_1 \rightarrow A_1\}$ with $H = \begin{Bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1 \end{Bmatrix}$ and L_{A_1} is an intermediate language corresponding to A_1 .

The other elements of the language of HAG are constructed through the left arrow head catenation of first hexagonal array and hence first element of the language must be the suffix hexagonal array of the second element of the language and so on, but which is not the language as in Example 3. Similarly the proof can be made if we use the right arrow head catenation in the grammar. \square

Theorem 2. *The family of hexagonal tiling system languages and the family of RHTRG languages are incomparable.*

Proof. First we prove that both the families of languages are not disjoint. The language

$$L_1 = \left\{ \begin{array}{cccc} x & x & & x & x & x \\ x & x & x & , & x & x & x & x & , & \dots \\ x & x & & & x & x & x \end{array} \right\}$$

is present in both the families.

If we consider the hexagonal tile set

$$\theta = \left[\left[\begin{array}{cccc} \# & \# & \# & \# \\ \# & 1 & 1 & 1 & \# \\ \# & 2 & 2 & 2 & 2 & \# \\ \# & 3 & 3 & 3 & \# \\ \# & \# & \# & \# \end{array} \right] \right]$$

Then

$$L_2 = \left\{ \begin{array}{cccc} 1 & 1 & & 1 & 1 & 1 \\ 2 & 2 & 2 & , & 2 & 2 & 2 & 2 & , & \dots \\ 3 & 3 & & & 3 & 3 & 3 \end{array} \right\} \\ = L(\theta).$$

This implies $L_2 \in \mathcal{L}(HLOC)$. If we use a projection π as $\pi(1) = \pi(2) = \pi(3) = x$ then we get $L_1 = \pi[L_2]$.

Thus the tiling system $\langle \Sigma, \Gamma, \theta, \pi \rangle$ where $\Sigma = \{x\}$, $\Gamma = \{1, 2, 3\}$, accepts the language L_1 . Hence L_1 is HTS recognizable.

Consider the RHTRG grammar, $G = \langle \Sigma, N, S, R \rangle$ where $\Sigma = \{x\}$ and $N = \{S, X, Y, Z, X', Y', Z'\}$

R consists of the rules

$$S \rightarrow \left[\left[\begin{array}{cccc} \# & \# & \# & \# \\ \# & X & X & X & \# \\ \# & Y & Y & Y & Y & \# \\ \# & Z & Z & Z & \# \\ \# & \# & \# & \# \end{array} \right] \right]; \quad X \rightarrow \left[\left[\begin{array}{cccc} \# & \# & \# & \# \\ \# & X' & X & X & \# \\ \# & \# & \# & \# \end{array} \right] \right]$$

$$Y \rightarrow \left[\left[\begin{array}{cccc} \# & \# & \# & \# \\ \# & Y' & Y & Y & \# \\ \# & \# & \# & \# \end{array} \right] \right]; \quad Z \rightarrow \left[\left[\begin{array}{cccc} \# & \# & \# & \# \\ \# & Z' & Z & Z & \# \\ \# & \# & \# & \# \end{array} \right] \right]$$

$X', X, Y', Y, Z', Z \rightarrow x$.

Clearly this RHTRG grammar generates all the elements of L_1 .

Thus $\mathcal{L}(HTS) \cap \mathcal{L}(RHTRG) \neq \phi$.

The language L_3 of hexagonal pictures of a 's with all the three sides equal is in HTS, $T = \langle \Sigma, \Gamma, \theta, \pi \rangle$ where $\Sigma = \{a\}$, $\Gamma = \{0, 1\}$,

$$\theta = \left[\left[\begin{array}{cccccccc} \# & \# & \# & \# & \# & \# & & \\ & \# & 1 & 0 & 0 & 0 & 0 & \# \\ & & \# & 0 & 0 & 1 & 0 & 0 & 0 & \# \\ & & & \# & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \# \\ & & & & \# & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \# \\ & & & & & \# & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \# \\ & & & & & & \# & 0 & 0 & 0 & 0 & 1 & 0 & \# \\ & & & & & & & \# & 0 & 1 & 0 & 0 & 1 & 0 & \# \\ & & & & & & & & \# & 0 & 0 & 1 & 0 & 0 & \# \\ & & & & & & & & & \# & 1 & 0 & 0 & 0 & \# \\ & & & & & & & & & & \# & \# & \# & \# & \# & \# \end{array} \right] \right]$$

$\pi(1) = \pi(0) = a$. But the only possible grammar rules generating strong homogeneous partition are

$$S \rightarrow \left[\left[\begin{array}{cccccccc} \# & \# & \# & \# & \# & \# & \# & \\ & \# & W_1 & B_0 & W_0 & B_0 & W_0 & \# \\ & & \# & B_0 & B_0 & W_1 & B_0 & W_0 & B_0 & \# \\ & & & \# & W_0 & W_1 & W_0 & B_0 & W_1 & B_0 & W_0 & \# \\ & & & & \# & B_0 & B_0 & B_0 & W_0 & B_0 & W_1 & B_0 & W_1 & B_0 & \# \\ & & & & & \# & B_0 & B_0 & W_1 & B_0 & W_0 & B_0 & W_1 & B_0 & W_1 & \# \\ & & & & & & \# & W_0 & W_1 & B_0 & W_1 & B_0 & W_0 & \# \\ & & & & & & & \# & B_0 & B_0 & W_1 & B_0 & W_0 & B_0 & \# \\ & & & & & & & & \# & W_1 & B_0 & W_0 & B_0 & W_0 & \# \\ & & & & & & & & & \# & \# & \# & \# & \# & \# \end{array} \right] \right]$$

$$S \rightarrow \left[\left[\begin{array}{cccccccc} \# & \# & \# & \# & \# & \# & \# & \\ & \# & B_1 & W_0 & B_0 & W_0 & B_0 & \# \\ & & \# & W_0 & W_0 & B_1 & W_0 & B_0 & W_0 & \# \\ & & & \# & B_0 & B_1 & B_0 & W_0 & B_1 & W_0 & B_0 & \# \\ & & & & \# & W_0 & W_0 & W_0 & B_0 & W_0 & B_1 & W_0 & \# \\ & & & & & \# & B_0 & B_0 & W_0 & B_0 & W_0 & B_1 & W_0 & B_1 & \# \\ & & & & & & \# & W_0 & W_0 & W_0 & B_0 & W_0 & B_1 & W_0 & \# \\ & & & & & & & \# & B_0 & B_1 & W_0 & B_1 & W_0 & B_0 & \# \\ & & & & & & & & \# & W_0 & W_0 & B_1 & W_0 & B_0 & \# \\ & & & & & & & & & \# & \# & \# & \# & \# & \# \end{array} \right] \right]$$

$W_1, B_1, W_0, B_0 \rightarrow a$.

But clearly the variable size rules are not generating regional pictures. So the language L_3 is not generated by any RHTRG grammar.

The set of all L -shaped palindromic left hexagonal arrowheads over the alphabet $\{a, b\}$ is generated by RHTRG having the rules

$$\begin{aligned}
 S &\rightarrow \left[\left[\begin{array}{cccccc} & & \# & \# & \# & \# & \# \\ & & \# & X & X & X & \# \\ & & \# & X & X & X & \# \\ & \# & X & X & X & X & \# \\ & \# & X & X & X & X & \# \\ & \# & X & X & X & X & \# \\ & & \# & X & X & X & \# \\ & & & \# & \# & \# & \# \end{array} \right] \right] \\
 X &\rightarrow \left[\left[\begin{array}{cccccc} & & \# & \# & \# & \# & \# \\ & & \# & R & X & X & \# \\ & & \# & R & X & X & \# \\ & \# & R & X & X & X & \# \\ & \# & R & X & X & X & \# \\ & & \# & R & X & X & \# \\ & & \# & R & X & X & \# \\ & & \# & \# & \# & \# & \# \end{array} \right] \right] / \left[\left[\begin{array}{cccccc} & & \# & \# & \# & \# \\ & & \# & R & \# & \# \\ & & \# & R & \# & \# \\ & \# & R & \# & \# & \# \\ & \# & R & \# & \# & \# \\ & & \# & R & \# & \# \\ & & \# & R & \# & \# \\ & & \# & \# & \# & \# \end{array} \right] \right] \\
 R &\rightarrow \left[\left[\begin{array}{cccccc} & & \# & \# & \# \\ & & \# & A' & \# \\ & & \# & R & \# \\ & \# & R & \# & \# \\ & \# & R & \# & \# \\ & & \# & R & \# \\ & & \# & A'' & \# \\ & & \# & \# & \# \end{array} \right] \right] / \left[\left[\begin{array}{cccccc} & & \# & \# & \# \\ & & \# & B' & \# \\ & & \# & R & \# \\ & \# & R & \# & \# \\ & \# & R & \# & \# \\ & & \# & R & \# \\ & & \# & B'' & \# \\ & & \# & \# & \# \end{array} \right] \right]
 \end{aligned}$$

$A', A'' \rightarrow a; B', B'' \rightarrow b; R \rightarrow a, b$

This language is clearly not generated by any HTS as there is no local strategy available for the corresponding palindromic alphabet positions. \square

Theorem 3. $\mathcal{L}(RHTRG) \subset \mathcal{L}(HTRG)$.

Proof. Since RHTRG grammar rules are restricted form of HTRG rules, every RHTRG rule is a rule in HTRG also. Therefore $\mathcal{L}(RHTRG) \subseteq \mathcal{L}(HTRG)$. But $\mathcal{L}(RHTRG) \neq \mathcal{L}(HTRG)$ is seen from the grammar given for the language L_3 where the variable size rules are of the form of HTRG and not in RHTRG. \square

6 Conclusion

Regional hexagonal tile rewriting grammars are the simple type of hexagonal tiling based array rewriting models. They have higher generative capacity than

CFHAG but less general than HTRG. Its incomparability and non-empty disjointness with hexagonal tiling system shows that it may include some subclass of HTS languages, which we have to explore further.

Practical applicability of image processing tasks (like pattern recognition, biomedical image analysis, 3 dimensional structure interpretation) remains to be investigated, which will largely depend on time complexity of this new model and on good parsing algorithm.

References

1. Cherubini, A., Crespi Reghizzi, S., Pradella, M.: Regional Languages and Tiling: A Unifying Approach to Picture Grammars. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 253–264. Springer, Heidelberg (2008)
2. Crespi Reghizzi, S., Pradella, M.: Tile Rewriting Grammars. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 206–217. Springer, Heidelberg (2003)
3. Dersanambika, K.S., Krithivasan, K., Martin-Vide, C., Subramanian, K.G.: Local and recognizable hexagonal picture languages. *International Journal of Pattern Recognition and Artificial Intelligence* 19, 853–871 (2005)
4. Middleton, L., Sivaswamy, J.: Hexagonal image processing: A practical approach. *Advances in Computer Vision and Pattern Recognition Series*. Springer (2005)
5. Prussa, D.: Two-dimensional Languages. Ph.D. Thesis (2004)
6. Siromoney, G., Siromomey, R.: Hexagonal arrays and rectangular blocks. *Computer Graphics and Image Processing* 5, 353–381 (1976)
7. Siromoney, G., Siromomey, R., Krithivasan, K.: Picture languages with array rewriting rules. *Information and Control* 22, 447–470 (1973)
8. Subramanian, K.G.: Hexagonal array grammars. *Computer Graphics and Image Processing* 10, 388–394 (1979)
9. Thomas, D.G., Sweet, F., Kalyani, T.: Results on Hexagonal Tile Rewriting Grammars. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part I. LNCS, vol. 5358, pp. 945–952. Springer, Heidelberg (2008)

Partial Commutation on Array Languages

Thangasamy Kamaraj¹, Durairaj Gnanaraj Thomas²,
H. Geetha³, and T. Kalyani³

¹ Department of Mathematics, Sathyabama University
Chennai - 119, India

kamaraj_mx@yahoo.co.in

² Department of Mathematics, Madras Christian College
Tambaram, Chennai - 600 059, India

dgthomasbcc@yahoo.com

³ Department of Mathematics, St. Joseph's College of Engineering
Chennai - 600 119, India

geethahvenki@hotmail.com

Abstract. Algebraic characterization of recognizable trace languages were studied and the relation of recognizable trace languages to elementary Petri nets were established by using partial commutation as a tool. Motivated by the above studies in string languages, we have extended the notion of partial commutation to two-dimensional array languages and established that if L is local then $\phi(L)$ need not be local, where ϕ is a partial commutation mapping. We have proved that $L(\phi(\theta))$ and $\phi(L(\theta))$ are not disjoint, where θ is a finite set of 2×2 tiles over the alphabet $\Gamma \cup \{\#\}$. We have also considered partial commutation mapping on Siromoney matrix languages and proved some interesting results.

Keywords: Tiling system, Projection, Local recognizability, Partial commutation and trace languages, Siromoney matrix grammar.

1 Introduction

A two-dimensional language is a set of two-dimensional patterns which appear in the studies concerning cellular automata, parallel computing and image analysis [5, 8, 9, 12, 13]. The notion of recognizability in two-dimension is the most ambitious property for picture languages. The generalization of finite state automaton to two-dimensional languages can be attributed to M. Blum and C. Hewitt [1] and they have introduced the notion of 4-way automaton. The idea of recognizability of a set of pictures in terms of tiling system has been introduced in [4]. The underlying idea is to define recognizability by projection of local properties. Siromoney et al. introduced various matrix grammars for generating rectangular picture languages [10].

On the other hand the theory of traces has been motivated by the theory of Petri nets and by the theory of formal languages and automata [3]. The original attempt of the theory of traces was to use the well developed tools of formal language theory for the analysis of concurrent systems as it is done in Petri nets.

Based on the behaviour of elementary net systems Mazurkiewicz introduced the concept of partial commutation on string languages to the computer science community [6]. The abstract description of a concurrent process is then called a trace, being defined as a congruence class of word modulo identities of the form $ab \equiv ba$ for some pairs of letters. The success of Mazurkiewicz approach results from the fact that the partial commutation copes with some important phenomena in concurrency. Since the work of Cartier and Foata [2] in combinatorics, the trace theory has grown in breadth and depth.

Motivated by the above studies we have considered partial commutation on rectangular arrays and examined whether the language obtained by applying partial commutation on a local language is local or not. Similar question has been analyzed for Siromoney matrix languages.

2 Preliminaries

In this section we recall the notions of recognizability of two-dimensional picture languages [4] and Siromoney matrix grammars [10] and partial commutation on strings [6].

Definition 1. *A two-dimensional string (or a picture) over Σ is a two-dimensional rectangular array of elements of Σ . The set of all two-dimensional strings over Σ is denoted by Σ^{**} . A two-dimensional language over Σ is a subset of Σ^{**} .*

Given a picture $p \in \Sigma^{**}$, let $\ell_1(p)$ denote the number of rows of p and $\ell_2(p)$ denote the number of columns of p . The pair $(\ell_1(p), \ell_2(p))$ is called the size of the picture p . The empty picture is the only picture of size $(0, 0)$ denoted by Λ . The set of all pictures of size (m, n) with $m, n > 0$ is represented by $\Sigma^{m \times n}$.

The notion of recognizability of a set of pictures in terms of tiling systems is introduced in [4]. The recognition in a tiling system is defined in terms of a finite set of square pictures of size two by two over a new alphabet Γ . The tiles define a local language over Γ and a projection from Γ to Σ , applied to this local language, produces the language over Σ recognized by the tiling system.

Definition 2. *Let $p \in \Gamma^{**}$ be a picture. The projection by the mapping π of a picture p is the picture $p' \in \Sigma^{**}$ such that $p'(i, j) = \pi(p(i, j))$ for all $1 \leq i \leq \ell_1(p)$ and $1 \leq j \leq \ell_2(p)$, where $p(i, j)$ is the (i, j) th element of p .*

Definition 3. *Let $L \subseteq \Gamma^{**}$ be a picture language. The projection by the mapping π of L is the language $L' = \{p' | p' = \pi(p) \forall p \in L\} \subseteq \Sigma^{**}$. We denote the language obtained by the projection of L by the mapping π as $\pi(L)$.*

Let p be a picture of size (m, n) . We denote by \tilde{p} the picture of size $(m+2, n+2)$ obtained by bordering p with a special boundary symbol $\# \notin \Sigma$. $B_{h,k}(p)$ denotes the set of all subpictures of p of size (h, k) . A tile is a picture of size $(2, 2)$.

Definition 4. *A tiling system is a 4-tuple $T = (\Sigma, \Gamma, \theta, \pi)$ where Σ and Γ are alphabets, θ is a finite set of tiles over the alphabet $\Gamma \cup \{\#\}$ and $\pi : \Gamma \rightarrow \Sigma$ is a projection.*

Definition 5. Let Γ be a finite alphabet. A two-dimensional language $L \subseteq \Gamma^{**}$ is local if there exists a finite set θ of tiles over the alphabet $\Gamma \cup \{\#\}$ such that $L = \{p \in \Gamma^{**} \mid B_{2,2}(\tilde{p}) \subseteq \theta\}$. The language L is denoted by $L(\theta)$.

The family of local picture languages is denoted by LOC. The set θ represents the set of allowed blocks for pictures belonging to the local language L . Given a language L , we can consider the set θ as the set of all possible blocks of size $(2, 2)$ of pictures that belong to L (when considered with the frame of $\#$ symbols).

Definition 6. A language $L \subseteq \Sigma^{**}$ is tiling recognizable if there exists a tiling system $T = (\Sigma, \Gamma, \theta, \pi)$ such that $L = \pi(L(\theta))$.

$\mathcal{L}(TS)$ is the family of all two-dimensional languages recognizable by tiling systems.

We now consider Siromoney matrix grammars [10].

Definition 7. A Siromoney matrix grammar G is a 2-tuple (G_1, G_2) , where (i) $G_1 = (V_1, I_1, P_1, S)$ is a regular or a context-free or a context-sensitive or a phrase-structure grammar with V_1 , a finite set of horizontal non-terminals; $I_1 = \{S_1, \dots, S_k\}$, a finite set of intermediates, $V_1 \cap I_1 = \phi$; P_1 , a finite set of production rules called horizontal production rules and $S \in V_1$, the start symbol and (ii) $G_2 = \bigcup_{i=1}^k G_{2i}$ where $G_{2i} = (V_{2i}, T_i, P_{2i}, S_i)$, $1 \leq i \leq k$, a regular grammar with V_{2i} , a finite set of vertical non-terminals, $V_{2i} \cap V_{2j} = \phi$, $i \neq j$; T_i , a finite set of terminals, P_{2i} , a finite set of right linear production rules called vertical production rules of the form $X \rightarrow aY$ or $X \rightarrow a$ where $X, Y \in V_{2i}$, $a \in T_i$ and $S_i \in V_{2i}$, the start symbol of G_{2i} .

The grammar G is called a regular (RMG), context-free (CFMG), context-sensitive (CSMG) and phrase-structure (PSMG) Siromoney matrix grammar if G_1 is regular, context-free, context-sensitive and phrase-structure respectively.

Derivations are defined as follows: First a string, $S_{i1}S_{i2} \dots S_{in} \in I_1^*$ is generated horizontally using the horizontal production rules of P_1 in G_1 . i.e., $S \Rightarrow S_{i1}S_{i2} \dots S_{in} \in I_1^*$.

Vertical derivations proceed as follows: We write

$$\begin{array}{c} A_{i1} \dots A_{in} \\ \Downarrow \\ B_{i1} \dots B_{in} \\ a_{i1} \dots a_{in} \end{array}$$

if $A_{ij} \rightarrow a_{ij}B_{ij}$ are rules in P_{2j} , $1 \leq j \leq n$. The derivation terminates if $A_{ij} \rightarrow a_{mj}$ are terminal rules in G_2 .

The set $L(G)$ generated by G consists of all $m \times n$ arrays $[a_{ij}]$ such that $1 \leq i \leq m$, $1 \leq j \leq n$ and $S \xRightarrow{*}_{G_1} S_{i1}S_{i2} \dots S_{im} \xRightarrow{*}_{G_2} [a_{ij}]$.

Definition 8. $L(G)$ is called a phrase-structure matrix language (PSML), context-sensitive matrix language (CSML), context-free matrix language (CFML), regular matrix language (RML) if G is a PSMG, CSMG, CFMG, RMG respectively.

The Siromoney matrix grammars were extended as Tabled Matrix Grammars [11], by specifying a finite set of tables of rules in the second phase of generation with each table having either right-linear non terminal rules or terminal rules. The resulting families of picture array languages are denoted by TRML (TCFML, TCSML) and are known to properly include class of RML (CFML, CSML). These models generate wide class of picture languages, including simple pictures like H and ladder, which Siromoney matrix models cannot generate.

We now recall partial commutation on strings [6,7].

Let Σ be a finite alphabet. Let $I \subseteq \Sigma \times \Sigma$ be a symmetric and irreflexive relation over the alphabet Σ , called the independence (or commutation) relation. $(a, b) \in I$ means that $ab = ba$.

The relation I induces an equivalence relation \sim_I over Σ^* . Two words x and y are equivalent under \sim_I , denoted by $x \sim_I y$, if there exists a sequence z_1, z_2, \dots, z_k of words such that $x = z_1$, $y = z_k$, and for all i , $1 \leq i < k$, there exist words z'_i, z''_i , and letters a_i, b_i satisfying:

$$z_i = z'_i a_i b_i z''_i, \quad z_{i+1} = z'_i b_i a_i z''_i, \quad \text{and } (a_i, b_i) \in I.$$

Thus, two words are equivalent by \sim_I if one can be obtained from the other by successive transpositions of neighboring independent letters. It is easy to verify that \sim_I is the least congruence over Σ^* such that $ab \sim_I ba$ for all pairs $(a, b) \in I$. The quotient of Σ^* by the congruence \sim_I is the free partially commutative monoid induced by the relation I , denoted by $M(\Sigma, I)$. The elements of $M(\Sigma, I)$, which are equivalence classes of words of Σ^* under the relation \sim_I , are called traces. Consequently, $M(\Sigma, I)$ is called a trace monoid.

For a word x of Σ^* the equivalence class of x under \sim_I is denoted by $[x]_I$. Thus, $[x]_I$ is the set of words which are equivalent to a given word x ,

$$\text{i.e., } [x]_I = \{y \in \Sigma^* | y \sim_I x\}.$$

For instance, if we consider $I = \{(a, d), (d, a), (b, c), (c, b)\}$, we have:

$$[baadcb]_I = \{baadcb, baadbc, badacb, badabc, bdaabc, bdaacb\}.$$

A trace language is any subset of $M(\Sigma, I)$. If $X \subseteq \Sigma^*$, then $[X] = \{[x]_I | x \in X\}$. Clearly, $[X]$ is a trace language.

For further details on partial commutation and traces we refer to [6,7].

3 Trace of Array Languages

In this section, we extend the notion of trace languages to two dimensional picture languages.

Definition 9. Let Σ be an alphabet and $p \in \Sigma^{**}$. The row partial commutation mapping $\phi_R : \Sigma \times \Sigma \rightarrow \Sigma \times \Sigma$ is defined as $\phi_R(a, b) = (b, a)$ and $\phi_R(b, a) = (a, b)$ for $a, b \in \Sigma$. i.e., a and b commute with each other. A row partial commutation is denoted by $(a \ b) \leftrightarrow (b \ a)$. It is applied on any row of p replacing ab by ba and

vice versa. The equivalence class of p obtained by row partial commutation is a trace, denoted by $[p]_R$ or $\phi_R(p)$. The set of all equivalence classes obtained by row partial commutation on a language L , denoted by $\phi_R(L)$, is a trace language. If $L \subseteq \Sigma^{**}$, then $\phi_R(L) = \{[p]_R | p \in L\}$.

Definition 10. Let Σ be an alphabet. The column partial commutation mapping $\phi_C : \left(\begin{smallmatrix} \Sigma \\ \Sigma \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} \Sigma \\ \Sigma \end{smallmatrix}\right)$ is defined as $\phi_C \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}$ and $\phi_C \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$ for $a, b \in \Sigma$. A column partial commutation is denoted by $\begin{pmatrix} a \\ b \end{pmatrix} \leftrightarrow \begin{pmatrix} b \\ a \end{pmatrix}$. It is applied on any column of a picture $p \in \Sigma^{**}$. The equivalence class of p obtained by column partial commutation is a trace, given by $[p]_C$ or $\phi_C(p)$. If $L \subseteq \Sigma^{**}$, then $\phi_C(L) = \{[p]_C | p \in L\}$.

Note. If every element of Σ commutes with each other then the partial commutation is called a total commutation.

Example 1. Let

$$\theta = \left\{ \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & 1 \\ \hline \# & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \# & \# \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|c|} \hline 1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & 0 \\ \hline \# & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline 0 & \# \\ \hline 0 & \# \\ \hline \end{array} \right\}.$$

The local language $L = L(\theta)$ is the language of squares over $\Sigma = \{0, 1\}$ such that the principal diagonal elements are ‘1’ and remaining elements are all ‘0’.

$$\text{i.e., } L = L(\theta) = \{p \in \Sigma^{**} | \ell_1(p) = \ell_2(p) \text{ such that } a_{ii} = 1, \\ a_{ij} = 0 \text{ for } i \neq j, i = 1, 2, \dots, n\}$$

$$\text{i.e., } L = \left\{ \begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}, \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}, \dots \right\}.$$

By applying the row commutativity $\begin{pmatrix} 0 & 1 \end{pmatrix} \leftrightarrow \begin{pmatrix} 1 & 0 \end{pmatrix}$ to each element of L , we get

$$\phi_R(L) = \left\{ \begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}, \begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}, \dots \right\}.$$

$\phi_R(L)$ cannot be local, for if it is local, then there exists a set of tiles θ' such that $L(\theta') = \phi_R(L)$. Then $\begin{array}{cccccc} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{array}$ should be in θ' since

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array} \in \phi_R(L). \text{ But } \begin{array}{cccccc} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array} \text{ is in } L(\theta') \text{ and not in } \phi_R(L).$$

Now by applying the row partial commutation to θ , we get

$$\phi_R(\theta) = \left\{ \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|} \hline \#1 \\ \hline \#0 \\ \hline \end{array}, \begin{array}{|c|} \hline 10 \\ \hline 01 \\ \hline \end{array}, \begin{array}{|c|} \hline 10 \\ \hline 10 \\ \hline \end{array}, \begin{array}{|c|} \hline 01 \\ \hline 01 \\ \hline \end{array}, \begin{array}{|c|} \hline 01 \\ \hline 10 \\ \hline \end{array}, \begin{array}{|c|} \hline 0\# \\ \hline 1\# \\ \hline \end{array}, \begin{array}{|c|} \hline \#0 \\ \hline \#\# \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|} \hline 10 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 01 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 1\# \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline \#\# \\ \hline 00 \\ \hline \end{array}, \begin{array}{|c|} \hline 00 \\ \hline 01 \\ \hline \end{array}, \begin{array}{|c|} \hline 00 \\ \hline 10 \\ \hline \end{array}, \begin{array}{|c|} \hline 0\# \\ \hline 0\# \\ \hline \end{array}, \begin{array}{|c|} \hline \#\# \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 00 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 10 \\ \hline 00 \\ \hline \end{array}, \begin{array}{|c|} \hline 01 \\ \hline 00 \\ \hline \end{array}, \begin{array}{|c|} \hline 00 \\ \hline 00 \\ \hline \end{array} \right\}.$$

Now $101 \in L(\phi_R(\theta))$. But $101 \notin \phi_R(L(\theta))$.

010 010

$010 \in \phi_R(L(\theta))$. But $010 \notin L(\phi_R(\theta))$

001 001

Hence $\phi_R(L(\theta)) \neq L(\phi_R(\theta))$.

By applying column partial commutation $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ on L we get

$$\phi_C(L) = \left\{ \begin{array}{cccccc} 00 & 01 & 11 & 10 & 100 & 100 \\ 11 & 10 & 00 & 01 & 010 & 011, \dots \\ & & & & 010 & 000 \end{array} \right\}$$

Again $\phi_C(L)$ cannot be local and this can be proved in the same way as in the case of $\phi_R(L)$ and $\phi_C(L(\theta)) \neq L(\phi_C(\theta))$.

Example 2. We consider a local language $L = \{1\ 2\ 3, \begin{array}{c} 1\ 2\ 3 \\ 1\ 2\ 3 \end{array}, \begin{array}{c} 1\ 2\ 3 \\ 1\ 2\ 3 \end{array}, \dots\}$ over

$\Sigma = \{1, 2, 3\}$, where

$$\theta = \left\{ \begin{array}{|c|c|} \hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 1 & 2 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 2 & 3 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# & \# \\ \hline 3 & \# \\ \hline \end{array}, \begin{array}{|c|} \hline 3\# \\ \hline 3\# \\ \hline \end{array}, \begin{array}{|c|} \hline \#1 \\ \hline \#1 \\ \hline \end{array}, \right.$$

$$\left. \begin{array}{|c|} \hline 12 \\ \hline 12 \\ \hline \end{array}, \begin{array}{|c|} \hline 23 \\ \hline 23 \\ \hline \end{array}, \begin{array}{|c|} \hline \#1 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 12 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 23 \\ \hline \#\# \\ \hline \end{array}, \begin{array}{|c|} \hline 3\# \\ \hline \#\# \\ \hline \end{array} \right\}.$$

By applying the row partial commutation $(1\ 2) \leftrightarrow (2\ 1)$ on L we get

$$\phi_R(L) = \left\{ 1\ 2\ 3, 2\ 1\ 3, \begin{array}{c} 2\ 1\ 3 \\ 1\ 2\ 3 \end{array}, \begin{array}{c} 1\ 2\ 3 \\ 2\ 1\ 3 \end{array}, \begin{array}{c} 2\ 1\ 3 \\ 2\ 1\ 3 \end{array}, \begin{array}{c} 1\ 2\ 3 \\ 1\ 2\ 3 \end{array}, \dots \right\}.$$

$\phi_R(L)$ cannot be local, for if it is local, then there exists a set of tiles θ' such that $L(\theta') = \phi_R(L)$.

Since $\begin{matrix} 2 & 1 & 3 \\ 1 & 2 & 3 \end{matrix}$, $\begin{matrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{matrix}$, $\begin{matrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{matrix} \in \phi_R(L)$, the set of tiles contained in the picture

$\begin{matrix} \# & \# & \# & \# & \# \\ \# & 2 & 1 & 2 & 3 & \# \\ \# & 1 & 2 & 1 & 3 & \# \\ \# & \# & \# & \# & \# & \# \end{matrix}$ is a subset of θ' .

This implies, $\begin{matrix} 2 & 1 & 2 & 3 \\ 1 & 2 & 1 & 3 \end{matrix} \in L(\theta')$ but not in $\phi_R(L)$. Hence $\phi_R(L)$ is not local.

By applying the row partial commutation $(1\ 2) \leftrightarrow (2\ 1)$ to the elements of θ , we get

$$\phi_R(\theta) = \left\{ \begin{matrix} \# & \# \\ \# & 1 \end{matrix}, \begin{matrix} \# & \# \\ 1 & 2 \end{matrix}, \begin{matrix} \# & \# \\ 2 & 1 \end{matrix}, \begin{matrix} \# & \# \\ 2 & 3 \end{matrix}, \begin{matrix} \# & \# \\ 3 & \# \end{matrix}, \begin{matrix} \# & 1 \\ \# & 1 \end{matrix}, \begin{matrix} 1 & 2 \\ 1 & 2 \end{matrix}, \begin{matrix} 1 & 2 \\ 2 & 1 \end{matrix} \right\},$$

$$\left\{ \begin{matrix} 2 & 1 \\ 1 & 2 \end{matrix}, \begin{matrix} 2 & 1 \\ 2 & 1 \end{matrix}, \begin{matrix} 2 & 3 \\ 2 & 3 \end{matrix}, \begin{matrix} 3 & \# \\ 3 & \# \end{matrix}, \begin{matrix} \# & 1 \\ \# & \# \end{matrix}, \begin{matrix} 2 & 3 \\ \# & \# \end{matrix}, \begin{matrix} 3 & \# \\ \# & \# \end{matrix}, \begin{matrix} 1 & 2 \\ \# & \# \end{matrix}, \begin{matrix} 2 & 1 \\ \# & \# \end{matrix} \right\}.$$

Now $\begin{matrix} 1 & 2 & 1 & 2 & 3 \\ 1 & 2 & 1 & 2 & 3 \end{matrix} \in L(\phi_R(\theta))$.

But $\begin{matrix} 1 & 2 & 1 & 2 & 3 \\ 1 & 2 & 1 & 2 & 3 \end{matrix} \notin \phi_R(L(\theta))$.

Again $\begin{matrix} 2 & 1 & 3 \\ 2 & 1 & 3 \end{matrix} \in \phi_R(L(\theta))$. But $\begin{matrix} 2 & 1 & 3 \\ 2 & 1 & 3 \end{matrix} \notin L(\phi_R(\theta))$.

So, $\phi_R(L(\theta)) \neq L(\phi_R(\theta))$.

If we denote the transpose of θ by θ^t we see that $\phi_R(L(\theta^t))$ is local and $\phi_R(L(\theta^t)) = L(\phi_R(\theta^t))$.

By applying column commutativity $\begin{pmatrix} 1 \\ 2 \end{pmatrix} \leftrightarrow \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ to the elements of L , we get

$\phi_C(L) = L$ and $\phi_C(\theta) = \theta$.

Clearly, $\phi_C(L)$ is local and $\phi_C(L(\theta)) = L(\phi_C(\theta))$.

But we see that $\phi_C(L(\theta^t))$ is not local and $\phi_C(L(\theta^t)) \neq L(\phi_C(\theta^t))$.

Remark 1. If all the elements of $\Sigma = \{1, 2, 3\}$ are commuted, that is, $(1\ 2) \leftrightarrow (2\ 1)$, $(2\ 3) \leftrightarrow (3\ 2)$ and $(1\ 3) \leftrightarrow (3\ 1)$ then $\phi_R(L(\theta))$ is not local. In other words whenever the partial commutation becomes a total commutation, $\phi_R(L(\theta))$ need not be local.

From the above examples and the remark, we have the following results.

Theorem 1. Suppose ϕ is either a row partial (or) a column partial commutation mapping and θ is a finite tile set over the alphabet $\Gamma \cup \{\#\}$.

- (i) If L is local then $\phi(L)$ need not be local.
- (ii) If L is local then $\phi(L(\theta))$ and $L(\phi(\theta))$ need not be equal
- (iii) $\phi(L(\theta))$ and $L(\phi(\theta))$ are not disjoint.

Proof. (i) and (ii) follow from Examples 1 and 2.

(iii) Since ϕ induces an equivalence relation on $L(\theta)$, $L(\theta) \subseteq \phi(L(\theta))$.

Since $\theta \subseteq \phi(\theta)$, $L(\theta) \subseteq L(\phi(\theta))$.

So $L(\theta) \subseteq \phi(L(\theta)) \cap L(\phi(\theta))$. In other words $\phi(L(\theta))$ and $L(\phi(\theta))$ are not disjoint. \square

Note. If we apply either row partial commutation mapping $(0 \ 1) \leftrightarrow (1 \ 0)$ or column partial commutation mapping $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ (one operation at a time) on L , given in Example 1, we get the array language

$$\left\{ \begin{array}{l} 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \\ 0 \ 1' \ 0 \ 1' \ 1 \ 0' \ 0 \ 0' \ 1 \ 1' \ 0 \ 0', 0 \ 1 \ 0, 0 \ 0 \ 1, \dots \end{array} \right\}$$

which can be denoted by $\phi_{R/C}(L)$. Clearly $\phi_{R/C}(L)$ is also not local and $L(\phi_{R/C}(\theta)) \neq \phi_{R/C}(L(\theta))$.

Proposition 1. *If L, L_1 and L_2 are any three array languages over Σ and ϕ_C and ϕ_R are respectively partial column and row commutation mappings, then, the following properties are true*

- (i) $\phi_C(L_1 \cup L_2) = \phi_C(L_1) \cup \phi_C(L_2)$
- (ii) $\phi_C(L_1) \subseteq \phi_C(L_2)$ if $L_1 \subseteq L_2$
- (iii) $\phi_C(\phi_R(L))$ need not be equal to $\phi_R(\phi_C(L))$.
- (iv) $\phi_C(L_1) \ominus \phi_C(L_2)$ need not be equal to $\phi_C(L_1 \ominus L_2)$
- $\phi_C(L_1) \oplus \phi_C(L_2)$ need not be equal to $\phi_C(L_1 \oplus L_2)$
- (v) $\phi_C(L_1 \cap L_2)$ need not be equal to $\phi_C(L_1) \cap \phi_C(L_2)$

Properties (ii), (iii), (iv), (v) hold good if we replace ϕ_C by ϕ_R .

4 Partial Commutation on Siromoney Matrix Languages

In this section, we consider Siromoney matrix grammars which are well known classical picture generating models and examine the partial commutativity applied on Siromoney matrix languages.

Example 3. Consider the regular matrix language consisting of $m \times n$ arrays ($m > 1, n > 1$) on $\{\cdot, X\}$ describing the L tokens (Fig. 1), which is generated by the RMG, $G = \langle G_1, G_2 \rangle$

where $G_1 = (\{S, A\}, \{I_1, I_2\}, \{S \rightarrow I_1 A, A \rightarrow I_2 A, A \rightarrow I_2\}, S)$

and $G_2 = G_{21} \cup G_{22}$,

with $G_{21} = (\{I_1, B\}, \{X\}, \{I_1 \rightarrow X B, B \rightarrow X B/X\}, I_1)$

and $G_{22} = (\{I_2, C\}, \{\cdot, X\}, \{I_2 \rightarrow \cdot C, C \rightarrow \cdot C/X\}, I_2)$

If we apply column partial commutation $\begin{pmatrix} \cdot \\ X \end{pmatrix} \leftrightarrow \begin{pmatrix} X \\ \cdot \end{pmatrix}$ on $L(G)$ then

$$\phi_C(L(G)) = \left\{ \begin{array}{l} X \cdot \quad X X \quad X \cdot \quad X \cdot \quad X X \\ X X \quad X \cdot \quad X \cdot \quad X X \quad X \cdot \cdot \cdot \dots \end{array} \right\}$$

$X \cdot \cdot \cdot \cdot$
 $X \cdot \cdot \cdot \cdot$
 $X \cdot \cdot \cdot \cdot$
 $X \cdot \cdot \cdot \cdot$
 $X \cdot \cdot \cdot \cdot$
 $X X X X X$

Fig. 1.

In $\phi_C(L(G))$, we observe that first column of every picture contains all X 's and each of other columns contains only one element as X and other elements as \cdot 's.

This language can be generated by the following RMG

$$G' = \langle G'_1, G'_2 \rangle$$

where $G'_1 = G_1$ and $G'_2 = G'_{21} \cup G'_{22}$ with $G'_{21} = G_{21}$ and $G'_{22} = (\{I_2, B', C, D, E\}, \{\cdot, X\}, \{I_2 \rightarrow XB' | \cdot C | \cdot D, B' \rightarrow \cdot B' | \cdot, C \rightarrow \cdot C | X, D \rightarrow \cdot D | XE, E \rightarrow \cdot E / \cdot\})$.

If we apply row partial commutation $(\cdot X) \leftrightarrow (X \cdot)$ on $L(G)$, then

$$\phi_R(L(G)) = \left\{ \begin{array}{cccccccc} X \cdot & \cdot X & X \cdot & \cdot X & X \cdot & \cdot X & \cdot X & \cdot X \\ X X & X X & X X & X X & X X & X X & X X & X X \end{array} , \dots \right\}$$

In $\phi_R(L(G))$ we observe that each row (except last) of every picture consists of one X (at any position) and remaining \cdot 's. The last row contains all X 's. This means that every column has a chance of having more than one X (entire column may be X 's).

If an RMG $G'' = \langle G''_1, G''_2 \rangle$ generates $\phi_R(L(G))$, then all the vertical production rules in G''_2 should produce column strings as discussed above. This

means, we have a picture $\begin{array}{c} X X \\ X X \\ X X \\ X X \end{array}$ in $L(G'')$, which clearly indicates that $L(G'') \neq$

$\phi_R(L(G))$.

Example 4. Consider the context free matrix language consisting of $m \times n$ arrays ($m \geq 3, n \geq 3$) on $\{\cdot, X\}$ describing I tokens (Fig. 2), which is generated by the CFMG, $G = \langle G_1, G_2 \rangle$

where $G_1 = (\{S, A\}, \{I_1, I_2\}, \{S \rightarrow I_1 S I_1, S \rightarrow I_1 A I_1, A \rightarrow I_2\}, S)$,

$G_2 = G_{21} \cup G_{22}$

with $G_{21} = (\{I_1, B, C\}, \{\cdot, X\}, \{I_1 \rightarrow X B, B \rightarrow \cdot C, C \rightarrow \cdot C / X\}, I_1)$

and $G_{22} = (\{I_2, O, E\}, \{X\}, \{I_2 \rightarrow X O, O \rightarrow X E, E \rightarrow X E / X\}, I_2)$


```

X X X X X X X
. . . X . . .
. . . X . . .
. . . X . . .
X X X X X X X
    
```

Fig. 2.

If we apply column partial commutation $\begin{pmatrix} \cdot \\ X \end{pmatrix} \leftrightarrow \begin{pmatrix} X \\ \cdot \end{pmatrix}$ on $L(G)$, then

$$\phi_C(L(G)) = \left\{ \begin{array}{l} X X X \quad \cdot X X \quad \cdot X \cdot \\ X X \cdot, X X \cdot, X X X, \dots \\ \cdot X X \quad X X X \quad X X X \end{array} \right\}$$

This language can be generated by the following CFMG $G' = \langle G'_1, G'_2 \rangle$ where $G'_1 = G_1$, $G'_2 = G'_{21} \cup G'_{22}$ with $G'_{21} = (\{I_1, B', C', D, F, I, J, K, L, M\}, \{\cdot, X\}, R, I_1)$ where

- $R = \{I_1 \rightarrow XB' / \cdot F,$
- $B' \rightarrow \cdot C',$
- $C' \rightarrow \cdot C' / XD / X,$
- $D \rightarrow \cdot D / \cdot,$
- $F \rightarrow \cdot K / XL,$
- $K \rightarrow \cdot K / XM,$
- $M \rightarrow \cdot M / XI / X,$
- $L \rightarrow \cdot L / XJ / X,$
- $I \rightarrow \cdot I / \cdot,$
- $J \rightarrow \cdot J / \cdot\}$

and $G'_{22} = G_{22}$.

Example 5. Consider the context-sensitive matrix language consisting of four pronged forks (without handle) of all sizes and proportions (but retaining equal intervals between forks) (Fig. 3) which is generated by the CSMG, $G = \langle G_1, G_2 \rangle$

```

X X X X X X X X X
X . . X . . X . . X
X . . X . . X . . X
X . . X . . X . . X
X . . X . . X . . X
    
```

Fig. 3.

where $G_1 = (\{S, S_1, A, B, C, D, E, F\}, \{I_1, I_2\}, R, S)$ with $R = \{S \rightarrow ABCS_1, S_1 \rightarrow ABCS_1, S_1 \rightarrow DI_1, BA \rightarrow AB, CA \rightarrow AC,$

$$\begin{aligned} &CB \rightarrow BC, CD \rightarrow DC, BD \rightarrow EBI_1, BE \rightarrow EB, AE \rightarrow FAI_1, \\ &AF \rightarrow FA, F \rightarrow I_1, I_1A \rightarrow I_1I_2, I_2A \rightarrow I_2I_2, I_1B \rightarrow I_1I_2, \\ &I_2B \rightarrow I_2I_2, I_1C \rightarrow I_1I_2, I_2C \rightarrow I_2I_2 \} \end{aligned}$$

and $G_{12} = G_{21} \cup G_{22}$

with $G_{21} = (\{I_1, J\}, \{X\}, \{I_1 \rightarrow XJ, J \rightarrow XJ, J \rightarrow X\}, I_1)$

and $G_{22} = (\{I_2, K\}, \{X, \cdot\}, \{I_2 \rightarrow XK, K \rightarrow \cdot K/\cdot\}, I_2)$

If we apply column partial commutation $\begin{pmatrix} \cdot \\ X \end{pmatrix} \leftrightarrow \begin{pmatrix} X \\ \cdot \end{pmatrix}$, then

$$\phi_C(L(G)) = \left\{ \begin{array}{l} X X X X X X X \quad X \cdot X X X X X \\ X \cdot X \cdot X \cdot X \cdot X \quad X X X \cdot X \cdot X, \dots \end{array} \right\}.$$

This language can be generated by the following CSMG.

$G' = \langle G'_1, G'_2 \rangle$ where

$$G'_1 = G_1$$

$$G'_2 = G'_{21} \cup G'_{22} \text{ with}$$

$$G'_{21} = G_{21}$$

$$G'_{22} = (\{I_2, B', C', D', E'\}, \{\cdot, X\}, \{I_2 \rightarrow XB'/\cdot C'/\cdot D', B' \rightarrow \cdot B'/\cdot, \\ C' \rightarrow \cdot C'/X, D' \rightarrow \cdot D'/XE', E' \rightarrow \cdot E'/\cdot\}, I_2)$$

Remark 2. In both examples 4 and 5, if we apply row partial commutation on L defined as $(\cdot X) \leftrightarrow (X \cdot)$, then every column has a chance of having more than one X . So, by a similar argument made in the example 3, we see that $\phi_R(L(G))$ cannot be generated by any CFMG or CSMG.

Theorem 2. (i) If L is a RML (CFML, CSML) then $\phi_C(L)$ is also a RML (CFML, CSML).

(ii) If L is a RML (CFML, CSML) then $\phi_R(L)$ need not be a RML (CFML, CSML).

Proof. Column partial commutation does not change the number of occurrences of each terminal in every column. For a RML (CFML, CSML) L defined on two symbols, we claim that there exists a RMG (CFMG, CSMG) $G' = \langle G'_1, G'_2 \rangle$ generating $\phi_C(L)$. To show this, let L be a RML (CFML, CSML) on alphabet $\{a, b\}$. If a column of $p \in \phi_C(L)$ contains only a 's and no b 's, then the vertical production rules of G'_2 are of the form $I \rightarrow aA, A \rightarrow aA, A \rightarrow a$. If a column of $p \in \phi_C(L)$ contains only one b and remaining a 's, then the vertical production rules of G'_2 are of the form $I \rightarrow bA/aB/aC, A \rightarrow aA/a, B \rightarrow aB/b, C \rightarrow aC/a/bD, D \rightarrow aD/a$. If a column of $p \in \phi_C(L)$ contains two or more b 's and remaining a 's, then the vertical production rules of G'_2 are of the form $I \rightarrow bA/aB, A \rightarrow aC, C \rightarrow aC/bD/b, D \rightarrow aD/a, B \rightarrow aE/bF, E \rightarrow aE/bH, H \rightarrow aH/bK/b, F \rightarrow aF/bJ/b, K \rightarrow aK/a, J \rightarrow aJ/a$.

By induction on the size of the alphabet (greater than or equal to 2) we can show that, given a RML (CFML, CSML) L , there exists a RMG (CFMG, CSMG) G' generating $\phi_C(L)$. Hence first part of the theorem follows.

Second part of the theorem is obvious from Example 3 and Remark 2, where we observe that $\phi_C(L(G)) \neq L(G')$ for any RMG (CFMG, CSMG) G' . \square

We now give an example of a Tabled Matrix grammar. This grammar brings out the dependence between several columns in vertical derivations unlike Siromoney Matrices grammars, which cannot bring such dependency.

Example 6. Consider the TRMG $G = \langle G_1, G_2 \rangle$ where the language generated by the horizontal grammar G_1 with the set of intermediates $\{I_1, I_2\}$ is $L(G_1) = \{I_1 I_2^n I_1 \mid n \geq 1\}$. The tables of G_2 are

- $t_1 = \{I_1 \rightarrow XI_1, I_2 \rightarrow \cdot I_2\},$
- $t_2 = \{I_1 \rightarrow XI_1, I_2 \rightarrow \cdot A\},$
- $t_3 = \{I_1 \rightarrow XI_1, A \rightarrow XB\},$
- $t_4 = \{I_1 \rightarrow XI_1, B \rightarrow \cdot B\},$
- $t_5 = \{I_1 \rightarrow X, B \rightarrow \cdot\},$

The tabled regular matrix language L generated by G is the set of all $m \times n$ ($m, n \geq 3$) arrays on $\{X\}$, describing the token H of X 's of different sizes and proportions (Fig. 4).

$$\begin{array}{c} X \cdot \cdot X \\ X \cdot \cdot X \\ X \cdot \cdot X \\ X X X X \\ X \cdot \cdot X \end{array}$$

Fig. 4.

The language $\phi_C(L)$ obtained by applying column partial commutation $(\cdot X) \leftrightarrow \begin{pmatrix} X \\ \cdot \end{pmatrix}$ on L , can be generated by the following TRMG $G' = \langle G'_1, G'_2 \rangle$ where the horizontal grammar $G'_1 = G_1$ and the tables of vertical grammar G'_2 are:

- $t_1 = \{I_1 \rightarrow XI_1/XB, B \rightarrow XB, I_2 \rightarrow XC, C \rightarrow \cdot D, D \rightarrow \cdot D, I_2 \rightarrow \cdot E/\cdot F,$
 $E \rightarrow \cdot J, J \rightarrow \cdot J, F \rightarrow \cdot F/XK, K \rightarrow \cdot K\}$
- $t_2 = \{B \rightarrow X, D \rightarrow \cdot, J \rightarrow \cdot, K \rightarrow \cdot\}.$

But $\phi_R(L)$, obtained by applying row partial commutation $(\cdot X) \leftrightarrow (X \cdot)$ on L , cannot be generated by any TRMG G'' , which we can argue in the same way as in Example 3.

We can also have examples on TCFML and TCSML, by choosing the horizontal languages to be the CFL $\{I_1^n I_2 I_1^n \mid n \geq 1\}$ and the CSL $\{I_1^n I_2 I_1^n I_2 I_1^n \mid n \geq 1\}$ respectively and retaining the same vertical tables as in Example 6. Again, $\phi_C(L)$ (where L is TCFML or TCSML) can be generated by a TCFMG or TCSMG G' , whose construction is similar to that of TRMG G' .

Again $\phi_R(L)$ when L is a TCFML or TCSML cannot be generated by any TCFMG or TCSMG G'' .

- Proposition 2.** (i) *If L is a TRML (TCFML, TCSML), then $\phi_C(L)$ is also a TRML (TCFML, TCSML).*
(ii) *If L is a TRML (TCFML, TCSML), then $\phi_R(L)$ need not be a TRML (TCFML, TCSML).*

Proof follows in similar lines with that of Theorem 2.

5 Conclusion

The study of partial commutation for strings yielded many algorithmic and algebraic results. The study of commutativity in 2D languages is novel and worth examining to obtain the algorithmic and algebraic results. In section 3 we have dealt with row partial commutation mapping ϕ_R and column partial commutation mapping ϕ_C . We have proved that if L is local and ϕ is either a row or column partial commutation mapping, then $\phi(L)$ need not be local. We have established a theorem: $L(\phi(\theta))$ and $\phi(L(\theta))$ are not disjoint. In section 4 we have proved that if L is a RML (CFML, CSML), then $\phi_C(L)$ is a RML (CFML, CSML). But $\phi_R(L)$ need not be a RML (CFML, CSML). Partial commutation on recognizable languages and other classes of array languages is under investigation. Application of this study in analyzing the structure of images and parsing can be explored in future.

References

1. Blum, M., Hewitt, C.: Automata on a 2-dimensional tape. In: Proceedings of IEEE Symposium on Switching and Automata Theory, pp. 155–160 (1967)
2. Cartier, P., Foata, D.: Problemes combinatoires de commutation et rearrangement. Lecture Notes in Mathematics, vol. 85. Springer, Heidelberg (1969)
3. Diekert, V., Rozenberg, G. (eds.): The Book of Traces. World Scientific Publishers, Singapore (1995)
4. Giammarresi, D., Restivo, A.: Recognizable picture languages. International Journal on Pattern Recognition and Artificial Intelligence; Nivat, M., Soudi, A., Wang, P.S.P. (eds.) Special Issue on Parallel Image Processing, pp. 31–46 (1992)
5. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Handbook of Formal Languages, vol. 3, pp. 215–226. Springer, Berlin (1997)
6. Mazurkiewicz, A.: Introduction to Trace Theory. In: Diekert, V., Rozenberg, G. (eds.) The Book of Traces, pp. 3–41. World Scientific Publishers, Singapore (1995)
7. Ochmański, E.: Recognizable trace languages. In: Diekert, V., Rozenberg, G. (eds.) The Book of Traces, ch. 6, pp. 167–204. World Scientific Publishers, Singapore (1995)
8. Rosenfeld, A.: Picture languages: Formal Models for Picture Recognition. Academic Press (1979)
9. Rosenfeld, A., Siromoney, R.: Picture languages - A survey. Languages of Design 1, 229–245 (1993)
10. Siromoney, G., Siromoney, R., Krithivasan, K.: Abstract families of matrices and picture languages. Computer Graphics and Image Processing 2, 284–307 (1972)
11. Siromoney, R., Subramanian, K.G., Rangarajan, K.: Parallel/sequential rectangular arrays with tables. International Journal of Computer Mathematics 6A, 143–158 (1977)
12. Subramanian, K.G.: P Systems and Picture Languages. In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 99–109. Springer, Heidelberg (2007)
13. Thomas, W.: On Logics, Tilings and Automata. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 441–453. Springer, Heidelberg (1991)

Incremental Learning of the Model for Watershed-Based Image Segmentation

Anja Attig and Petra Perner

Institute of Computer Vision and Applied Computer Sciences, IBAI, Leipzig, Germany
pperner@ibai-institut.de
www.ibai-institut.de

Abstract. Many image analysis methods need a lot of parameters that have to be adjusted to the particular image in order to achieve the best results. Therefore, methods for parameter learning are required that can assist a system developer in building a model. This task is usually called meta-learning. We consider meta-learning for learning the image segmentation parameters so that the image segmenter can be applied to a wide range of images while achieving good image segmentation quality. The meta-learner is based on case-based reasoning. The cases in the case base are comprised of an image description and the solutions that are the associated parameters for the image segmenter. First, the image description is calculated from a new image. The image description is used to index the case base. The closest case is retrieved based on a similarity measure. Then the associated segmentation parameters are given to the image segmenter and the actual image is segmented. We explain the architecture of such a case-based reasoning image segmenter. The case-description as well as the similarity function are described. Finally, we give results on the image segmentation quality.

Keywords: image segmentation, case-based learning, image similarity, feature description.

1 Introduction

The aim of image processing is to develop methods for automatic extraction of desired information from an image or a video. The developed system should assist the user in processing and understanding the content of a complex signal, such as an image. Usually, an image consists of thousands of pixels. This information can hardly be quantitatively analyzed by the user. In fact, some problems related to the subjective factor or to the tiredness of the user arise, which may influence the interpretation. Therefore, an automatic procedure for analyzing images is necessary.

Although in some cases it might make sense to process a single image and to adjust the parameters of the image processing algorithm to this single image manually, mostly the automation of the image analysis makes sense only if the developed methods have to be applied to more than one single image. This is still an open problem in image processing. The parameters involved in the selected processing method have to

be adjusted to the specific image. It is often hardly possible to select the parameters for a class of images in such a way that the best result can be ensured for all images of the class. Therefore, methods for parameter learning are required that can assist a system developer in building a model [8] for the image processing task.

While the meta-learning task has been extensively studied for classifier selection it has not been studied so extensively for parameter learning. Soares et. al [14] studied parameter selection for the identification of the kernel width of a support-vector machine, while Perner [7] studied parameter selection for image segmentation.

The meta-learning problem for parameter selection can be formalized as following: For a given signal that is characterized by specific signal properties A and domain properties B find the parameters P of the processing algorithm that ensure the best quality of the resulting output signal/information:

$$f : A \cup B \rightarrow P \quad (1)$$

Meta-data for images may consist of image-related meta-data (gray-level statistics) and non-image related meta-data (sensor, object data) [9]. In general the processing of meta-data from signals and images should not require too heavy processing and should allow characterizing the properties of the signal that influence the signal processing algorithm.

The mapping function f can be realized by any classification algorithm, but the incremental behavior of Case-Based Reasoning (CBR) fits best to many data/signal processing problems, where the signal-class cannot be characterized ad-hoc since the data appear incrementally. The right similarity metric that allows mapping data to parameter-groups and, as consequence, allows obtaining good output results should be more extensively studied. Performance measures that allow to judge the achieved output and to automatically criticize the system performance are another important issue [10].

Abstraction of cases to learn domain theory would allow better understanding the behavior of many signal processing algorithms that cannot be described by means of standard system theory [16].

The aim of our research is to develop methods allowing learning a model for the desired task from cases without heavy human interaction (see Fig. 1). The specific emphasis of this work is on development of a methodology for finding the right image description for the case that groups similar images in terms of parameters within the same group and maps the case to the right parameters in question.

We recently investigated [11] the theoretical and implementation aspects of the watershed transformation, which allowed us to draw conclusions for suitable image descriptions. Four different image descriptions have been considered in [11], respectively based on: statistical and texture features; marginal distributions of columns, rows, and diagonals; similarity between regional minima; and central moments. In this paper we study promising descriptors based on statistical and texture features in which we changed the influence of the difference features.

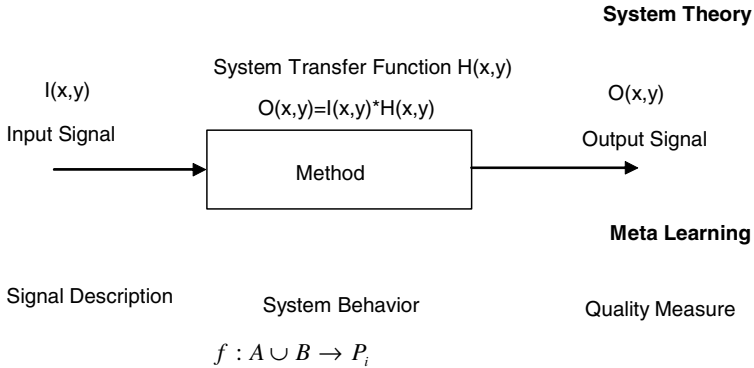


Fig. 1. Problem description in modeling

The basic idea for meta-learning with case-based reasoning for image processing tasks was introduced in this section. In Section 2, we briefly describe the segmentation based on watershed transformation and on the use of Case-Based Reasoning, and point out that the behavior of the watershed transformation may influence the result of the segmentation when the same image is processed after rotation or scaling. The test images and the corresponding best segmentation parameters are given in Section 3, where also the problems concerning the evaluation of the results are briefly addressed. The results of the descriptor based on weighted statistical and texture feature are described in Section 4. Results are given in Section 5 and the conclusions are presented in Section 6.

2 Watershed Transformation Based on CBR

Many segmentation algorithms based on watershed transformation have been developed (for a survey, see, e.g., [13]). As for the basic watershed transform algorithm, we implemented it according to the approach suggested by Vincent and Soille [15]. For the oversegmentation reduction process we followed the approach by Frucci [2] and Frucci and Sanniti di Baja [5] and transformed the crisp rules into a CBR-approach [3, 4] that made the whole process more flexible.

Another way to reduce the over-segmentation of the conventional watershed algorithm is by using the marker controlled watershed algorithm. The marker controlled watershed algorithm has the disadvantage that one has to compute the foreground and background objects before the algorithm start. This can be done by a human or by preprocessing. For example, Jobin Christ and Parvathi [6] use k -means clustering to compute the regions from which the marked controlled watershed algorithms started.

Zanaty and Afifi [17] use an algorithm based on seed region growing and image entropy to reduce the oversegmentation in magnetic resonance images (MRIs).

The conventional watershed algorithm usually produces over-segmentation, which is reduced by combining an iterative computation of the watershed transform with processes called digging and flooding [2]. Flooding merges a non-significant basin to

adjacent basins by suitably increasing the gray level of the bottom of the non significant basin. This way, when the watershed transform is newly computed, no regional minimum is found within the non-significant basin. Digging merges a partially significant basin to specific adjacent regions. The basin A , regarded as partially non-significant, is merged with an adjacent basin B , by digging a canal into the watershed line separating A and B , to prevent that a regional minimum is found in A when applying again the watershed transformation. As a result of merging, the number of local minima found at each iteration diminishes. Flooding and digging and watershed computation are iterated until only significant basins are left.

In order to determine whether a basin X has to be merged to a basin Y , Frucci et al. [3] perform the following check:

$$\frac{1}{2} \left(a \cdot \frac{SA_{XY}}{At} + b \cdot \frac{D_{XY}}{Dt} \right) \geq T, \text{ with } a, b, T \geq 0 \quad (2)$$

where At and Dt are threshold values (for their automatic computation see [2]). The values a and b are constants setting different influences to the parameters. The constant T is a threshold for the rule that tells us when to accept the merging. SA_{XY} is a similarity parameter that is the difference between the regional minima of X and Y , and D_{XY} is the relative depth of the basin X with respect to the adjacent basin Y (for more details see [2, 3]).

We are interested in analyzing the image properties in order to detect the proper values for the constants a , b and T . The constants a and b control the influence of the similarity parameter and the relative depth. T can be regarded as a threshold. Different test shows, that if T is 0 and $a, b \geq 0$, then we obviously get the same segmentation like the one produced by the conventional Vincent-Soille algorithm [15]. If we choose in our CBR-based watershed algorithm the parameters $a, b = 2$ and $T = 1$ we obtain often similar results as those obtained by using the crisp rule-based algorithm described in [2].

It is essential to study the real behavior of the used Watershed Algorithm and implementation in order to build a general image segmentation model. The behavioral aspects of the Vincent-Soille algorithm [15] for the watershed transformation that influence the segmentation results have been studied in detail in [11]. In summary, we can say that some of the Watershed Algorithms are not invariant for image rotation and scaling due to their dependence on the order of visiting the pixels. The detailed theoretical description and demonstration of this fact can be found in [11]. Since the basic Watershed Algorithm is rotation and scaling dependent, the Watershed Algorithm based on Case-Based Reasoning is also dependent on rotation and scaling. The best values of a , b and T can be different for two images after scaling or rotation. Other problems are likely to arise, since the watershed lines may be missing or may be too thick. Thus, we have to find an image description that can take into account the behavior of the algorithm.

Making a compromise between computational cost and quality of the obtained segmentation results, we have finally opted for the Vincent-Soille algorithm as basis for CBR-based Watershed Algorithm. In future work, we will carry out further tests on the different behavior of the basic watershed algorithms. The hope is that the choice of the basic algorithm can be included as parameter into a CBR-based Watershed Algorithm.

3 Test Images and Parameters for Watershed Segmentation

For demonstration of our study we use nine running image examples of different types (biological images, faces and animals, see Fig. 2a-j). The images {neu1,neu2, neu3,neu4,neu4_r180} shown in Fig. 2 c-h belong to the same class, except that neu4_r180 is the 180 degree rotated image of neu4 and neu4 is at slightly larger scale with respect to the other four images.

The parameters for the watershed segmentation have been obtained by running a number of times the Watershed Algorithm based on Case-Based Reasoning [11], adjusting the parameters until the result has the best segmentation quality. For example, in Fig. 3 three different segmentation results are shown for the image gan128, obtained with different selections of the parameters. N=67 basins are obtained in the best segmentation.

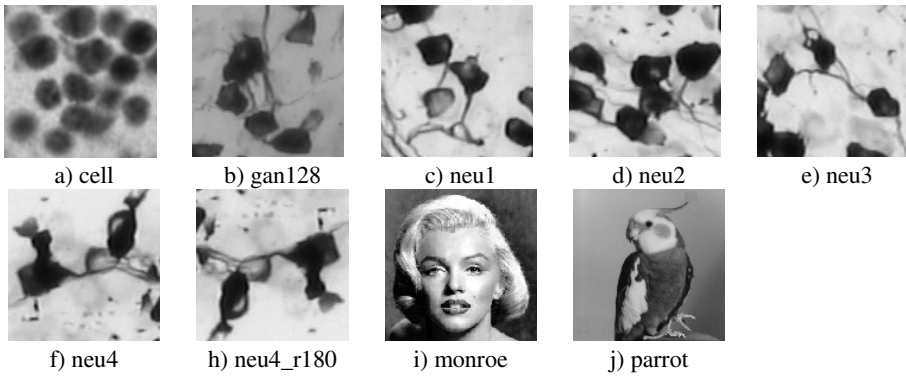


Fig. 2. Images used for the study

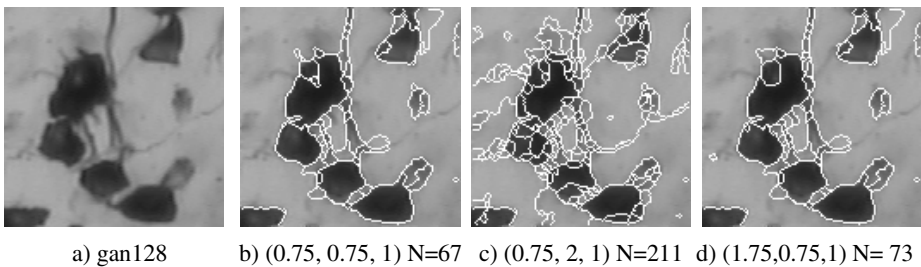


Fig. 3. Influence of the selected parameter-set to the segmentation result for the image gan128

The parameters corresponding to the best segmentation results for the nine test images in Fig. 2 are shown in Table 1. The parameter were obtained by running of the Watershed Algorithm based on Case-Based Reasoning with different parameter values and by evaluation of the obtained images by an expert.

Note, that in Table 1 only two of the neuron images have the same parameters (neu1 and neu3). Actually, neu4 contains cells slightly larger than those in the other neuron images and cannot be segmented by using the same parameters of the other images, because the Vincent-Soille algorithm is not invariant with respect to scaling. Moreover, image neu2 differs from neu1 and neu3 by the presence of a larger number of cells, which justifies a different set of parameters.

Table 1. Segmentation parameters a , b , and T for the test images shown in Fig. 2

Image	a	b	T
monroe	0,5	2	1
gan128	0,75	0,75	1
parrot	0,75	2	1
cell	1	1	1
neu1	2	1	1
neu2	0,25	1	1
neu3	2	1	1
neu4	0,75	2	1
neu4_r180	1	0,5	1

A problem related to the determination of the segmentation parameters for the CBR-based watershed algorithm is how to judge the best segmentation quality. Evaluation done by humans is subjective and can result in differently segmented versions of the same input image. For example, two humans asked to select the preferred segmentation between three possible results (shown in Fig. 3b-d) gave opposite answers (see Fig. 3b-c). In turn, the best segmentation automatically obtained by comparing the watershed lines of the segmented image to the edge image, generated by the Prewitt-Operator [10], of the input based on similarity procedure described in [3] is shown in Fig. 3d. An automatic evaluation of the segmentation results is really necessary, but even an automatic evaluation may have some weakness.

The automatically computed binarized gradient image is only an auxiliary method for getting the true segmented image (gold standard) to which we have to compare. The gold standard could be obtained by manually labeling the regions of the input image, which is not appropriate.

4 Elicitation of Image Descriptions and Assessment of Similarity for Watershed Transform

The aim of image description is to find out among a set of images the group of images that needs the same processing parameters to achieve the best segmentation results. To give an example, the images neu1 and neu3 should be grouped together in one group based on the best parameters a , b , and T (see Table 1) and the images neu4 and parrot should be grouped together in another group.

We consider different image descriptions in our study [11] that should allow us to group images based on the image features and by doing this to learn a model for image segmentation by samples.

Cases are composed normally of

- non-image information
- features specifying image characteristics, and
- parameters for solution (image segmentation parameters).

Non-image information is different depending on the application. In our study we use images from different domains, such as biological images, faces and animals. Our aim is to describe image similarity only by general image features. Hence our cases are composed of

- features specifying image characteristics, and
- parameters a, b, T for segmentation.

Images, which are classified as being similar based on the image features, should be segmented with the same parameters of the segmentation algorithm, which should produce the best segmentation for any of them.

To understand which features are to be used, we resort to hierarchical clustering. This gives us a graphical representation of the different image groups. Single linkage is used to show outlier while the distance between two classes is defined as the minimal.

The question is: What are the right image features that allow us to map the images to the proper image segmentation parameters for the watershed transformation?

The image description should reflect the behavioral approach of the watershed transformation with respect to the particular image characteristics. Therefore, we studied the theoretical details and the implementation limits of the watershed transformation in [11] to get insights into this question. Based on this work we decided to test four image descriptions based on:

- Statistical and Texture Features,
- Marginal Distribution for Columns, Rows, and Diagonals,
- Similarity between the Regional Minima, and
- Central Moments.

The details of this study are given in [11]. The most promising description is based on Statistical and Texture Features. This description is used for the study in this paper.

4.1 Image Description Based on Statistical and Texture Features

According to Perner [7], who used this description for meta-learning the parameters for a CBR-based image segmentation model, we used statistical features (like centroid, energy, entropy, kurtosis, mean, skewness, variance and variation coefficient) and textures feature (energy, correlation, homogeneity, contrast) for case description. The input image is the gradient image of the original image, since the watershed transformation works on that image. First results on this image description are reported in

Frucci et al. [3]. The texture features have been chosen to describe the particular distribution of the regional minima in an image, while the statistical features describe the signal characteristics.

Like in Perner [7], the distance between two images A and B is computed by:

$$dist_{AB} = \frac{1}{k} \sum_{i=1}^k \omega_i \left| \frac{C_{iA} - C_{iB}}{C_{i\max} - C_{i\min}} \right|, \quad (3)$$

where k is the number of features in the data base, $C_{i\max}$ and $C_{i\min}$ are the maximum and minimum value of the i th feature for all images in the data base, C_{iA} (C_{iB}) is the value of the i th feature for image A (B) and the ω_i are weights.

The following equation holds for the weights:

$$\sum_{i=1}^k \omega_i = 1. \quad (4)$$

4.2 Equal Weighting of the Statistical and Texture Features

To get an equal weighting of the statistical and texture features we set the weights in formula (3):

$$\omega_i = 1/k \quad \forall i \in [1, \dots, k] \quad (5)$$

The results are reported in Fig. 4.

If we virtually cut the dendrogram by the cophenetic similarity of 0.0043 we obtain the groups $G1=\{\text{neu4}, \text{neu4_r180}, \text{neu1}, \text{neu3}\}$, $G2=\{\text{neu2}\}$, $G3=\{\text{parrot}\}$, $G4=\{\text{gan128}\}$, $G5=\{\text{monroe}\}$, and $G6=\{\text{cell}\}$.

The images neu4 and the image neu4_r180 (which is the 180 degree rotated version of neu4) are grouped into the same group, although they have completely different segmentation parameters. We obtained the best result for neu4 with the parameter-set $a = 0.75$, $b = 2$ and $T = 1$, while for neu4_r180 the best segmentation was obtained with the parameter-set $a = 1$, $b = 0.5$ and $T = 1$. By using the latter parameter-set for neu4 , we would get an undersegmented result. Overall we observe that not all images having the same image segmentation parameters are grouped into one group such as neu4 and parrot .

4.3 Unequal Weighting of the Statistical and Texture Feature

To sort out rotated images from the group that includes the un-rotated images we have to give more emphasis to the feature centroid, because this features is only one which is not invariant for rotations. Therefore, we divide the image features set into three groups: texture features, centroid, and the remaining statistical features. In the following we tested different weighting of these three groups. Inside the three feature groups the features are equal weighted.

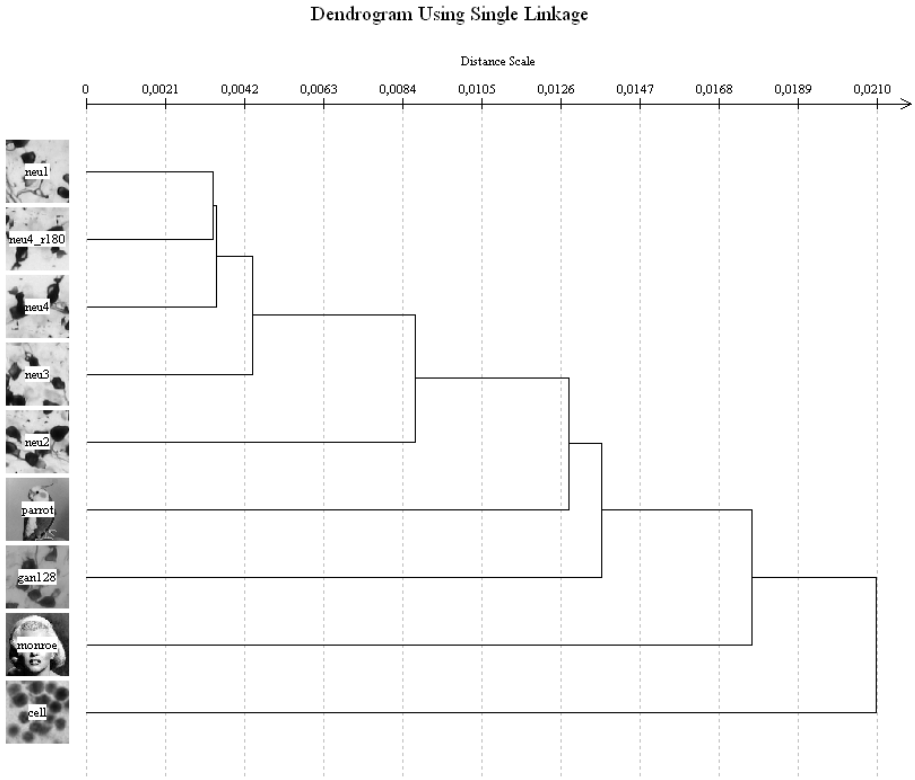


Fig. 4. Dendrogram for image description based on texture and statistical features

First we test, if each group gets a total weight ω_g of $1/3$ ($1 \leq g \leq 3$). The weights ω_{gi} inside of each group, are computed as follow

$$\omega_g = \sum_{i=1}^l \omega_{gi} = \sum_{i=1}^l \frac{1}{3 * l_g} = \frac{1}{3}, \tag{6}$$

where l_g is the number of features in the group.

In the resulting dendrogram the images neu4 and neu4_r180 are clustered in different groups (see Fig. 5).

If we virtually cut the dendrogram by a cophenetic similarity value of 0,0057 then we obtain the following groups $G1=\{neu1, neu3\}$, $G2=\{neu4_r180\}$, $G3=\{neu2\}$, $G4=\{neu4\}$, $G5=\{gan128\}$, $G6=\{parrot\}$, $G7=\{cell\}$, and $G8=\{monroe\}$. Except for the images neu4 and parrot, these groups seem to reflect better the relationship between the image description and the parameter-set.

In the following we demonstrate that a small preference of the weighting for the group features “centroid” separation of the images neu4 and neu4_r180.

Therefore let ω_2 be the total weight of the feature group “centroid”. The weights ω_{g_i} inside of each group are computed as follow

$$\omega_{g_i} = \begin{cases} \frac{1}{l_1+z l_2+l_3} & \text{if } g = \{1, 3\} \\ \frac{z}{l_1+z l_2+l_3} & \text{if } g = 2 \end{cases} \quad \text{with } z \in \mathbb{N} \quad (7)$$

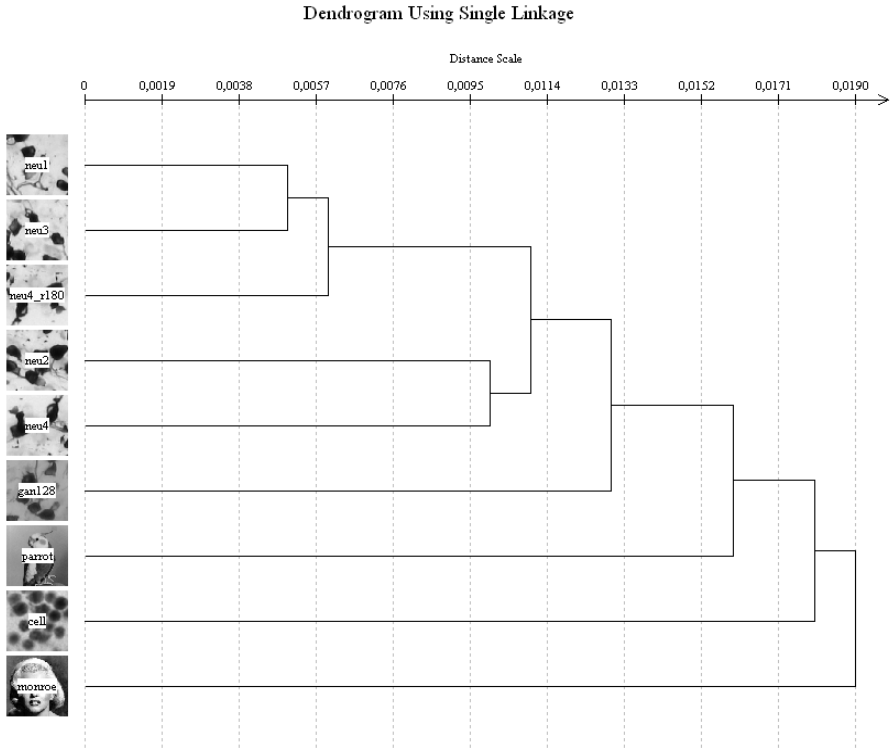


Fig. 5. Dendrogram for image description based on weighted texture, centroid and statistical features using formula (6)

The results for $z = 2$ and $z = 3$ are shown in Fig. 6 and Fig. 7. Fig. 5, Fig 6 and Fig. 7 demonstrate that when we increase the preference of the group “centroid” the two images become more and more dissimilar.

If we virtually cut the dendrogram by a cophenetic similarity value of 0.0042 (Fig. 6) respectively 0.061 (Fig. 7), then we obtain the following groups $G1=\{neu1, neu3, neu4_r180\}$, $G2=\{neu4\}$, $G3=\{neu2\}$, $G4=\{gan128\}$, $G5=\{parrot\}$, $G6=\{cell\}$, and $G7=\{monroe\}$. But by using the parameter-set $a = 2$, $b = 1$ and $T = 1$ for neu4_r180, we would get an undersegmented result for the image neu4_r180.

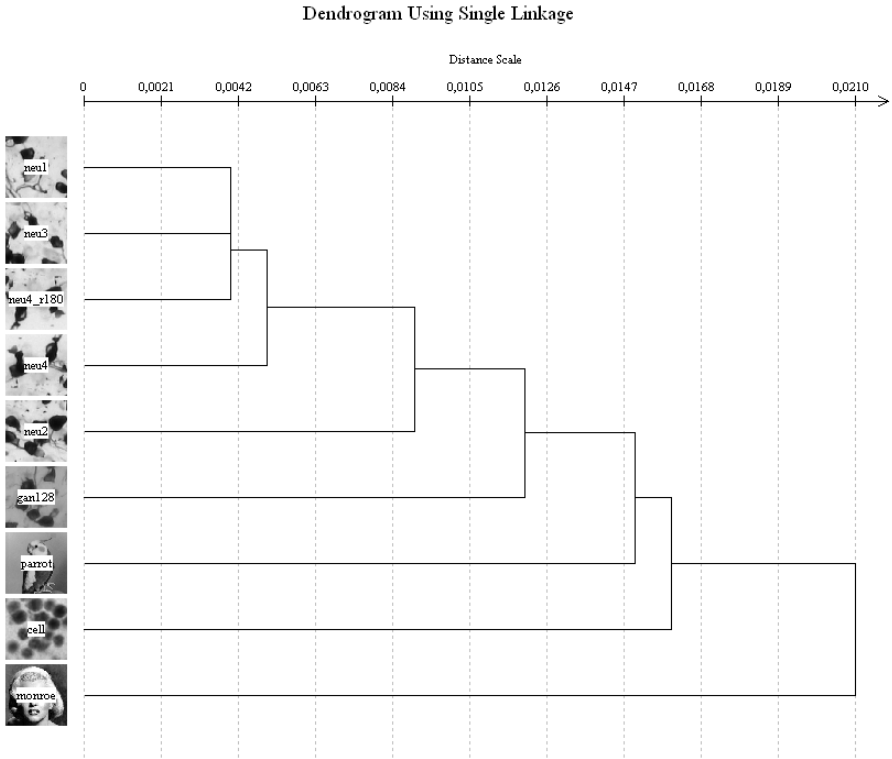


Fig. 6. Dendrogram for image description based on weighted texture, centroid and statistical features with weighting after formula $z=2$

It can be problematic that we need a high preference of the group “centroid”. In that case it can happen that during other tests we need for other features a higher weight in order to separate other images. Depending on the feature weighting, we can obtain results as the ones in Fig. 5, but we can also get results as in Fig. 4 where the rotated images are not separated.

The proper weighting of the features can improve the grouping of the images. In future work we will work on automatic determination of the proper weighting.

5 Discussion

Out of the four image descriptions which we studied in [11] based on statistical and texture features; marginal distributions of columns, rows, and diagonals; similarity between regional minima; and central moments, the descriptions working better for the watershed transformation are those based on statistical and texture features (STDescribe) and on central moments. During this study we have considered the descriptions based on unequally weighted statistical and texture features.

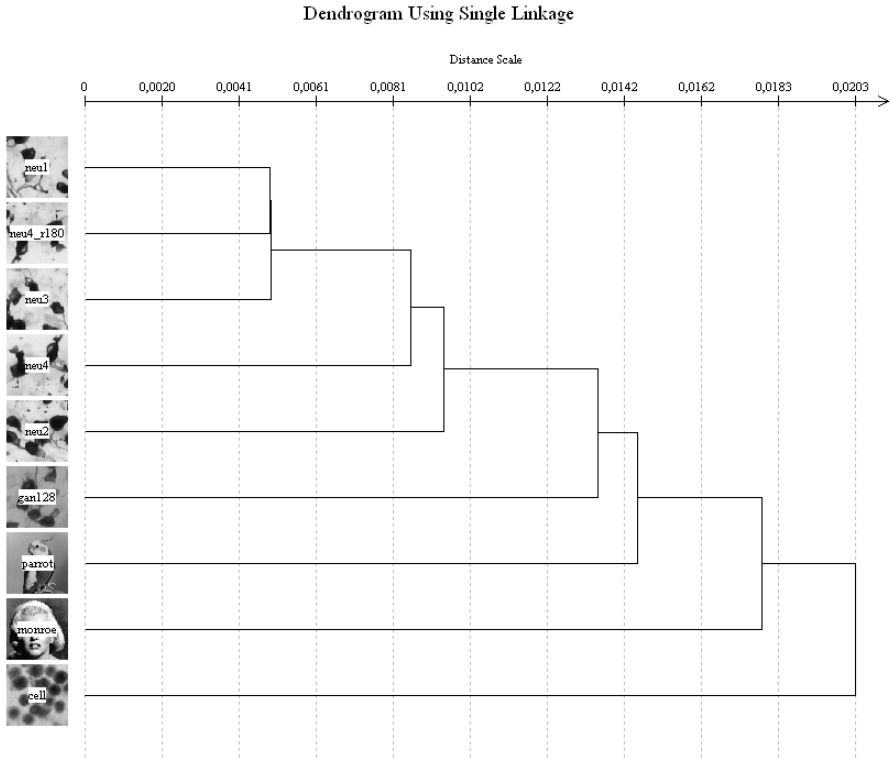


Fig. 7. Dendrogram for image description based on weighted texture, centroid and statistical features with weighting after formula $z=3$

In contrast to equally weighting of the statistical and texture features, we are by unequally weighting of the feature variable for different kinds of images because we can adjust the weighting on the feature to the situation and can separate rotated and scaled images. We demonstrate that a small preference of the weighting for the group features “centroid” is enough to separate rotated images. The best result we get for our test image using weighting of the three feature groups (weighted STDescribe). The obtained groups for the two descriptions are the following:

STDescribe

$G1=\{neu4, neu4_r180, neu1, neu3\}, G2=\{neu2\}, G3=\{parrot\}, G4=\{gan128\}, G5=\{monroe\},$ and $G6=\{cell\}$

weighted STDescribe

$G1=\{neu1, neu3\}, G2=\{neu4_r180\}, G3=\{neu2\}, G4=\{neu4\}, G5=\{gan128\}, G6=\{parrot\}, G7=\{cell\},$ and $G8=\{monroe\}$

The groups obtained by means of the weighted STDescript seem to reflect better the relationship between the image characteristics and the segmentation parameters than the groups obtained by the other description. The computation time of the image descriptions is more or less the same. Thus, we can say that the weighted statistical and texture features description is the best image description that we have found so far during our study.

6 Conclusions

The aim of our work was to find an image description based on weighted statistical and texture features that characterizes each particular image with respect to the behavior of the image segmentation method. The image description should allow retrieving the best possible segmentation parameters. We first studied the theoretical and implementation aspects of the watershed transformation in order to draw conclusions for the image description.

The watershed transformation produces different results if the image is rotated or rescaled. The particular implementation of the algorithm puts constraints on the behavior of the algorithm. As result of our study we concluded that we need an image description that describes the distribution of the regional minima and that is not invariant against rotation and scaling.

In previous studies we researched four different image descriptions, respectively based on: statistical and texture features; marginal distributions of columns, rows, and diagonals; similarity of regional minima; and central moments.

Two of the above four image descriptions did not lead to any success. The best and most variable image description is the description based on weighted statistical and texture features. This image description seems to well represent the relationship between the image characteristics of the particular image and the segmentation parameters. Cases having the same segmentation parameters could be grouped into the same group using an image description based on weighted statistical and texture features. This will make possible the generalization over these groups of cases, which is expected to lead to a complete image segmentation model. In future work we plan to study, how we can automatically choose the proper weight values of the statistical and texture features for different kinds of images.

References

1. Attig, A., Perner, P.: A study on the case image description for learning the model of the watershed segmentation. *Transactions on Case-Based Reasoning* 2(1), 41–54 (2009)
2. Frucci, M.: Oversegmentation reduction by flooding regions and digging watershed lines. *International Journal of Pattern Recognition and Artificial Intelligence* 20(1), 15–38 (2006)
3. Frucci, M., Perner, P., Sanniti di Baja, G.: Case-based reasoning for image segmentation by watershed transformation. In: Perner, P. (ed.) *Case-Based Reasoning on Images and Signals*. SCI, vol. 73, pp. 319–353. Springer, Berlin (2008)

4. Frucci, M., Perner, P., Sanniti di Baja, G.: Case-based reasoning for image segmentation. *International Journal of Pattern Recognition and Artificial Intelligence* 22(5), 1–14 (2008)
5. Frucci, M., di Baja, G.S.: A New Algorithm for Image Segmentation via Watershed Transformation. In: Maino, G., Foresti, G.L. (eds.) *ICIAP 2011, Part II*. LNCS, vol. 6979, pp. 168–177. Springer, Heidelberg (2011)
6. Jobin Christ, M.C., Parvathi, R.M.S.: Segmentation of medical image using K-means clustering and marker controlled watershed algorithm. *European Journal of Scientific Research* 71(2), 190–194 (2012)
7. Perner, P.: An architecture for a CBR image segmentation system. *Engineering Applications of Artificial Intelligence* 12(6), 749–759 (1999)
8. Perner, P.: Why Case-Based Reasoning Is Attractive for Image Interpretation. In: Aha, D.W., Watson, I. (eds.) *ICCBR 2001*. LNCS (LNAI), vol. 2080, pp. 27–44. Springer, Heidelberg (2001)
9. Perner, P.: Case-based reasoning for image analysis and interpretation. In: Chen, C., Wang, P.S.P. (eds.) *Handbook on Pattern Recognition and Computer Vision*, 3rd edn., pp. 95–114. World Scientific Publisher (2005)
10. Perner, P.: Case-based reasoning and the statistical challenges. *Journal Quality and Reliability Engineering International* 24(6), 705–720 (2008)
11. Attig, A., Perner, P.: Meta-learning for Image Processing Based on Case-Based Reasoning. In: Bichindaritz, I., Vaidya, S., Jain, A., Jain, L.C. (eds.) *Computational Intelligence in Healthcare 4*. SCI, vol. 309, pp. 229–264. Springer, Heidelberg (2010)
12. Prewitt, J.M.S.: Object enhancement and extraction. In: Lipkin, B.S., Rosenfeld, A. (eds.) *Picture Processing and Psychopictorics*, pp. 75–149. Academic Press, New York (1970)
13. Roerdink, J.B.T.M., Meijster, A.: The watershed transform: definitions, algorithms and parallelization strategies. *Fundamenta Informaticae* 41, 187–228 (2001)
14. Soares, C., Brazdil, P.B., Kuba, P.: A meta-learning method to select the kernel width in support vector regression. *Machine Learning* 54, 195–209 (2004)
15. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Patt. Anal. Mach. Intell.* 13(6), 583–598 (1991)
16. Wunsch, G.: *Systemtheorie*. Akademische Verlagsgesellschaft, Leipzig (1975)
17. Zanuty, E.A., Afifi, A.: A watershed approach for improving medical image segmentation. *Computer Methods in Biomechanics and Biomedical Engineering* (forthcoming paper, 2012)

Fast Level-Wise Convolution

Damien Gonzalez¹, Rémy Malgouyres¹,
Henri-Alex Esbelin¹, and Chafik Samir²

¹ Clermont-Université, LIMOS, Complexe des Cézeaux, 63172 Aubière, France
{damien.gonzalez,remy.malgouyres}@u-clermont1.fr,

Alex.Esbelin@univ-bpclermont.fr

² Clermont-Université, ISIT, Complexe des Cézeaux, 63172 Aubière, France
chafik.samir@u-clermont1.fr

Abstract. Estimation of differentials of discrete signals is almost mandatory in digital segmentation. We present a new fast method based on convolutions by a mask with a logarithmic number of constant layers. Then we compare it to other multigrid convergent methods in the field such as the Binomial Convolution, the Digital Straight Segment Tangent Estimator, and the Taylor Polynomial Fitting. Our convolution method's main advantage is its complexity of $O(2n \cdot \log_2(m))$, which makes it competitive to the convolution by Fast Fourier Transform (FFT) latest implementation. In the experimental part, we also tested the precision of the first order derivative estimation, its resistance to noise and its convergence rate.

Keywords: Differential estimator, Discrete differential operator, Fast convolution, Noise resistant, Sparse differential operator, FFT.

1 Introduction

Digital segmentation algorithms such as active contour model often use signal parameters as energy. Estimation of differentials is almost mandatory for most of them as they use regularization terms like the Snake Algorithm [9] and other deformable models [15]. Previous work is divided into two categories: non-convolutional methods and convolutional methods. The Digital Straight Segment (DSS) Tangent Estimator [12,2] extracts maximal DSS and computes their tangents. One of the advantages of this method is its ability to detect corners. Its convergence rate is $O(\frac{1}{3})$. The Taylor Polynomial Approximation [17] fits the values of a digital function by a polynomial. It introduces a roughness parameter to relax the function values in an interval. It has a bounded maximal error for the k^{th} derivative and a resolution h of $O(h^{\frac{1}{1+k}})$. Its convergence rate is $O(\frac{1}{1+k})$. The Binomial Convolution [16,3] approximates differentials with the finite differences after applying a digital version of the scale space [13] function smoothing, using integer-only binomial coefficients as the convolution mask. It is noise resistant, has a convergence rate of $O((\frac{2}{3})^k)$ and a complexity of $O(n*m)$ with n the size of the image and m the one of the convolution mask. We introduce

a new convolution method mainly focusing on complexity, similar to the convolution by FFT [1] latest implementation [5], however based on a much simpler approach. Due to its nature, it is more resistant to noise than the first version of the DSS [19]. We were not able to compare it to the latest version of DSS [11,10], however, due to lack of time and data. For the first order derivative, our method is more precise than the Taylor Polynomial Fitting and is faster than the binomial convolution and FFT with a complexity of $O(2n \cdot \log_2(m))$.

The paper is organized as follows. Section 1 presents convolutions. In the following section we propose two compatible kernels based on the Gaussian function and the binomial coefficients. Then we describe a sparse operator able to quickly estimate the differential for a single pixel and to reduce the overall complexity of the smoothing and derivative process. In experimental results we show a comparison of the computational cost, estimated convergence rate, noise resistance, and precision.

2 Level-Wise Convolution

When dealing with derivatives estimation, one of the most classical method is to use finite differences. Although effective in continuous geometry, it cannot be applied as such to discrete images, because derivative values would be limited to integers. A solution is to average each pixel of the image with its neighborhood, a process called smoothing. This mathematical operation is known as the convolution product of a function in the integers interval $f : [0, n] \rightarrow [0, n]$ (the image to be convolved) and a function $H : [0, n] \rightarrow [0, n]$ (the averaging kernel). Using the Gaussian function as a kernel is the standard approach in this field, as described by Lindeberg in the scale space theory [13]. The resulting image can then serve to compute differentials, using finite differences with a convolution by a differential operator δ as a kernel.

Figure 1 shows an example of the convolution of a digital function 1, 2, 2, 4, 5 by a binomial coefficients kernel 1, 2, 1. To preserve the image scale, each value has to be divided by the mass (or weight) of the kernel (the sum of all its values), in this case $n = 2$. The weight being the mass of the mask $W = 2^n = 4$ in the given example.

Then we convolve the smoothed image by a differential operator to obtain the derivatives (first order with this operator). There are three kinds of first order derivative operator, the centered one as in the example, the backward difference ($f(0) - f(-1)$) and the forward difference ($f(1) - f(0)$). The choice of the derivative operator can slightly shift the values, if the symmetry with the smoothing operator is not respected. For a centered kernel (odd size) we use the centered operator. An even size kernel can be convolved with left or right shift (we use the opposite smoothing kernel shift). Out of the range of the discretization there is no information about the function values. The convolution loses precision when those pixels are required. This is known as the border problem. For the example of Fig. 1 the unknown values are set to 0. In the experimental part we only use functions for which we know those values.

The major drawback of this method in the discrete paradigm is its complexity of $O(n * m)$, with n being the size of the image and m the size of the kernel.

$$(f * H)(x) = W \cdot \sum_{i=-k}^{+k} f(x - i) \cdot H(i) \tag{1}$$

Equation (1) shows the discrete convolution product of an image f with a kernel H for the pixel x of f ; W represent the weight of the kernel. Looking at the right part of Fig. 1 the kernel can be viewed in a multilevel way. The discrete convolution product can be rewritten to apply on one level of the kernel at a time, as shown in Eq. (2), the convolution of the function f with the kernel H only for the pixel n . The first loop $\sum_{j=0}^k$ iterates through all levels and the second one $\sum_{i=-k+j}^{k-j}$ through the whole current level.

The complexity has changed to $O(m * \sum_{i=0}^{\lfloor k/2 \rfloor} k - i)$, but Eq. (2) only concerns one pixel and since each level of the kernel has the same value, we only need to convolve for the first pixel as indicated in Eq. (3) and Fig. 2. The left part is the convolution of pixel n and the right part is the convolution of pixel $n + 1$ using the previous result. Since each level of the kernel has the same value (Fig. 1) we only need to subtract the product of H_{-m} and F_{n-m-1} and to add the product H_m and F_{n+m} to the convolution of pixel F_n to obtain the result.

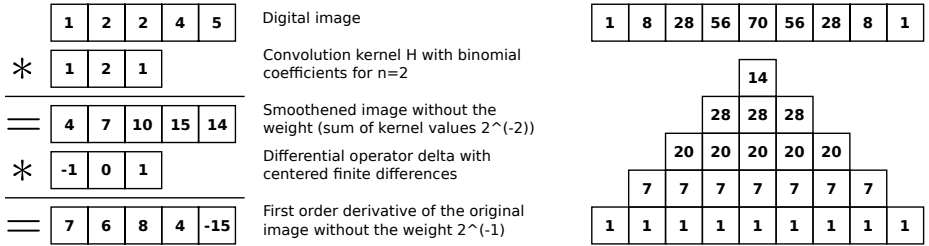


Fig. 1. Left. First the detailed process of convolving a digital image with a binomial coefficients kernel, second the convolution of the smoothed image with the central differential operator of central finite differences. Right. Level view of the binomial kernel for $\binom{n}{k}$ with $n = 8$ and k the line index.

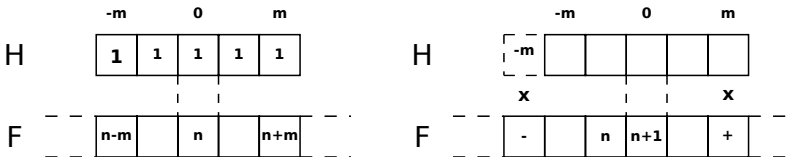


Fig. 2. Convolution of image f with kernel H of size $2m + 1$ centered at 0

$$(f * H)(n) = \sum_{j=0}^k \sum_{i=-k+j}^{k-j} f(n+i) \cdot (H(-k+j) - H(-k+j-1)) \quad (2)$$

$$(f * H)(n+1) = (f * H)(n) - H_{-m} \cdot f_{n-m-1} + H_m \cdot f_{n+m} \quad (3)$$

2.1 Complexity

The resulting complexity depends on the number of levels of the kernel. We only use kernels with \log_2 bounded number of levels. We have a $O(2n * \log_2(m))$ complexity which is smaller than the binomials convolution of $O(n * m)$ in [16,3] and theoretically slightly better to the latest complexity of the FTT of $O(\frac{39}{4}N * \log_2(N))$ in [7,14,8]. In addition, the FFT has to be done two times, once for the image and once for the kernel and in addition their multiplication in Fourier space must be performed. Using other types of boundaries than the logarithmic ones can increase or decrease the complexity. Thus, it could be interesting to have a kernel with a fixed number of levels.

2.2 Kernel Compatibility and Extension to Higher Dimensions

In order for a kernel to be compatible with this method, there must exist a level-wise decomposition of it and it must be symmetric to avoid data shift. There is a \log_2 bounded number of levels in order to have the same complexity. The convolution should work in higher dimensions using the tensor product of one dimension kernels. The use of n dimension kernels is possible if they are separable in 1 dimension elements.

3 Kernels Presentation

We introduce two discrete kernels of low complexity. They are level-wise versions of two known kernels, namely binomial coefficient kernel (which is the fastest converging method for the first three order derivative) and the Gaussian kernel (which is the fastest in terms of computational times).

3.1 Pseudo-gaussian Kernel

To create this kernel, we started from the continuous Gauss formula Γ (4) with real constant parameters $\alpha = \frac{1}{\sigma\sqrt{2\pi}}$, σ is the standard deviation and λ the expectation. When $\lambda = 0$ the function is centered in 0 on the x axis. We have created a rough kernel with its level number bounded by the \log_2 function H_Γ (5). It is always centered in 0 with α representing the weight such as the integral of the kernel is equal to 1 in order not to scale the image after convolution. Parameter Λ represents the standard deviation and γ controls the number of levels.

$$\Gamma = \alpha e^{-\frac{(x-\lambda)^2}{2\sigma^2}} \quad \text{or} \quad \alpha e^{-\lambda_2 x^2} \quad \text{for } \lambda = 0 \quad (4)$$

$$H_\Gamma = \alpha 2^{\mu^2 \lceil \gamma * \log_2(i) \rceil} \quad \text{where} \quad \mu = \Lambda^{1/\gamma} \quad (5)$$

3.2 Building Complexity

Since we cannot predict for which value of i the level change will occur, we have to compute the values for the whole kernel size $O(m)$ with m being the size of the kernel. The building process can be speed up by using the left arithmetic shift to compute the powers of 2.

3.3 Pseudo-binomial Kernel

We used the binomial coefficients as a basis for this kernel since it is a good discretization of the Gaussian function Γ . It is a level-wise kernel and the values of the levels are the sum of the binomial coefficients between two boundaries controlled by the floor function. Eqs. (6) and (7) calculate the pseudo-binomial kernel B . The integers parameters are: m the size of the kernel, B_0 the centered value, B_k the other values for $1 < k < n$, n the number of the level and $s(k)$ is the signature of the value. Figure 3 illustrates the relation between the two kernels. At the top, H is represented through different levels with sizes equal to the increasing powers of two. B level values are the average of the values in the corresponding level.

$$B_0 = \frac{1}{2^{2^{\alpha+1}-2}} \binom{2^{\alpha+1}-2}{2^\alpha-1} \quad \text{for } n \neq 0 \quad (6)$$

$$B_k = \frac{1}{2^{2^{\alpha+1}-2} + \lceil \log_2(|k|) \rceil} \left(\sum_{i=2^{\lceil \log_2(|k|) \rceil}}^{i=2^{1+\lceil \log_2(|k|) \rceil}-1} \binom{2^{\alpha+1}-2}{2^\alpha-1+s(k)i} \right) \quad (7)$$

$$\text{where} \quad \begin{cases} s(k) = +1 & \text{if } n > 0 \\ s(k) = -1 & \text{if } n < 0 \end{cases} \quad \text{and} \quad m = 2^{\alpha+1} - 2$$

3.4 Building Complexity

To minimize the complexity we use the Pascal triangle building method to compute the binomial coefficients. For memory management, we use the upper bound $\frac{4^n}{8n\sqrt{\pi n}}$ to allocate our triangle's line.

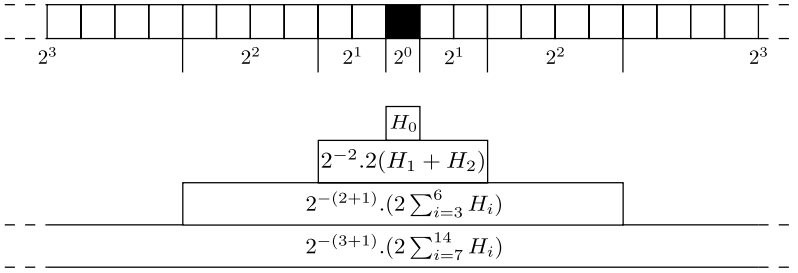


Fig. 3. Top. Odd length binomial kernel H with a center marked by a black square. Bottom. Pseudo-binomial kernel B . The lower line represents power of 2 coefficients determining the level size.

4 Sparse Differential Operator

Up to now, our methods used finite differences on the image smoothed by a convolution. It is efficient when we want derivatives for the whole image, but it can be costly in terms of memory and computational time when we are only interested in the information on a single pixel. Using the associative property of the convolution product in Eq. (8) with level-wise kernels and the centered finite differences, we can create a differential operator δ_2 with many zeros as shown in Fig. 4. H is the kernel level L , δ is the centered finite differences kernel, and f is the image.

$$\begin{aligned}
 \delta_2(x) &= (H_L * \delta) * (f(x)) \\
 \delta_2(x) &= \sum_{i=0/i \in \mathbb{Z}}^n (H_L * \delta)(i) f(x - i) \\
 \delta_2(x) &= \sum_{i=0/H_L(i) \neq 0}^n (H_L * \delta)(i) f(x - i)
 \end{aligned}
 \tag{8}$$

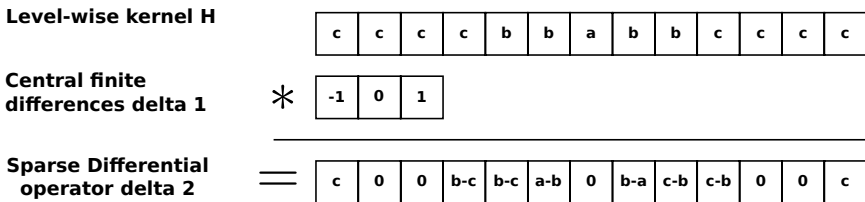


Fig. 4. The sparse differential operator δ_2 is a result of the convolution of the level-wise kernel H with the differential operator of the central finite differences δ_1 (first order derivative)

This reduces the steps for differential computation by making a new differential operator from the smoothed kernel and the centered finite differences kernel. It can be applied to any level-wise kernel and is able to work on a single pixel without having the need to convolve the whole image as it was before.

4.1 Complexity

As shown in Fig. 4, the remaining coefficients appear in the places where the level changes were. The complexity that was for one pixel $O(m)$ with m being the size of the kernel, is now $O(4 \cdot (\text{level_number}))$ for each convoluted pixel.

5 Experiments

The tests of the two kernels have been done with a variety of functions (sine, exponential, arctangent, square, the polynomial $(P(x) = 0.5 * (0.5 * x^3 + 0.5 * t^2 + 0.5 * t - 13) + 2 \cos(5t) + 5)$ and with different function factors to vary the mean curvature). For lack of space we decided to only show one of the best and one of the worse cases. In the implementation of our method we used the GMP library [4], the GSL library [6] and the FFTW library [5]. For the parameters of the pseudo Gaussian kernel we used $\gamma = 10$, $\lambda = m$, $m = \frac{1.6}{h}$ and for the pseudo binomial kernel we used $\alpha = 5 \cdot \log(\frac{h}{2})$ with h the resolution used.

5.1 Precision

In Fig. 5 we show the derivative of the sine function. The comparison of the estimation by the PGM [17] and the level-wise convolution estimation for both kernels. While the pseudo-Gaussian lacks precision in low curvature areas of the sine function, the pseudo-binomial is more precise than the other methods.

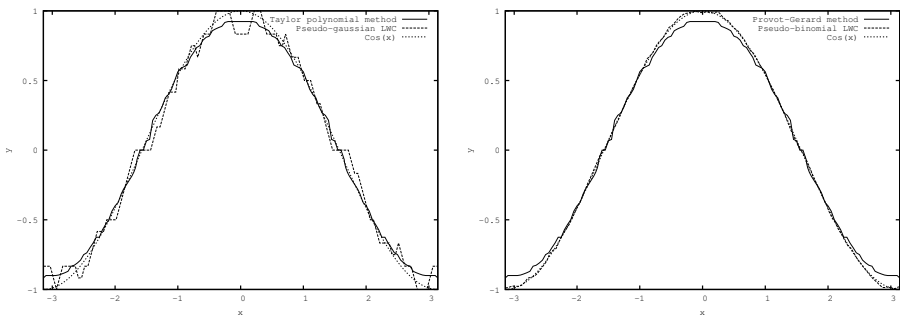


Fig. 5. Sine derivative of first order from its approximation with the Taylor polynomial method and the level-wise convolution method with discretization step of 0.05 : Left. Pseudo Gaussian kernel. Right. Pseudo binomial kernel.

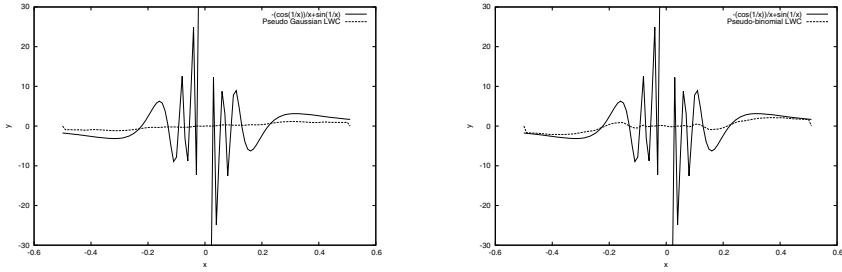


Fig. 6. Derivative of $x.\sin(1/x)$ first order (from discretized function of step 0.01) and its approximation with level-wise convolution method : Left. Pseudo Gaussian kernel. Right. Pseudo binomial kernel.

Figure 6 is the same test on the derivative of $u(x) = x.\sin(\frac{1}{x})$ and does not include data from the PGM. Clearly a hard test, the pseudo-binomial gives better results in the same type of areas.

5.2 Robustness

For the test of Fig. 7 we used a uniform Gaussian noise of $\frac{1}{40}$ the amplitude of the sine function to show its effect on the level-wise convolutions differential approximation. The kernel length is set higher than in the previous tests. The results are close to the expected value and the Gaussian-kernel outperforms the binomial-kernel on the low-curvature areas.

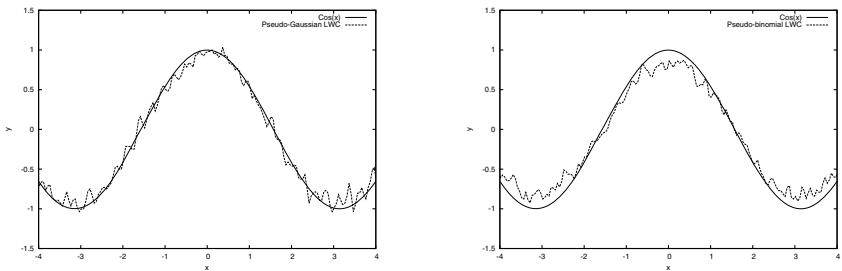


Fig. 7. Approximation of the first derivative of a noisy discretization (step 0.05 and uniform noise of ± 1 pixel) of the sine function. Left. The Pseudo-Gaussian kernel. Right. The Pseudo-binomial kernel.

5.3 Convergence Rate

An estimation of multigrid convergence is achieved experimentally by increasing the discretization step (thus decreasing the discretization error) and the kernel size at the same time. The kernel size and the discretization step are the axis,

the slope formed by the error between theoretical and estimated derivative is the convergence rate. Figure 8 is the experimental convergence rate of the sine function derivative approximation. The image size is the same as the kernel size and we did not use per function optimal parameters. The rates are close to one of the binomial coefficients kernel for this test.

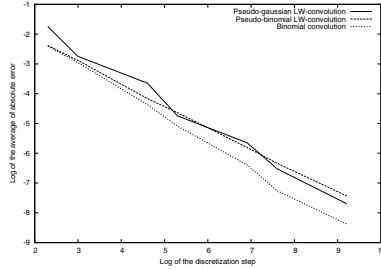


Fig. 8. Experimental convergence rates on the sine function derivatives

5.4 Computational Cost

Being the principal asset of the level-convolution, we tested the convolution speed along with the FFT implementation in the FFTW library [5]. In Fig. 9 we confirm the theoretical complexity. The results are slightly better than the FFTW library which is very much optimized.

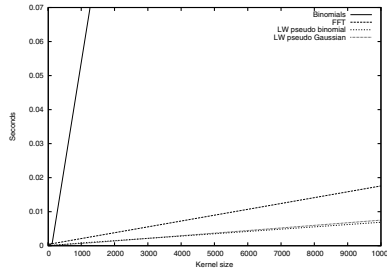


Fig. 9. Convolution computational cost

6 Conclusion

We have presented a new fast digital convolution method with promising experimental results, specifically in terms of computational cost. We also introduced a sparse differential operator able to compute the properties of a single pixel as well as a small group, which can be useful for updating estimation on the fly in segmentation algorithms. In our future work we will focus on the parameters selection and the adaptivity of our kernel using multi-pass convolutions for higher

order differentials. Also we will test higher dimension and higher derivative order. The last goal will be the GPU implementation allowing a multi-pass approach to improve the runtime. In a near future we plan to prove mathematically the multigrid convergence properties of the method and the kernels.

Acknowledgement. The research leading to these results has received funding from the KIDICO project of the French *Agence Nationale de la Recherche* (Grant Agreement ANR-2010-BLAN-0205-02).

References

1. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex fourier series. *Math. Comput.* 19(90), 297–301 (1965)
2. De Vieilleville, F., Lachaud, J.O.: Comparison and improvement of tangent estimators on digital curves. *Pattern Recognition* 42(8), 1693–1707 (2009)
3. Esbelin, H., Malgouyres, R., Cartade, C.: Convergence of binomial-based derivative estimation for 2 noisy discretized curves. *Theoretical Computer Science* 412(36), 4805 (2011)
4. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software* 33(2), 13:1–13:15 (2007)
5. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. *Proceedings of the IEEE* 93(2), 216–231 (2005); Special issue on “Program Generation, Optimization, and Platform Adaptation”
6. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., Rossi, F.: GNU scientific library. Network Theory Ltd. (2002)
7. Heideman, M., Johnson, D., Burrus, C.: Gauss and the history of the fast fourier transform. *IEEE ASSP Magazine* 1(4), 14–21 (1984)
8. Johnson, S., Frigo, M.: A modified split-radix fft with fewer arithmetic operations. *IEEE Transactions on Signal Processing* 55(1), 111–119 (2007)
9. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. *International Journal of Computer Vision* 1(4), 321–331 (1988)
10. Kerautret, B., Lachaud, J.-O.: Curvature estimation along noisy digital contours by approximate global optimization. *Pattern Recognition* 42(10), 2265–2278 (2009)
11. Kerautret, B., Lachaud, J.-O., Naegel, B.: Comparison of Discrete Curvature Estimators and Application to Corner Detection. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) *ISVC 2008, Part I. LNCS*, vol. 5358, pp. 710–719. Springer, Heidelberg (2008)
12. Lachaud, J., Vialard, A., de Vieilleville, F.: Fast, accurate and convergent tangent estimation on digital contours. *Image and Vision Computing* 25(10), 1572–1587 (2007)
13. Lindeberg, T.: *Scale-space theory in computer vision*. Springer (1994)
14. Lundy, T., Van Buskirk, J.: A new matrix approach to real ffts and convolutions of length 2^k . *Computing* 80(1), 23–45 (2007)
15. Ma, Z., Tavares, J., Jorge, R., Mascarenhas, T.: A review of algorithms for medical image segmentation and their applications to the female pelvic cavity. *Computer Methods in Biomechanics and Biomedical Engineering* 13(2), 235–246 (2010)

16. Malgouyres, R., Brunet, F., Fourey, S.: Binomial Convolutions and Derivatives Estimation from Noisy Discretizations. In: Coeurjolly, D., Sivignon, I., Tougne, L., Dupont, F. (eds.) DGCI 2008. LNCS, vol. 4992, pp. 370–379. Springer, Heidelberg (2008)
17. Provot, L., Gérard, Y.: Estimation of the derivatives of a digital function with a convergent bounded error. In: Discrete Geometry for Computer Imagery, pp. 284–295. Springer (2011)
18. Vialard, A.: Geometrical Parameters Extraction from Discrete Paths. In: Miguet, S., Ubéda, S., Montanvert, A. (eds.) DGCI 1996. LNCS, vol. 1176, pp. 24–35. Springer, Heidelberg (1996)
19. de Vieilleville, F., Lachaud, J., Feschet, F.: Maximal digital straight segments and convergence of discrete geometric estimators. *Image Analysis*, 988–997 (2005)

Combinatorial Properties of 2D Discrete Rigid Transformations under Pixel-Invariance Constraints[★]

Phuc Ngo¹, Yukiko Kenmochi¹, Nicolas Passat^{2,3}, and Hugues Talbot¹

¹ Université Paris-Est, LIGM, UPEMLV-ESIEE-CNRS, France

² Université de Strasbourg, LSIIT, UMR 7005 CNRS, France

³ Université de Reims, CReSTIC, EA 3804, France

Abstract. Rigid transformations are useful in a wide range of digital image processing applications. In this context, they are generally considered as continuous processes, followed by discretization of the results. In recent works, rigid transformations on \mathbb{Z}^2 have been formulated as a fully discrete process. Following this paradigm, we investigate – from a combinatorial point of view – the effects of pixel-invariance constraints on such transformations. In particular we describe the impact of these constraints on both the combinatorial structure of the transformation space and the algorithm leading to its generation.

Keywords: Combinatorial structure, Discrete rigid transformation, Pixel-invariance constraints.

1 Introduction

Rigid transformations, (*i.e.*, transformations based on translations and rotations) are involved in the design of many computer vision and image processing techniques (see, *e.g.*, [8,9]). Such transformations are generally performed by considering the Euclidean space (\mathbb{R}^n) associated to the Eulerian space (\mathbb{Z}^n) of the data. As a consequence, they need to be interfaced with a subsequent digitization process to finally produce results in \mathbb{Z}^n .

In [5], we have recently proposed to study rigid transformations on \mathbb{Z}^2 as a *fully discrete process*. In this context, three main questions were considered: (i) How many rigid transformations can be defined on a finite subspace of \mathbb{Z}^2 ? (ii) How to generate all of them? (iii) What are the topological relationships between them? Some combinatorial and algorithmic answers, inspired by the approaches developed in [3,4], were provided, and then contributed to the state of the art in this research field [3,4,2,7]. In [5], a combinatorial structure – namely a *graph* – is used to represent the 2D discrete rigid transformations. This structure has a polynomial complexity $O(N^9)$ where $N \times N$ is the size of images. However, this high complexity makes it difficult to generate the whole graph for large images, and to further find admissible transformations that best match two given images, namely a template and a target image; the later problem is called *image registration*. Practically in computer vision, some constrained search paradigms are used for registration issues (see, *e.g.*, [9,1]). Indeed the constraints introduce prior knowledge of transformations and contribute to reduce the research space.

[★] The research leading to these results has received funding from the French *Agence Nationale de la Recherche* (Grant Agreement ANR-2010-BLAN-0205 03).

In this article, we extend the study initiated in [5], by investigating the effects of geometric constraints on the proposed graph. In particular, we focus on *pixel-invariance constraints*, which consist of forcing the correspondence between points in an initial (sub)space (of \mathbb{Z}^2) and transformed points – or more generally regions.

This study is organised as follows. We first recall background notions on discrete rigid transformations (Sec. 2), and express pixel-invariance constraints in the associated parameter space (Sec. 3). We then develop an algorithmic process for generating a combinatorial structure modeling all the discrete rigid transformations and their relationships under the given constraints (Sec. 4). A complexity analysis is proposed for this algorithm and the induced structure (Sec. 5). Finally, we conclude the article (Sec. 6).

2 Background Notions of Discrete Rigid Transformations

2.1 Digital Images and Digital Rigid Transformations

In the continuous framework, an image can be formalised as a function $\mathcal{I} : \mathbb{R}^2 \rightarrow V$, where V is any value space. A *digital image* associated to \mathcal{I} can then be defined as $I : \mathbb{Z}^2 \rightarrow V$, by sampling \mathcal{I} on the discrete space \mathbb{Z}^2 . In other words, we have $I = \mathcal{I}|_{\mathbb{Z}^2}$, and for each $\mathbf{p} \in \mathbb{Z}^2$, the value $I(\mathbf{p})$ models the value of \mathcal{I} on the associated pixel $\mathbf{p} + [-\frac{1}{2}, \frac{1}{2}]^2$, namely the Voronoi cell of \mathbb{R}^2 induced by \mathbb{Z}^2 around \mathbf{p} . This paradigm relies on the digitisation function, where $[\cdot]$ is a rounding operator, defined as

$$\left| \begin{array}{l} D : \mathbb{R}^2 \longrightarrow \mathbb{Z}^2 \\ (x, y) \longmapsto ([x], [y]) \end{array} \right. \tag{1}$$

In the continuous framework, a 2D rigid transformation, composed of translation and rotation, is expressed as a bijection $\mathcal{T} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined, for any $\mathbf{x} = (x, y) \in \mathbb{R}^2$, by

$$\mathcal{T}(\mathbf{x}) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \tag{2}$$

where $a, b, \theta \in \mathbb{R}$ and $\theta \in [0, 2\pi[$. In particular, such a transformation is unambiguously modeled by the triplet of parameters (a, b, θ) , and will sometimes be noted by $\mathcal{T}_{ab\theta}$. When applied on an image $\mathcal{I} : \mathbb{R}^2 \rightarrow V$, it provides a new transformed image $\mathcal{I} \circ \mathcal{T}^{-1} : \mathbb{R}^2 \rightarrow V$.

Following the digitisation paradigm proposed above, a *digital rigid transformation* $T : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ associated to \mathcal{T} can be defined, for any $\mathbf{p} = (p, q) \in \mathbb{Z}^2$, by

$$T(\mathbf{p}) = D \circ \mathcal{T}(\mathbf{p}) = \begin{pmatrix} [p \cos \theta - q \sin \theta + a] \\ [p \sin \theta + q \cos \theta + b] \end{pmatrix} \tag{3}$$

In general, this function is not bijective. However, by setting $T^{-1} : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$ as $T^{-1} = D \circ \mathcal{T}^{-1}$, it becomes possible to define the digital transformed image $I \circ T^{-1} : \mathbb{Z}^2 \rightarrow V$ with respect to T . In the sequel of this article, we focus on such digital rigid transformations. From this point on – for the sake of readability and without loss of correctness – we will denote them T instead of T^{-1} , due to the bijectivity of \mathcal{T} and \mathcal{T}^{-1} .

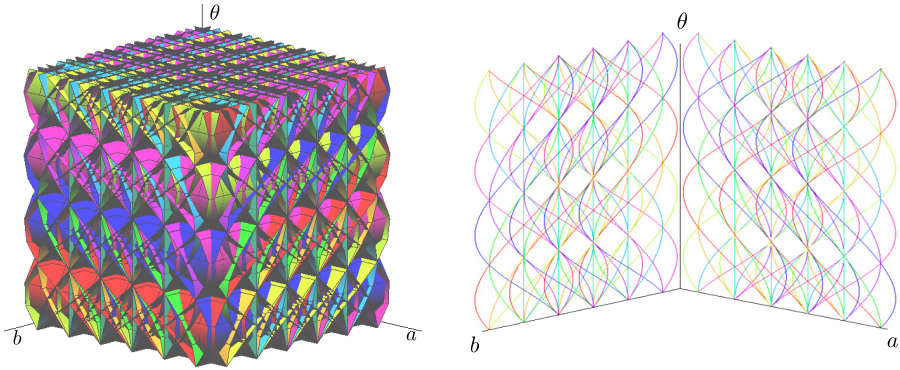


Fig. 1. Tipping surfaces in the 3D parameter space (a, b, θ) (left) and their cross-sections, namely tipping curves, in the 2D planes (a, θ) and (b, θ) (right)

2.2 Discontinuities of Digital Rigid Transformations

For any \mathbf{x} in Eq. (2), if we change the value of (a, b, θ) slightly, then the new point $\mathcal{T}(\mathbf{p})$ also changes slightly. More formally, the function $(a, b, \theta) \mapsto \mathcal{T}_{ab\theta}$ is continuous. Contrariwise, in Eq. (3), an infinitesimal variation of (a, b, θ) may lead to a variation of $T(\mathbf{p})$ from a point of \mathbb{Z}^2 to another one. More precisely, the parameter space \mathbb{R}^3 of (a, b, θ) is divided into 3D open cells where the function $(a, b, \theta) \mapsto T_{ab\theta} = D \circ \mathcal{T}_{ab\theta}$ is piecewise constant, bounded by 2D surfaces where it is discontinuous.

We focus in particular on the triplets (a, b, θ) and their associated transformations $\mathcal{T}_{ab\theta}$, which lead to such discontinuities in the space of digital rigid transformations. Such *critical* transformations are those that map at least one discrete point onto the discrete half-grid $\mathcal{H} = (\mathbb{R} \times (\mathbb{Z} + 1/2)) \cup ((\mathbb{Z} + 1/2) \times \mathbb{R})$ (i.e., the boundaries of the Voronoi cells of \mathbb{R}^2 induced by \mathbb{Z}^2).

Definition 1 ([5]). Let $(a, b, \theta) \in \mathbb{R}^3$, and $\mathcal{T}_{ab\theta} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be its associated rigid transformation. We say that $\mathcal{T}_{ab\theta}$ is a critical transformation if $\exists \mathbf{p} \in \mathbb{Z}^2$ s.t. $\mathcal{T}_{ab\theta}(\mathbf{p}) \in \mathcal{H}$.

It is plain that in the parameter space (a, b, θ) , the critical transformations are modeled by 2D surfaces analytically defined, for any $\mathbf{p} = (p, q) \in \mathbb{Z}^2$ and $k, l \in \mathbb{Z}$, by

$$\left\{ \begin{array}{l} \Phi_{pqk} : \mathbb{R}^2 \longrightarrow \mathbb{R} \\ (b, \theta) \longmapsto a = \phi_{pqk}(\theta) = k + \frac{1}{2} + q \sin \theta - p \cos \theta, \end{array} \right. \quad (4)$$

$$\left\{ \begin{array}{l} \Psi_{pql} : \mathbb{R}^2 \longrightarrow \mathbb{R} \\ (a, \theta) \longmapsto b = \psi_{pql}(\theta) = l + \frac{1}{2} - p \sin \theta - q \cos \theta. \end{array} \right. \quad (5)$$

The surfaces Φ_{pqk} (resp. Ψ_{pql}) are termed *tipping surfaces* [5]. Their intersection ϕ_{pqk} (resp. ψ_{pql}) on the 2D plane (a, θ) (resp. (b, θ)) are called *tipping curves*. These tipping surfaces/curves, which correspond to the discontinuities of digital rigid transformations expressed in the parameter space (a, b, θ) , are illustrated in Fig. 1.

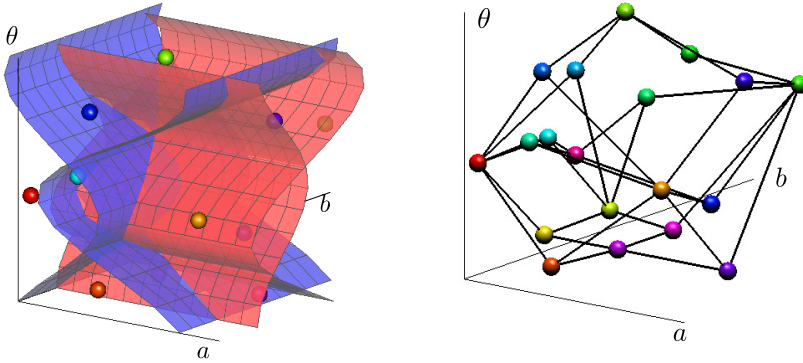


Fig. 2. Parameter space subdivided by four tipping surfaces (left) and its DRT graph (right)

2.3 Partition of Parameter Space and Discrete Rigid Transformation Graph

The digitisation process (Eq. (1)) generally maps distinct rigid transformations (Eq. (2)) onto the same digital rigid transformation (Eq. (3)). More precisely, the set of all the *non-critical transformations* can be partitioned into equivalence classes induced by the equivalence relation \sim defined by $(\mathcal{T}_{ab\theta} \sim \mathcal{T}_{a'b'\theta'}) \iff (T_{ab\theta} = T_{a'b'\theta'})$. This also leads to the straightforward definition of an equivalence relation on the parameters (a, b, θ) associated to these transformations. In this isomorphic framework, each equivalence class is called a *discrete rigid transformation (DRT)*, and is modeled by 3D open cells bounded by 2D tipping surfaces, which subdivide the parameter space (a, b, θ) (see Fig. 1(a)).

In [5], we have shown that this subdivision of the parameter space could be modeled using a dual combinatorial structure, namely a graph. In particular, each 3D open cell (*i.e.*, each DRT) is associated to a vertex, and each tipping-surface segment (linked to a critical transformation) shared by two adjacent 3D open cells, is associated to an edge. The resulting graph is called a *DRT graph* [5] (see Fig. 2).

From a theoretical point of view, the notions introduced above are correctly defined for images and transformations on \mathbb{Z}^2 and \mathbb{R}^2 . Practically, our purpose is to study such transformations on images of *finite sizes*. Under this hypothesis, only a finite subset of digital rigid transformations are relevant, namely those which actually affect such finite images. From this point on, we focus on this finite case and assume that the digital images are defined on subsets of \mathbb{Z}^2 of size $N \times N$ ($N \in \mathbb{N}$). We have the next property.

Property 2 ([5]). *The DRT graph associated to a digital image of size $N \times N$ has a space complexity of $O(N^9)$.*

In [5], an exact computation algorithm was also proposed to build this graph in linear time w.r.t. its size. The DRT graph models a kind of “neighbouring” relationship between DRTs. Indeed, the existence of an edge between two vertices indicates that the associated transformations differ in one pixel among the N^2 ones. This property opens a way of involving this combinatorial structure in image processing tasks.

We now study the effects of forcing the correspondence between points in the initial and transformed spaces. We focus in particular on the subdivision of the parameter space and the induced graph from the algorithmic and combinatorial points of view.

3 Constraints and Feasible Rigid Transformation Set

3.1 Pixel-Invariance Constraints and Interpretation in the Parameter Space

In the context of rigid transformations in \mathbb{R}^2 , forcing the correspondence between points \mathbf{p} in the initial space and \mathbf{p}' in the transformed one leads to restricting the transformations \mathcal{T} . Moreover, forcing the correspondence between $k \geq 2$ distinct pairs of points $(\mathbf{p}_i, \mathbf{p}'_i)$ restricts the number of feasible transformations \mathcal{T} to at most one. Indeed, from the relation expressed in Eq. (2) we obtain for every pair of corresponding points two equations representing the trigonometric surfaces. We then need at least two pairs of corresponding points to obtain a rigid transformation as the feasible transformation. As illustrated in Fig. 3(a), each pair of surfaces represents a pair of corresponding points, and the intersection of two pairs of surfaces determines the feasible transformation.

Restricting discrete rigid transformations, under similar constraints, is more permissive. Indeed, when forcing the correspondence between one or several pairs of pixels $(\mathbf{p}_i, \mathbf{p}'_i)$ of \mathbb{Z}^2 , a larger space of transformations may remain valid (see Fig. 3(b-g)).

Definition 3. Let $\mathbf{p} = (p, q) \in A \subset \mathbb{Z}^2$ and $\mathbf{p}' = (p', q') \in B \subset \mathbb{Z}^2$, such that A, B are of size $N \times N$. There exists a pixel-invariance constraint between \mathbf{p} and \mathbf{p}' if the authorised digital rigid transformations T between A and B satisfy the equality $T(\mathbf{p}) = \mathbf{p}'$, i.e., if

$$p' - 1/2 < p \cos \theta - q \sin \theta + a < p' + 1/2, \tag{6}$$

$$q' - 1/2 < p \sin \theta + q \cos \theta + b < q' + 1/2. \tag{7}$$

More generally, there exist pixel-invariance constraints between two sets $\{\mathbf{p}_i\}_{i=1}^m$ and $\{\mathbf{p}'_i\}_{i=1}^m$ ($m \geq 1$) if $T(\mathbf{p}_i) = \mathbf{p}'_i$ (i.e., if Eqs. (6)–(7) are satisfied) for every $i \in \llbracket 1, m \rrbracket$.

In absence of constraints, the 3D parameter space (a, b, θ) , induced by the subset of size $N \times N$ where the image is defined, is divided into cells whose boundaries are all the tipping surfaces Φ_{pqk} and Ψ_{pql} , with $p, q \in \llbracket 0, N - 1 \rrbracket$ and $k, l \in \llbracket 0, N \rrbracket$. In this context, the whole volume of the parameter space models adequate rigid transformations.

In contrast, under a pixel-invariance constraint, some discrete rigid transformations may become irrelevant. Equivalently, only a part of the parameter space – namely the subspace of parameters (a, b, θ) that satisfy this constraint – remains valid. From Eqs. (6)–(7), this parameter subspace is defined by the intersection of 4 half-spaces associated to 4 tipping surfaces for one pixel correspondence (see Fig. 3(b)).

3.2 Feasible Rigid Transformation Set

More generally, if a set \mathcal{P} of m pixel correspondences is provided, the parameter subspace of relevant transformations is defined as the intersection of m regions induced by

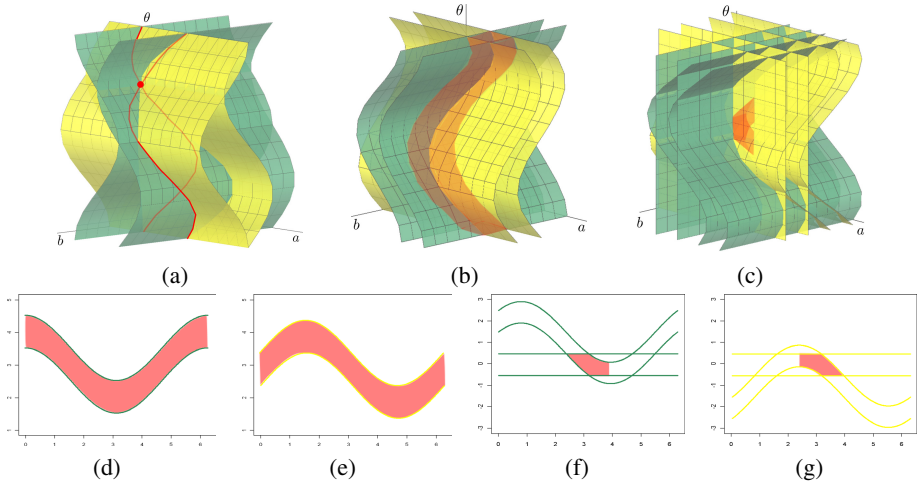


Fig. 3. Rigid transformation sets induced by geometric constraints in continuous (a) and discrete (b,c) frameworks. (a) Transformations with one-point correspondence (red line) and two-point correspondences (red dot). (b,c) Transformations with one-pixel (b) and two-pixels (c) correspondences (red volumes). (d,e) (resp. (f,g)) Cross-sections of (b) (resp. (c)) on the planes (a, θ) and (b, θ) via the use of tipping curves.

these constraints, *i.e.*, as the intersection of $4m$ half-spaces defined by Eqs. (6)–(7). Let us first define the half-spaces induced by tipping surfaces Φ_{pqk} and Ψ_{pql} :

$$H_{pqk}^+ = \{(a, b, \theta) \mid a > \Phi_{pqk}(b, \theta)\} \text{ and } H_{pqk}^- = \{(a, b, \theta) \mid a < \Phi_{pqk}(b, \theta)\}, \quad (8)$$

$$V_{pql}^+ = \{(a, b, \theta) \mid b > \Psi_{pql}(a, \theta)\} \text{ and } V_{pql}^- = \{(a, b, \theta) \mid b < \Psi_{pql}(a, \theta)\}. \quad (9)$$

The subspace of interest, called *feasible rigid transformation set*, is defined as follows.

Definition 4. Let $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{p}'_i)\}_{i=1}^m$ ($m \geq 1$) be a set of corresponding pixel pairs. The feasible rigid transformation set (FRTS) associated to \mathcal{P} is the subspace $\mathcal{R} \subset \mathbb{R}^3$ of the parameter space (a, b, θ) , defined as

$$\mathcal{R} = \bigcap_{i \in \llbracket 1, m \rrbracket} \left(H_{p_i q_i p'_i}^+ \cap H_{p_i q_i p'_i + 1}^- \cap V_{p_i q_i q'_i}^+ \cap V_{p_i q_i q'_i + 1}^- \right).$$

Note that a constraint of one pixel pair (namely, $m = 1$), the FRTS is observed as a “tube” in the parameter space (a, b, θ) (see Fig. 3(b)). We can consider that a pixel-invariance constraint leads to all rotations with a center $\mathbf{x} \in \mathbb{R}^2$ inside of pixel $\mathbf{p} \in \mathbb{Z}^2$ (*i.e.*, $\mathbf{x} \in \mathbf{p} + [-1/2, 1/2]^2$). For constraints of two pixel pairs (namely, $m = 2$), the FRTS becomes a “bounded region” (see Fig. 3(c)).

The FRTS generated by m pixel correspondences is divided into DRTs induced from the $(N^2 - m)$ remaining pixels of the given image of size $N \times N$. In particular, the combinatorial structure of DRTs in a FRTS, modeling this subdivision, is a subgraph of the DRT graph. We introduce, in the following, a notion of *directional convexity* of a region R , such that R is “convex along an axis”, and show that any FRTS \mathcal{R} is directionally convex. This property is used in the next sections to study the combinatorial structure of DRTs in \mathcal{R} .

Definition 5. We say that a region $R \subseteq \mathbb{R}^n$ in an n -variable space (x_1, \dots, x_n) is x_k -convex if, for any two points $\mathbf{p}_1, \mathbf{p}_2 \in R$ such that the segment $[\mathbf{p}_1\mathbf{p}_2] = \{\alpha \cdot \mathbf{p}_1 + (1-\alpha) \cdot \mathbf{p}_2 \mid \alpha \in [0, 1]\}$ is parallel to the x_k -axis, $[\mathbf{p}_1\mathbf{p}_2]$ is included in R .

Property 6. The FRTS \mathcal{R} is both a - and b -convex.

4 Combinatorial Structure of Discrete Rigid Transformations in a Feasible Rigid Transformation Set

So far, we know that a FRTS contains all rigid transformations satisfying all constraints, and is subdivided into DRTs (see Sec. 2.3). This section presents a method for constructing a combinatorial structure of DRTs in a FRTS by following the three stages: (1) finding the boundaries of a FRTS, (2) finding tipping surfaces passing a FRTS and (3) constructing a DRT graph in a FRTS. Before describing these stages, we first explain an algorithm for building a graph modeling a subdivision of the parameter space from a given set of tipping surfaces, which is used later in the first and third stages.

4.1 Sweeping Algorithm for Incremental Partition Graph Construction

We may generalize the problem of subdivision of the parameter space by tipping surfaces as follows: given a set of tipping surfaces S , we would like to construct a graph modeling the subdivision of the parameter space (a, b, θ) induced by S . Such a graph is called a *partition graph* and denoted by G . In G each vertex is associated to a 3D open cell of the subdivision, and each tipping-surface segment shared by two adjacent 3D open cells, is associated to an edge. This problem can be answered with the help of 3D arrangements of surfaces [6]. In [5], we have proposed the *sweeping algorithm* for constructing a DRT graph of a given image (see Section 2.3), which is a specific case of generating a partition graph. Such a method has a complexity $\mathcal{O}(n^3)$, where n is the number of surfaces. Here we present a similar method that builds the partition graph G based on the relations that link tipping surfaces and tipping curves (see Eqs (4)–(5) and Fig. 1). Such a subdivision can be fully described from its two cross-sections on the planes (a, θ) and (b, θ) , respectively expressed by two sets of tipping curves. Therefore, instead of constructing directly the partition graph in the 3D parameter space (a, b, θ) , we will first build the structures of the graphs in the 2D planes (namely, (a, θ) and (b, θ) planes), and then combine them to build the complete partition graph. Note that G becomes a DRT graph if we consider all tipping surfaces for a given image.

We first define a *cut* for a plane – either (a, θ) or (b, θ) – denoted by γ , as a monotonic line intersecting exactly once for each tipping curve in the plane. A cut is then represented by its sequence of intersecting tipping curves (see Fig. 4(a)). Such a cut can be modeled by a directed graph according to their sequences of tipping curves.

Definition 7. Let $\gamma = (\phi_1, \phi_2, \dots)$ be a cut. A graph $G_\gamma = (V_\gamma, E_\gamma)$ w.r.t. γ consists of
– a set of vertices $V_\gamma = \{v_0, v_1, \dots\}$, and
– an ordered set of labelled edges $E_\gamma = ((v_0, v_1, \phi_1), (v_2, v_1, \phi_2), \dots)$ and each edge $(u, w, f) \in E_\gamma$ connects two vertices $u, w \in V_\gamma$ separated by the tipping curve f , which is considered as an edge label.

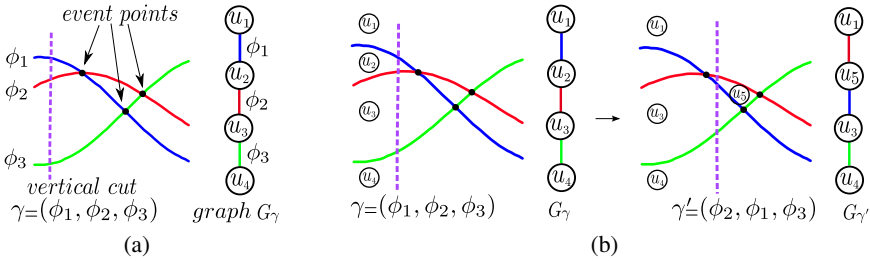


Fig. 4. (a) Example of a cut and its graph. (b) Progress of the cut at an event point by which the cut is updated and the corresponding graph is modified.

In practice, the elements of E_γ are also ordered in the same way as γ (see Fig. 4(a)).

The main idea of the sweeping method in 2D is that a cut is swept through all tipping curves on the plane in finite time, allowing us to construct the graph afterwards. We assume that γ starts at $\theta = 0$ and ends at $\theta = 2\pi$. While sweeping the cut, its sequence changes only at intersections of tipping curves, called *event points*. The moment at which a cut reaches an event point, the algorithm performs an update of its sequence, and generates new vertices and edges in the graph (see Fig. 4(b)). We call this an *elementary step* of the algorithm. The set of event points forms a series of elementary steps. Therefore, instead of moving the cut continuously, we need to maintain a set of sorted event points w.r.t. θ , and progress the cut in their increasing order to build the graph incrementally.

For building a graph G in 3D, two *cuts* are used such that each cut sweeps on either the plane (a, θ) or (b, θ) . We denote those cuts by γ_a and γ_b respectively. For each update of the cuts, γ_a and γ_b , the associated graphs, G_{γ_a} and G_{γ_b} , are respectively modified, so that a part of G is generated. We call such a part of G a *partial graph*, denoted by δG . In fact, δG is a combination of the two graphs G_{γ_a} and G_{γ_b} as follows (see Fig. 5).

Definition 8. The partial graph $\delta G = (\delta V, \delta E)$ is generated from $G_{\gamma_a} = (V_{\gamma_a}, E_{\gamma_a})$ and $G_{\gamma_b} = (V_{\gamma_b}, E_{\gamma_b})$, such that

- $\delta V = \{(v_a, v_b) \mid v_a \in V_{\gamma_a}, v_b \in V_{\gamma_b}\}$, and
- $\delta E = \{((u_1, v), (u_2, v), \phi_u) \mid u_1, u_2 \in V_{\gamma_a}, v \in V_{\gamma_b}, (u_1, u_2, \phi_u) \in E_{\gamma_a}\} \cup \{((u, v_1), (u, v_2), \phi_v) \mid v_1, v_2 \in V_{\gamma_b}, u \in V_{\gamma_a}, (v_1, v_2, \phi_v) \in E_{\gamma_b}\}$.

Therefore, when an elementary step is applied, the sweep progresses as the partial graph δG is generated and integrated in G for constructing the final graph as well. The following proposition has been originally proposed in [5] for constructing a DRT graph, and this is valid for a partition graph as well.

Proposition 9. Let S be a set of tipping surfaces and G be a partition graph modeling the subdivision of the parameter space by S . We have

$$G = \bigcup_{i \in \llbracket 1, e \rrbracket} \delta G_i,$$

where δG_i is a partial graph at the i -th elementary step and e is the number of ordered event points.

More details about the sweeping algorithm for tipping surfaces can be found in [5].

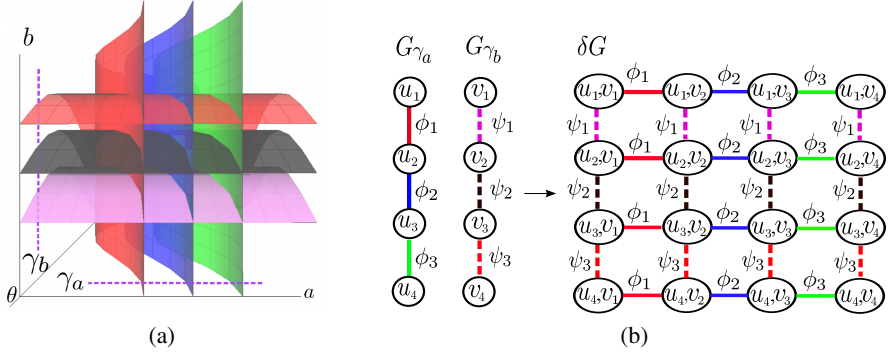


Fig. 5. Construction of a partial graph δG by combining two graphs G_{γ_a} and G_{γ_b}

4.2 Finding the Feasible Rigid Transformation Set Boundary

It is possible to describe a FRTS \mathcal{R} by a set of half-spaces constituting the boundary of \mathcal{R} , instead of using all the half-spaces of \mathcal{R} as described in Definition 4. This section explains how to find such a set using the above sweeping algorithm.

A FRTS \mathcal{R} in the parameter space (a, b, θ) can be fully described from its two cross-sections \mathcal{R}_H and \mathcal{R}_V on the planes (a, θ) and (b, θ) , defined as

$$\mathcal{R}_H = \bigcap_{i \in \llbracket 1, m \rrbracket} \left(h_{p_i, q_i}^+ \cap h_{p_i, q_i'}^- \right) \text{ and } \mathcal{R}_V = \bigcap_{i \in \llbracket 1, m \rrbracket} \left(v_{p_i, q_i}^+ \cap v_{p_i, q_i'}^- \right) \quad (10)$$

where h_{p_i, q_i}^+ and $h_{p_i, q_i'}^-$ (resp. v_{p_i, q_i}^+ and $v_{p_i, q_i'}^-$) are the cross-sections in the plane (a, θ) (resp. (b, θ)) of H_{p_i, q_i}^+ and $H_{p_i, q_i'}^-$ (resp. V_{p_i, q_i}^+ and $V_{p_i, q_i'}^-$). This is illustrated in Fig. 3(d–g). They are expressed as shown in Eq. (8) (resp. Eq. (9)) by replacing the tipping surfaces Φ_{pqk} and Ψ_{pql} by the tipping curves ϕ_{pqk} and ψ_{pql} respectively. We call $h_{p_i, q_i}^+, v_{p_i, q_i}^+$ *upper half-planes* and $h_{p_i, q_i'}^-, v_{p_i, q_i'}^-$ *lower half-planes*.

Relying on the similarity of \mathcal{R}_V and \mathcal{R}_H , hereafter we consider only \mathcal{R}_H . Our problem is then specified as follows: given a constraint set of half-planes of \mathcal{R}_H defined from m corresponding pixel pairs, $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{p}'_i)\}_{i=1}^m$, report the boundary half-planes of \mathcal{R}_H . From Property 6, it is obvious that \mathcal{R}_H contains two sets of boundary half-planes:

- a *upper boundary sequence* $U = (h_{p_i, q_i}^+, \dots)$ contains only the upper half-planes;
- a *lower boundary sequence* $L = (h_{p_i, q_i'}^-, \dots)$ contains only the lower half-planes.

The 2D sweeping algorithm, presented in Sec. 4.1, is used to find such U and L of \mathcal{R}_H , such that the cut γ is now represented as a sequence of half-planes intersecting it. Note that no partition graph is built in this stage, and we only need to observe the sequence of the cut γ during its update in order to obtain U and L . Indeed, while sweeping γ its sequence changes at event points. We remark that γ is in \mathcal{R}_H when its sequence of half-planes is separated into two successive sequences of γ^+ and γ^- , namely $\gamma = \gamma^+ \gamma^-$, where γ^+ contains only the upper half-planes and γ^- contains only the lower half-planes. Moreover, we see that the last element of γ^+ and the first element of γ^- determine the upper and lower boundaries of \mathcal{R}_H respectively. The cut is located out of \mathcal{R}_H when there is no more such a separation. According to the change of γ in \mathcal{R}_H , the

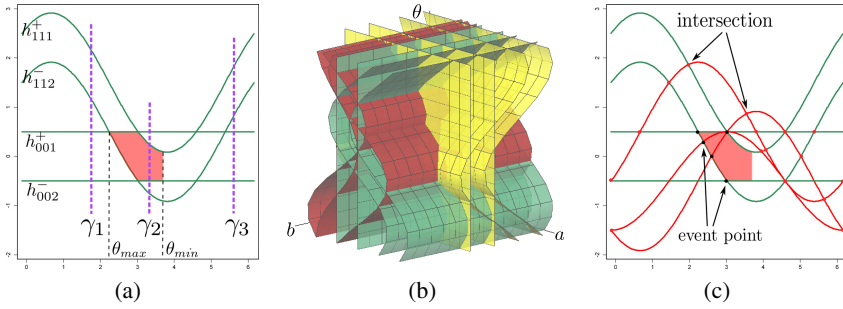


Fig. 6. (a) Progression of the cut γ in the cross-section \mathcal{R}_H of FRTS \mathcal{R} on the plane (a, θ) . The initial cut is $\gamma_1 = (h_{111}^+, h_{112}^-, h_{001}^+, h_{002}^-)$; when it goes in \mathcal{R}_H it has become $\gamma_2 = (h_{001}^+, h_{111}^+, h_{002}^-, h_{112}^-)$; when it goes out of \mathcal{R}_H it becomes $\gamma_3 = (h_{111}^+, h_{112}^-, h_{001}^+, h_{002}^-)$. (b) Example of tipping surfaces (in red) passing \mathcal{R} in the parameter space (a, b, θ) and (c) its cross-section \mathcal{R}_H on the plane (a, θ) .

upper and lower boundaries are added progressively in U and L at each event point. See Fig. 6(a) for the illustration.

By using two cuts γ_a and γ_b sweeping on the two planes (a, θ) and (b, θ) , we can find the boundary of a FRTS \mathcal{R} . At each event point either on (a, θ) or (b, θ) , the algorithm updates and checks the sequences of both cuts. From this, we obtain the boundary of \mathcal{R} , and the first θ at which both sequences of γ_a and γ_b are separated in two parts (resp. have no more separation), denoted by θ_{min} (resp. θ_{max}) of \mathcal{R} that need in the next stage.

4.3 Finding Tipping Surfaces Passing by a Feasible Rigid Transformation Set

As the subdivision of a FRTS \mathcal{R} is induced by the tipping surfaces existing in \mathcal{R} (see Fig. 6(b)), we need to determine such tipping surfaces among all Φ_{pqk} and Ψ_{pql} for $p, q \in \llbracket 0, N-1 \rrbracket$ and $k, l \in \llbracket 0, N \rrbracket$, where $N \times N$ is the image size. Now looking at the cross-sections of \mathcal{R} , this problem is equivalent to finding tipping curves ϕ_{pqk} (resp. ψ_{pql}) passing \mathcal{R}_H (resp. \mathcal{R}_V) (see Fig. 6(c)).

A tipping curve ϕ_{pqk} passes \mathcal{R}_H if it intersects one of the boundary segments of $\phi_{p'q'k'}$ of \mathcal{R}_H . This is easily verified by the following steps:

- (i) verify if ϕ_{pqk} and $\phi_{p'q'k'}$ intersect, *i.e.*, if the following relation is satisfied [5, Property 2]: $\Delta_1 + \Delta_2 > 0$ and $|KP \pm \sqrt{\Delta_1}| \leq P^2 + Q^2$ and $|KQ \pm \sqrt{\Delta_2}| \leq P^2 + Q^2$, where $P = p - p'$, $Q = q - q'$, $K = k - k'$, $\Delta_1 = P^2(P^2 + Q^2 - K^2)$ and $\Delta_2 = Q^2(P^2 + Q^2 - K^2)$;
- (ii) if they intersect, then calculate the following values at the intersection [5, Corollary 1]: $\sin \theta = \frac{KQ \pm \sqrt{\Delta_1}}{P^2 + Q^2}$ and $\cos \theta = \frac{KP \pm \sqrt{\Delta_2}}{P^2 + Q^2}$, and verify if $\theta_{min} \leq \theta \leq \theta_{max}$, where θ_{min} and θ_{max} are obtained in Sec. 4.2;
- (iii) if (ii) is verified, then calculate $a_{upper} = \max_{h_{pqk}^+ \in U} \phi_{pqk}(\theta)$ and $a_{lower} = \min_{h_{pqk}^- \in L} \phi_{pqk}(\theta)$, and verify if $a_{upper} \leq a \leq a_{lower}$, where the value a at the above intersection θ is calculated from Eq. (4).

Note that the values $\cos \theta$ and $\sin \theta$ are used to represent θ . All $\cos \theta$, $\sin \theta$, $\cos \theta_{min}$, $\sin \theta_{min}$, $\cos \theta_{max}$, $\sin \theta_{max}$, a , a_{min} and a_{max} are quadratic irrationals¹. As shown in [5], their comparisons can be achieved exactly in constant time.

¹ A quadratic irrational is an irrational number that is a solution of some quadratic equations.

We define *tipping surfaces of interest* as a set of tipping surfaces that bound or pass a FRTS \mathcal{R} . Similarly, *tipping curves of interest* of $\mathcal{R}_{\mathcal{H}}$ (resp. $\mathcal{R}_{\mathcal{V}}$) is the set of the boundary tipping curves of $\mathcal{R}_{\mathcal{H}}$ (resp. $\mathcal{R}_{\mathcal{V}}$) and the tipping curves passing $\mathcal{R}_{\mathcal{H}}$ (resp. $\mathcal{R}_{\mathcal{V}}$).

4.4 DRT-Graph Construction in a Feasible Rigid Transformation Set

In order to build the DRT graph in a FRTS \mathcal{R} , we use the sweeping algorithm described in Sec. 4.1. However the cut γ in this part sweeps from θ_{min} to θ_{max} instead of $[0, 2\pi]$, and contains only tipping surfaces existing between the upper and lower boundaries of \mathcal{R} . The following question then arises: how can we detect event points in \mathcal{R} ? Or when is an elementary step applied? Because of the similarity of $\mathcal{R}_{\mathcal{V}}$ and $\mathcal{R}_{\mathcal{H}}$, in the following we consider only the cross-section $\mathcal{R}_{\mathcal{H}}$ of \mathcal{R} . Event points in $\mathcal{R}_{\mathcal{H}}$ are now defined as intersections of tipping curves of interest being either on a boundary segment or inside of $\mathcal{R}_{\mathcal{H}}$, as illustrated in Fig. 6(c). According to its nature, it is called either a *boundary event point* or an *inside event point*. Similarly to the method in Sec. 4.3 if an intersection coordinate (θ, a) satisfies $\theta_{min} \leq \theta \leq \theta_{max}$ and $a_{min} \leq a \leq a_{max}$, then it is verified to be an event point. The algorithm described in Sec. 4.1 deals with any inside event points. In contrary, the boundary event points must be treated separately as follows.

As described in Sec. 4.1, an elementary step at each event point consists of (i) updating the graphs G_{γ_a} and G_{γ_b} according to the change of γ_a and γ_b respectively (explained below) and (ii) building the partial graph δG from G_{γ_a} and G_{γ_b} (see Definition 7).

In [5], we classified inside event points into two cases: simple intersections and degeneracies (see Fig. 7). Figure 8 shows an elementary step at a simple intersection. In [5], the degeneracies are processed by modifying this simple case.

Regarding boundary event points, they can be classified into the following six cases (Fig. 9), which are easily detected by checking the tipping curves intersecting at the event point with the tipping curves in γ and the upper and lower bound sequences U and L . The procedure for handling them in simple cases is explained below, while the degenerate cases are treated similarly to [5] and omitted in this paper due to the page number limitation. As illustrated in Fig. 9, an event point:

- changes the boundary, which is either upper (a) or lower (b);
- does not change the boundary, such that one of the tipping curves
 - goes in (resp. out) \mathcal{R} by the upper boundary (c) (resp. (d));
 - goes in (resp. out) \mathcal{R} by the lower boundary (e) (resp. (f)).

We first explain how to update the cut for (a) and (b). Without loss of generality, let $\mathbf{q} = \{\phi_u, \phi_v\}$ be a boundary event point generated by two tipping curves ϕ_u, ϕ_v and γ , γ' be the cuts before and after \mathbf{q} respectively. Assuming $\gamma = (\phi_1, \phi_2, \dots, \phi_{n-1}, \phi_n)$, if \mathbf{q} is

- the upper boundary, i.e., $\phi_u = \phi_1$ and $\phi_v = \phi_2$, then $\gamma' = (\phi_v, \phi_2, \dots, \phi_{n-1}, \phi_n)$,
- the lower boundary, i.e., $\phi_u = \phi_n$ and $\phi_v = \phi_{n-1}$, then $\gamma' = (\phi_2, \phi_3, \dots, \phi_{n-1}, \phi_v)$.

Similarly, the procedures for updating the cut for (c) and (d) are given as follows. Let $\mathbf{q} = \{\phi_u, \phi_v\}$ be an event point on the upper boundary, i.e., $\phi_u = \phi_1$. We have two cases:

- when ϕ_v goes in $\mathcal{R}_{\mathcal{H}}$, i.e., $\phi_v \neq \phi_2$, then $\gamma' = (\phi_1, \phi_v, \phi_2, \dots, \phi_n)$;
- when the curve ϕ_v goes out $\mathcal{R}_{\mathcal{H}}$, i.e., $\phi_v = \phi_2$, then $\gamma' = (\phi_1, \phi_3, \dots, \phi_n)$.

The procedures for (e) and (f) can be considered in the same way. Fig. 10 illustrates he elementary steps for those boundary event points.

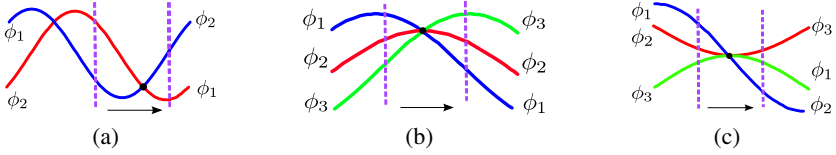


Fig. 7. Inside event point classification: a *simple intersection* if it is generated by only two tipping curves (a), otherwise it is a *degeneracy* (b,c), i.e., when there are more than two tipping curves

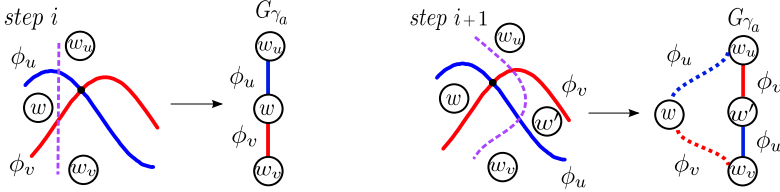


Fig. 8. Updating the graph G_{γ_a} w.r.t. the change of the cut γ_a at a simple intersection

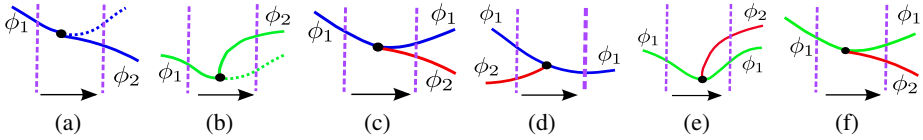


Fig. 9. Classification of simple boundary event points, an event point having a tipping curve that: changes a boundary (a,b), goes in and out by an upper boundary (c,d), goes in and out by a lower boundary (e,f). Upper and lower boundaries are colored in blue and red respectively.

5 Complexity and Experiments

5.1 Complexity Analysis

The space complexity of a DRT graph is proportional to its number of vertices and edges. In absence of constraints, the DRT graph G for an image of size $N \times N$, has a $O(N^9)$ complexity (Property 2). This results from the fact that (i) the number of event points of the whole space is $O(N^6)$, and (ii) at each elementary step (i.e., for each event point), there are $O(N^3)$ vertices generated in the partial graph δG of G [5]. Given one pixel-invariance constraint, some of the potential DRTs become irrelevant. Following the similar proof scheme as above, we can show that the number of event points (i) decreases from $O(N^6)$ to $O(N^5)$ (due to Property 4 in [5] on tipping curves periodicity), and (ii) at each elementary step, $O(N^2)$ vertices are generated instead of $O(N^3)$ since, as explained in Sec. 4, δG of G is generated from two graphs G_{γ_a} and G_{γ_b} of two cuts γ_a and γ_b sweeping on the planes (a, θ) and (b, θ) respectively. Each of the cut intersects at most $O(N^2)$ tipping curves on the plane. Then at each intersection, there are $O(N^2)$ vertices generated in δG . This leads to the following property.

Property 10. *The DRT graph G associated to a digital image of size $N \times N$ under one pixel-invariance constraint has a space complexity of $O(N^7)$.*

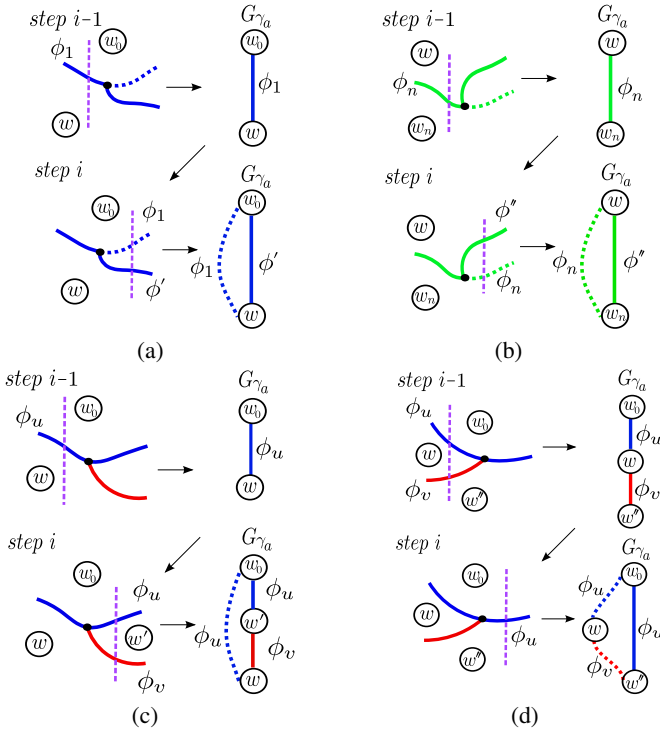


Fig. 10. Illustrations of elementary steps –update γ and generate its graph G_γ – for a tipping curve changing (a) an upper or (b) a lower boundary, and going (c) in or (d) out of an upper boundary

Regarding more complex pixel invariance constraints, we cannot use a similar approach to obtain the theoretical upper bound complexity of G in a FRTS. This is due to the space complexity of G depending on the distance between the involved pixels. Thus, we construct G using our method and we investigate its complexity in practice.

5.2 Computational Experiments

Experiments were carried out on 2D digital images I of size $N \times N$, for 1, 2, 3, 5 and 10 constraints, to investigate how these constraints affect the complexity of the DRT graph. The first experiment (Fig. 11(a)) validates the theoretical $\mathcal{O}(N^7)$ space complexity for one given pixel-invariance constraint. Previous works in [27] on discrete rotations provide a complexity of only $\mathcal{O}(N^3)$. However, they consider only a rotation center at a pixel center, while our approach makes no such assumption. In other words, we consider any discrete rotation whose rotation center is located inside a pixel region, due to one pixel-invariance constraint, instead of a pixel center. For this reason, the complexity is increased from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^7)$. Given two pixel-invariance constraints, the bounded FRTS \mathcal{R} varies randomly with the selected corresponding pixels. Results obtained for an image of size 5×5 are shown in Fig. 11(b) for two random pixel choices with some fixed distances. Results for different image sizes are shown in Fig. 11(c). By taking into account the largest complexity for each image size we obtain a worst case complexity $\mathcal{O}(N^{5.5})$ of a DRT graph G in \mathcal{R} .

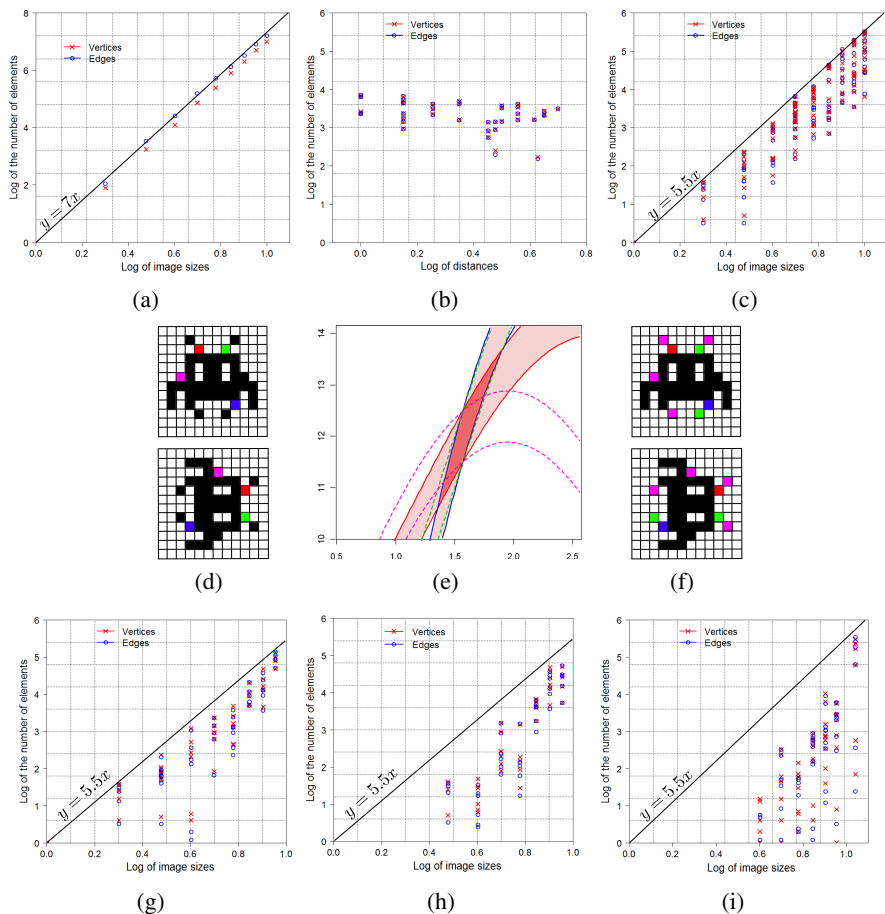


Fig. 11. Experiments for one (a), two (b,c), three (d-g), five (h) and ten (i) pixel-invariance constraints (see Sec. 5.2)

We could expect that \mathcal{R} is strictly reduced when given supplementary pixel-invariance constraints, *i.e.*, more than two constraints. Nevertheless, the third experiment shows that it is not always true. Let us set two pixel-invariance constraints (in red and blue in Fig. 11(d)), and the associated FRTS \mathcal{R} (Fig. 11(e)). A supplementary constraint determined by green pixels in Fig. 11(d) does not reduce \mathcal{R} ; green dotted lines do not pass through \mathcal{R} . Actually (see Fig. 11(f)), there exist two kinds of pixels: one contains some pixels providing no contribution to \mathcal{R} such as the green ones, and another contains pixels allowing to reduce \mathcal{R} such as purple ones. Consequently, the complexity of G depends on given pixel-invariance constraints, but not on the number of constraints.

However, in practice, the more constraints are imposed, the lower the complexity of the DRT graph. This is shown in the experiments for 5 and 10 constraints illustrated in Fig. 11(h) and (i) respectively. Overall there is a downward trend in the numbers of vertices and edges in the DRT graph, though the experimental complexity do not exceed $O(N^{5.5})$.

6 Conclusion

This article continued the study initiated in [5] by investigating the effects of geometric constraints on rigid transformations of digital images. In this work, we addressed pixel-invariance constraints which consist of specifying the correspondence between points in an initial (sub)space (of \mathbb{Z}^2) and pixels in the transformed space. By enforcing correspondence between one or several pairs of pixels, we consequently restrict the feasible transformations into a parameter subspace, called a feasible rigid transformation set (FRTS), in which all such constraints are satisfied. A proposed algorithm allowed us to build a combinatorial structure (namely a graph) for modeling the subdivision of the FRTS on a subset of \mathbb{Z}^2 of size $N \times N$. We have theoretically analysed the complexity of the graph with one given pixel-invariance constraint to be $O(N^7)$. For more than one constraint, the complexity could not be theoretically calculated. However, using our proposed graph construction method we actually built the graph and experimentally investigated its complexity, which was shown not to exceed $O(N^{5.5})$.

Note that the FRTS is generated from a finite intersection of regions of the imposed constraints. In practice, due to the precision of pixel-invariance constraints, we may obtain an empty set of feasible rigid transformations induced by these constraints. In order to avoid this problem, one solution can be to change the resolution of the images. Namely, we degrade the image resolution as in [7] until we find a non-empty and bounded FRTS.

References

1. Amintoosi, M., Fathy, M., Mozayani, N.: A fast image registration approach based on SIFT key-points applied to super-resolution. *Imaging Science Journal* (2011)
2. Amir, A., Kapah, O., Tsur, D.: Faster two-dimensional pattern matching with rotations. *Theoretical Computer Science* 368(3), 196–204 (2006)
3. Hundt, C., Liśkiewicz, M.: Combinatorial Bounds and Algorithmic Aspects of Image Matching under Projective Transformations. In: Ochmański, E., Tyszkiewicz, J. (eds.) *MFCs 2008*. LNCS, vol. 5162, pp. 395–406. Springer, Heidelberg (2008)
4. Hundt, C., Liśkiewicz, M., Ragnar, N.: A combinatorial geometrical approach to two-dimensional robust pattern matching with scaling and rotation. *Theoretical Computer Science* 410(51), 5317–5333 (2009)
5. Ngo, P., Kenmochi, Y., Passat, N., Talbot, H.: Combinatorial structure of rigid transformations in 2D digital images. Technical Report, HAL 00643734 (2012)
6. Sharir, M.: Recent Developments in the Theory of Arrangements of Surfaces. In: Pandu Rangan, C., Raman, V., Sarukkai, S. (eds.) *FST TCS 1999*. LNCS, vol. 1738, pp. 1–21. Springer, Heidelberg (1999)
7. Thibault, Y.: Rotations in 2D and 3D discrete spaces. PhD thesis, University Paris-Est (2010)
8. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. *ACM Computing Surveys* 38(4), 1–45 (2006)
9. Zitová, B., Flusser, J.: Image registration methods: A survey. *Image and Vision Computing* 21(11), 977–1000 (2003)

Novel Morphological Algorithms for Dominating Sets on Graphs with Applications to Image Analysis

Anupama Potluri and Chakravarthy Bhagvati

Department of Computer and Information Sciences,
University of Hyderabad, Hyderabad, India
{apcs, chakcs}@uohyd.ernet.in

Abstract. In this paper, we extend the morphological operators defined for graphs by Cousty et al. to use structuring elements. We then apply these extended operators to develop algorithms for Minimum Dominating Set (MDS) and Minimum Independent Dominating Set (MIDS) on incomplete grid graphs which correspond to binary images with 4-connected neighbourhoods. We show that our algorithm performs as well as the best known heuristic for Minimum Independent Dominating Set. We apply the extended morphological graph operators and algorithms to various image analysis tasks such as distance transforms, skeletons and clustering. In particular, we propose a novel MIDS Skeleton that may potentially reduce the time for reconstructing the original objects. A hierarchical clustering algorithm (also using MIDS) is proposed. This algorithm is analogous to the conventional algorithms that use a distance threshold for clustering. We illustrate the proposed algorithms on several example images and conclude that they are useful in image analysis.

Keywords: Clustering, Grid Graphs, Image Analysis, Minimum Dominating Set, Minimum Independent Dominating Set, Morphological Operators.

1 Introduction

Image analysis, over the years, has brought within its scope a number of diverse techniques such as signal processing, statistics, discrete mathematics, graph theory, etc. Each of these have enriched the field and provided interesting and often powerful insights and led to algorithmic developments and novel applications. In this paper, we explore two such techniques: mathematical morphology and graph theory and propose novel *morphological algorithms* for two important graph theoretic problems viz. Minimum Dominating Sets (MDS) and Minimum Independent Dominating Sets (MIDS), and show that these graph theoretic problems, in turn, are useful in image analysis. In particular, the application of morphological algorithms to distance transforms, skeletonization and clustering is presented.

Mathematical morphology, a non-linear, set-based approach to image processing, developed originally by Matheron and Serra [25] provides a number of powerful tools for image analysis. Developments in morphology have taken mainly two paths. The first is extending the morphological operations developed for binary images to a number of domains such as gray scale images [28], graphs [19,4], fuzzy sets [23,3], colour

images[9][24], simplicial complex spaces [12] etc. The second path is the development of new operations such as opening and closing distributions[6], pattern spectra[20], area morphology[1], watersheds[21][10], etc.

Graph-theoretic approaches have found increasing applications in image analysis from the turn of this century. Some notable contributions are in segmentation[13][27] and computer vision[14]. Classical graph problems such as minimum spanning tree (MST), cuts and partitioning, and sub-graph isomorphism[5] have been applied to such image analysis tasks. In this paper, we show that dominating set problems on graphs may be used to advantage in image analysis tasks such as skeletons and clustering. Images are modelled as *grid graphs* which are a special subset of *unit-disk graphs*, on which dominating set problems are an active area of work[7][22].

This paper makes three distinct contributions by linking dominating sets on grid graphs with image analysis: novel extensions to the graph morphology operators proposed originally by Cousty, et. al[11]; new morphological algorithms for the classical dominating set problems; and, the applications of the proposed algorithms to image analysis.

The rest of the paper is organized as follows: we review and extend the graph morphological operators defined by Cousty et al [11] to use structuring elements in Section 2. The algorithms for computing MDS and MIDS using these operators are presented in Section 3. The comparison of the results of the proposed algorithms with those of the standard algorithms in literature, the application of the extended operators to distance transform and the application of dominating sets for skeleton and clustering of images are presented in Section 4. We conclude with Section 5.

2 Dilations and Erosions with Structuring Elements

Cousty, et. al [11] recently defined an entire hierarchy of graph morphological operators from a set of four fundamental operators that derive a set of edges from a given set of vertices and vice versa. Combinations that derive vertices from vertices and edges from edges are shown to be dilations and erosions. Briefly, consider a graph $X = (X^\bullet, X^\times)$ where X^\bullet is the set of vertices and X^\times is the set of edges. The four fundamental operators are: $\delta^\bullet(X^\times)$ which returns the set of all vertices that belong to the edges in X^\times ; $\epsilon^\times(X^\bullet)$ which returns the set of all edges, both ends of which are in X^\bullet ; $\epsilon^\bullet(X^\times)$ which returns only the isolated vertices of the complement of the graph; $\delta^\times(X^\bullet)$ which returns all the edges, at least one end of which is part of X^\bullet . These can then be combined to form vertex dilation ($\delta = \delta^\bullet \circ \delta^\times$) and erosion ($\epsilon = \epsilon^\bullet \circ \epsilon^\times$) as well as edge dilation ($\Delta = \delta^\times \circ \delta^\bullet$) and erosion ($\varepsilon = \epsilon^\times \circ \epsilon^\bullet$) operators. Combining these two leads to graph dilation and erosion. In addition, they define opening and closing, filters and granulometries as in classical mathematical morphology using these operators. In this paper, we extend these operators to incorporate structuring elements and apply them to MDS and MIDS problems.

Any binary image may be modelled as the complete grid graph \mathbb{G} shown in Fig. 1. Note that Cousty in [11] used the same grid graph to formulate his morphological operators. The powerset of \mathbb{G} represented by \mathcal{G} forms a complete lattice. We define graph morphological operators with structuring elements on the same lattice. Following the

conventional definition, a structuring element on graphs is a *small* graph $B = (B^\bullet, B^\times)$. It may be used to identify, isolate or operate on graphs and sub-graphs that are isomorphic to the structuring element.

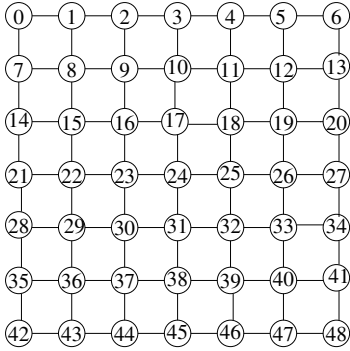


Fig. 1. The Complete Grid, which forms the workspace, \mathbb{G} . The subgraphs of this grid graph form the input topologies for the algorithms presented in this paper.

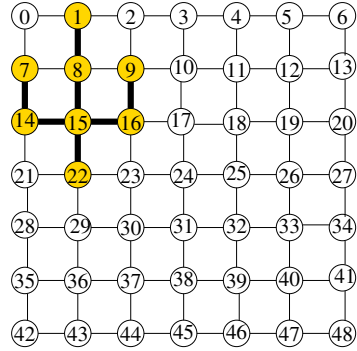


Fig. 2. Topology to illustrate the results of applying extended morphological operators that use structuring elements of Fig. 3

We define \mathcal{B} to be the set of all possible 1-hop structuring elements of \mathbb{G} . By 1-hop, we mean that a structuring element is composed of a node and the nodes adjacent to it in the grid. Thus, there are a total of 15 1-hop structuring elements – one with four edges, 4 with three edges, 6 with two edges and 4 with a single edge. These are shown in Fig. 3. \mathcal{B} is an ordered set of structuring elements where, the structuring elements are ordered in descending order of the number of edges in them. Thus, $B_1 < B_2 \Rightarrow |B_1^\times| < |B_2^\times|$.

As in the case of morphological operators defined on binary images, we translate the origin of the structuring element to each node in X^\bullet when defining the erosion and dilation operators. Given a structuring element, $B \in \mathcal{B}$ and a graph $X = (X^\bullet, X^\times)$, we define the following operators:

1. $\delta_B^\bullet : \mathcal{G}^\times \rightarrow \mathcal{G}^\bullet$ is such that:

$$\delta_B^\bullet(X^\times) = \{x \in \mathbb{G}^\bullet \mid \exists e_{x,y} \in (X^\times \cap B_x^\times)\} \tag{1}$$

2. $\delta_B^\times : \mathcal{G}^\bullet \rightarrow \mathcal{G}^\times$ is such that:

$$\delta_B^\times(X^\bullet) = \{e_{x,y} \in \mathbb{G}^\times \mid x \in (X^\bullet \cap B_x^\bullet) \text{ OR } y \in (X^\bullet \cap B_x^\bullet)\} \tag{2}$$

3. $\epsilon_B^\bullet : \mathcal{G}^\times \rightarrow \mathcal{G}^\bullet$ is such that:

$$\epsilon_B^\bullet(X^\times) = \{x \in \mathbb{G}^\bullet \mid \forall e_{x,y} \in B_x^\times, e_{x,y} \in (X^\times \cap B_x^\times)\} \tag{3}$$

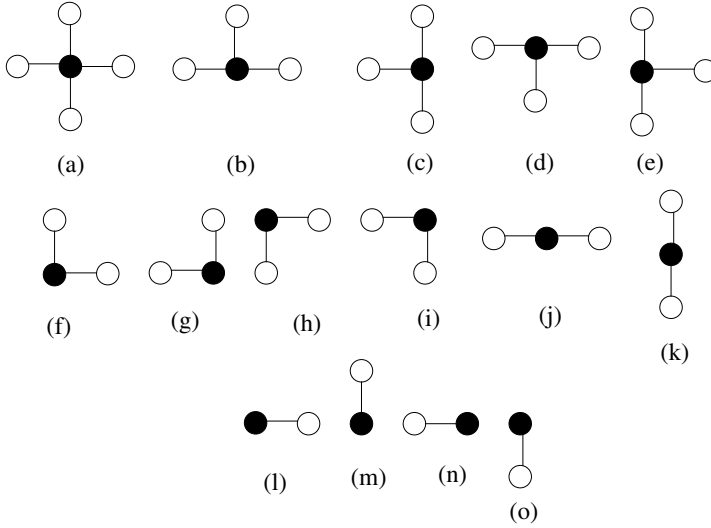


Fig. 3. The 15 structuring elements of a grid graph in descending order of the number of edges in the structuring element

4. $\epsilon_B^\times : \mathcal{G}^\bullet \rightarrow \mathcal{G}^\times$ is such that:

$$\epsilon_B^\times(X^\bullet) = \{e_{x,y} \in \mathbb{G}^\times \mid x \in (X^\bullet \cap B_x^\bullet) \text{ AND } y \in (X^\bullet \cap B_x^\bullet)\} \quad (4)$$

Operator δ_B^\bullet returns the set of all vertices that belong to the intersection of the edges of the given graph with those of the structuring element. The result of applying the structuring element given in Fig. 3(k) on the topology in Fig. 2 is shown in Fig. 4. We use the same structuring element with the other operators to illustrate the results of applying them on a given graph. The operator δ_B^\times returns the set of all edges, at least one end point of which is a member of the intersection of the vertices of the given graph and the vertices of the structuring element. The result is shown in Fig. 5. Operator ϵ_B^\bullet returns only those vertices where all the edges of the vertex in the graph are also the edges of the structuring element. ϵ_B^\times returns the set of edges where both the end points of the edge are in the intersection of the vertices of the given graph with those of the structuring element. The results for ϵ_B^\bullet and ϵ_B^\times are shown in Figs. 6 and 7.

As shown in [11], the operators δ_B^\bullet and δ_B^\times are dilations, ϵ_B^\bullet and ϵ_B^\times are erosions. If we compose the dilation and erosion operators to act on \mathcal{G}^\bullet and \mathcal{G}^\times , we get the vertex and edge dilations and erosions using structuring elements.

Vertex Dilation and Erosion. We define Vertex Dilation, δ_B and Vertex Erosion, ϵ_B that act on \mathcal{G}^\bullet (i.e., $\mathcal{G}^\bullet \rightarrow \mathcal{G}^\bullet$) by $\delta_B = \delta_B^\bullet \circ \delta_B^\times$ and $\epsilon_B = \epsilon_B^\bullet \circ \epsilon_B^\times$, where \circ is a composition and not opening.

Edge Dilation and Erosion. We define Edge Dilation, Δ_B and Edge Erosion, ε_B that act on \mathcal{G}^\times (i.e., $\mathcal{G}^\times \rightarrow \mathcal{G}^\times$) by $\Delta_B = \delta_B^\times \circ \delta_B^\bullet$ and $\varepsilon_B = \epsilon_B^\times \circ \epsilon_B^\bullet$.

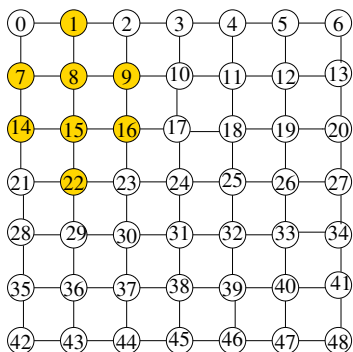


Fig. 4. δ_B^\bullet with structuring element of Fig. 3(k) applied on the graph in Fig. 2

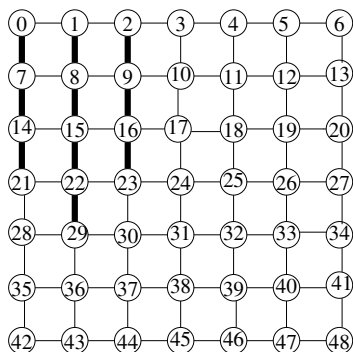


Fig. 5. δ_B^\times with structuring element of Fig. 3(k) applied on Fig. 2

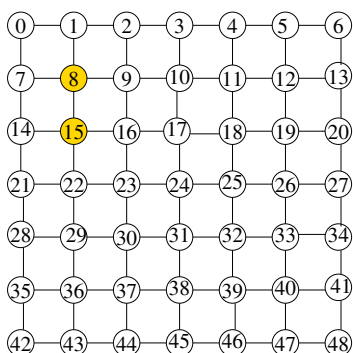


Fig. 6. ϵ_B^\bullet with structuring element of Fig. 3(k) applied on the graph in Fig. 2

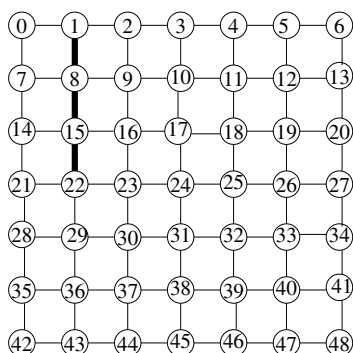


Fig. 7. ϵ_B^\times with structuring element of Fig. 3(k) applied on Fig. 2

Graph Dilation and Erosion. We can now define the graph dilation and erosion operators using structuring elements by composing the dilation and erosion operators of vertices and edges together. Thus, the graph dilation operator, $X \oplus B = \delta \oplus \Delta_B(X)$ is defined as $(\delta_B(X^\bullet), \Delta_B(X^\times))$. Similarly, the graph erosion operator, $X \ominus B = \epsilon \ominus \varepsilon_B(X)$, is defined as $(\epsilon_B(X^\bullet), \varepsilon_B(X^\times))$.

We applied our operators to compute the distance transform on images, which is presented in Section 4. We apply these operators, as well as those defined by Cousty, et. al [11], to compute dominating sets which is described in the next section.

3 Algorithms for MIDS and MDS Using Graph Morphological Operators

A dominating set of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that every node $v \in V$ is a member of S or is adjacent to a member of S . Any Maximal Independent Set (MIS)

is a dominating set. The dominating set with the minimum cardinality is called the Minimum Dominating Set (MDS). This cardinality is called the domination number, $\gamma(G)$, of the graph. An MIS with the minimum cardinality is called the Minimum Independent Dominating Set (MIDS). This cardinality is called the independent domination number, $i(G)$, of the graph. It is proven that finding MDS or MIDS is \mathcal{NP} -hard [16]. It is also proven that these are \mathcal{NP} -hard for Unit Disk Graphs and Grid Graphs [8]. Both Minimum Dominating Set and Minimum Independent Dominating Set have many applications, especially in clustering of wireless networks [7]. They have also been used in information retrieval etc. for clustering of the documents leading to multi-document summarization [26]. There are many heuristics proposed to solve MIDS, of which the maximum degree heuristic is one such [2]. The greedy algorithm for computing MDS is also the optimal approximation algorithm with an approximation ration of $\ln \Delta$, where Δ is the maximum degree in the graph [30].

The basic algorithm we propose here for both MIDS and MDS is essentially the same. The difference comes from the requirement of independence property of MIDS. The basic operation is to use the erosion operator ϵ_B^\bullet with each of the structuring elements and take the union of all the eroded nodes to construct the dominating set. We describe the algorithms for MIDS and MDS, in detail, in the next two sub-sections.

3.1 MIDS Algorithm

The MIDS algorithm $MIDS(X, Y^\bullet)$, for a given graph $X = (X^\bullet, X^\times)$ with the dominating set returned Y^\bullet is as follows:

1. For each structuring element $B \in \mathcal{B}$ do the following:
 - (a) $Z_B = \epsilon_B^\bullet(X^\times)$
 - (b) For each node $v \in Z_B$ do the following:
 - i. If $B_v \not\subset X$ where B_v is the structuring element B translated to the node v , then remove v from Z_B and go to Step 1b
 - ii. Construct the set of the dominating node, v , and its dominated nodes. This can be given as $\delta^\bullet \circ \alpha_{B_v}(X^\times)$ where, $\alpha_{B_v}(X^\times) = \delta_{B_v}^\times \circ \epsilon_{B_v}^\bullet$, where the subscript v indicates that all the operations are performed only at node v . The result is the set of edges that are incident on the node v . δ^\bullet operator from [11] returns all nodes incident on these edges, which are nothing but the dominating and the dominated nodes.
 - iii. The set of all edges incident on the dominating and dominated nodes is given by $\Delta \circ \alpha_{B_v}(X)$, where Δ is the edge dilation operator from [11]. We define $\beta_{B_v}(X) = (\delta^\bullet \circ \alpha_{B_v}(X), \Delta \circ \alpha_{B_v}(X))$ to be the subgraph of the dominating and dominated nodes and all edges incident on them.
 - iv. We need to remove the subgraph represented by $\beta_{B_v}(X)$ from the given graph X by defining the operator $\psi_{B_v}(X) = (X \setminus \beta_{B_v}(X) = (X^\bullet \setminus \delta^\bullet \circ \alpha_{B_v}(X), X^\times \setminus \Delta \circ \alpha_{B_v}(X)))$.
 - v. Update Z_B by removing all nodes that are not in $\psi_{B_v}(X)$
 - (c) $Y^\bullet = Y^\bullet \cup Z_B$
2. This leaves isolated nodes, if any. These are added to the dominating set to get the final dominating set: $Y^\bullet = Y^\bullet \cup X^\bullet$.

3.2 MDS Algorithm

The Minimum Dominating Set(MDS), $MDS(Y^\bullet, X)$ is very similar to MIDS. Note, however, that we are not iterating for the construction of MDS unlike in MIDS. We apply the operators using each structuring element on the whole graph.

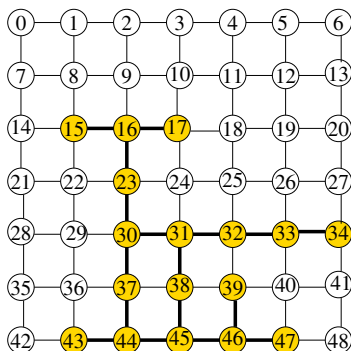


Fig. 8. Example topology to illustrate that MDS algorithm leads to redundant nodes in the dominating set

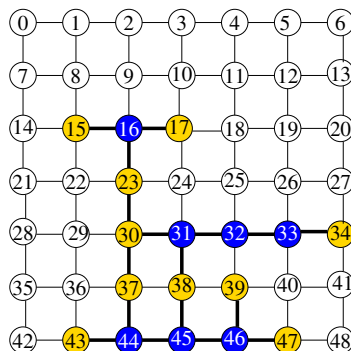


Fig. 9. Result of applying steps 1,2 of the MDS algorithm selects nodes 44, 45, 46, 16, 31, 32, 33 as the dominating set

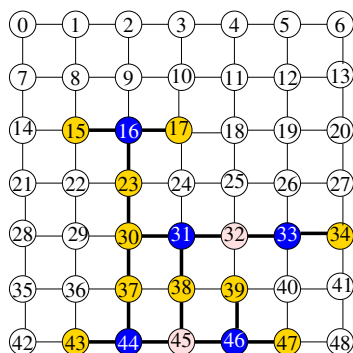


Fig. 10. Result of applying the step 3 of MDS algorithm removes nodes 32, 45 from the dominating set as they are redundant nodes

1. For each structuring element $B \in \mathcal{B}$ do the following:
 - (a) Use the erosion operator ϵ_B^\bullet with the structuring element B on X^\times . Let us say that this results in a set of nodes E^\bullet .
 - (b) If $E^\bullet = \phi$ go to the next structuring element B .
 - (c) Otherwise, $Y^\bullet = Y^\bullet \cup E^\bullet$.
 - (d) We apply the operator $\psi_B(X) = (X \setminus \beta_B(X) = (X^\bullet \setminus \delta^\bullet \circ \alpha_B(X), X^\times \setminus \Delta \circ \alpha_B(X))$ to remove the dominating and dominated nodes and the edges incident on them from the given graph X . Note that this removes many dominating nodes and their neighborhood in one iteration unlike in MIDS.

2. Add isolated nodes, if any, to the dominating set to get the final dominating set: $Y^\bullet = Y^\bullet \cup X^\bullet$.
3. We, now, apply a post-processing step to remove all redundant nodes from Y^\bullet . We call a dominating node redundant, if the node and all its neighbors are covered by other dominating nodes. An example of a redundant node is shown in Fig. 9 when applying structuring elements shown in Figs. 3(b) and 3(j) on the graph shown in Fig. 8. Nodes 31, 32, 33 and 44, 45 and 46 are in Y^\bullet . However, Node 45 and all its neighbors are covered by nodes 31, 44 and 46. Similarly, 32 is covered by 31 and 33. Therefore, nodes 32 and 45 are redundant and can be removed without affecting the domination property while reducing cardinality. $\Pi(A)$ is the set of dominated nodes of set A . Redundant node removal is expressed as $Y^\bullet = Y^\bullet \setminus \{x \in Y^\bullet \mid x \in \Pi(Y^\bullet \setminus x) \text{ and } \forall e_{x,y} \in X^\times, y \in \Pi(Y^\bullet \setminus x)\}$.

4 Experimentation and Results

As part of the experimentation, we developed an incomplete grid graph generator software. We generated grid graphs with number of edges in the graphs varying from 50-1000 edges. For each edge size, we generated 20 instances and computed the average of the dominating set cardinality using our algorithms as well as the standard heuristics. The results are presented in Tables 1 and 2. We find that our algorithm for MIDS performs as well as the maximum degree heuristic in terms of cardinality.

We also find that our MDS algorithm performs on the average about 8% worse than the optimal approximation algorithm [30]. This can be explained by the fact that, before applying the next structuring element, we are removing the neighborhood of nodes selected by the erosion operator. Sometimes, this can lead to an increase in cardinality. On the other hand, not removing the neighborhood leads to an increase in cardinality in other cases.

The time taken to compute MDS and MIDS using morphological operators is much more than the heuristics as we have not optimized the implementation for performance. Efficient implementations, both sequential and parallel, have been proposed by Vincent [29] and by Géraud et al. [17] which suggest that these running times may be reduced considerably making them comparable to the other approaches.

Table 1. Cardinality (i) of MIDS returned by the Max. Deg. Heuristic and Morph. Algorithms for Grid Graph Instances

Edges	Heuristic		Morphology	
	i	t (ms)	i	t (ms)
50	15.35	0.138	15.3	352.042
100	26.5	0.2688	26.8	700.504
250	53.9	0.59785	54.05	1729.8
500	100.3	1.1482	100.05	4331.52
750	140.6	1.66625	140	6701.06
1000	181.1	2.3031	181.35	10687.9

Table 2. Cardinality (γ) of MDS returned by the Opt. Approx. Algorithm and Morph. Algorithms for Grid Graph Instances

Edges	Heuristic		Morphology	
	γ	t (ms)	γ	t (ms)
50	14.25	0.7296	14.95	168.047
100	24.45	2.17165	25.85	247.804
250	50.2	10.9223	56	399.16
500	92.85	33.0356	98.65	644.136
750	127.85	74.8657	143.4	787.486
1000	168.9	123.271	182.25	989.726

In the rest of this section, we illustrate the use of our algorithms and operators for image analysis. Firstly, we use our structuring element based erosion for computation of the distance transform. Secondly, we show two uses of dominating sets in image analysis, i.e., the dominating set as the skeleton of the image and in clustering of the image.

4.1 Distance Transform Computation Using the Extended Graph Morphology Operators

We applied our extended graph morphological operators to compute the distance transform of an image to illustrate that they are exactly identical in operation to standard morphological operators on sets of points. The extended operators defined by us can be directly used on all images modeled as graphs. As yet, our operators are defined for binary images and so we illustrate the distance transform on binary images only. Erosion operation with the structuring element shown in Fig. 3(a) computes distance transform on the graph model that is identical to the one defined on binary images as point sets.

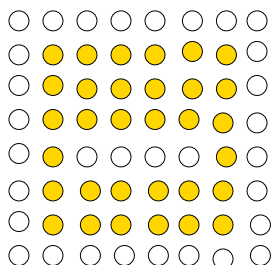


Fig. 11. Image to illustrate distance transform using ultimate erosion with structuring elements

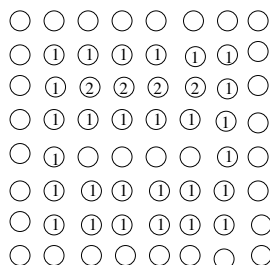


Fig. 12. Distance transform of the image, with nodes in the given graph labeled with the distance computed by the ultimate erosion using ϵ_B^\bullet with SE of Fig. 3(a)

The algorithm for computation of distance transform works as follows: we initially mark the distance of all nodes in the graph 1, to indicate they are at a minimum distance of 1 from the background pixels. We apply the ϵ_B^\bullet operator with the structuring element of Fig. 3(a) on the edge set of the given graph. The nodes returned by this operator, S , are marked to be distance 2 from the background pixels. We, now, apply the same operator on the subgraph induced on S . Any nodes returned are marked to be at a distance 3 from the background pixels. This operation is repeated until $S = \phi$. This is nothing but ultimate erosion with the structuring element given. We give an example image in Fig. 11 and its corresponding distance transform computed by our algorithm in Fig. 12.

4.2 MIDS Skeleton

In this section, we demonstrate the use of dominating sets, and in particular, MIDS, to represent the skeleton of an image. In Fig. 13, we show the image used by Gonzalez

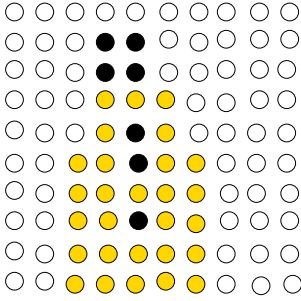


Fig. 13. Image from Gonzalez and Woods [18], with the nodes in black representing the skeleton obtained using a 3×3 structuring element

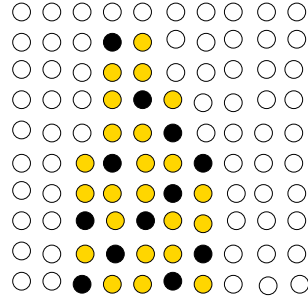


Fig. 14. Skeleton of the same image with the MIDS nodes (in black) representing the skeleton

and Woods [18] to illustrate the computation of a skeleton using ultimate erosion with structuring elements. The skeleton of the image using a 3×3 structuring element is represented by the black nodes in the figure. The skeleton represented by the MIDS of the image is shown in Fig. 14. By also storing the structuring element that caused a node to be selected as the dominating node, we can reconstruct the image by applying dilation operations. A second example is taken from [15] and is shown in Fig. 15. The skeleton obtained using the standard 3×3 structuring elements is, once again, represented by the black nodes in the figure. The skeleton represented by the MIDS of the image is given in Fig. 16, with the MIDS nodes shown in black.

The MIDS Skeleton is an interesting alternative to the skeletons based on distance transforms, Medial Axis transform and others. The other skeletons generally tend to capture the centrality of the object while the MIDS Skeleton is more uniformly distributed. The potential advantage of the MIDS Skeleton is that it may be possible to reconstruct the original graph with relatively fewer operations as the independence property ensures that any node is reconstructed from only one dominating node (i.e., from the skeleton). Further, from the examples shown, it appears that the size of the MIDS Skeleton is comparable to the other skeletons.

4.3 Hierarchical Clustering Using Dominating Sets

We present a hierarchical clustering scheme which uses an overlay of dominating sets and illustrate the scheme with a couple of examples. The algorithm is as follows:

1. Repeat the following on the given graph X until MIDS constructed has only one node or there are only disconnected components.
 - (a) Using the 4-node structuring element in Fig. 19(b) at each node y , compute $\epsilon_{B_y}^\bullet(X^\times)$. Add the node y , if it satisfies the erosion property, to the dominating set Y^\bullet . Calculate the residual graph $\psi_{B_y}(X)$. Repeat this step until there are no nodes that are part of the eroded set using this structuring element.
 - (b) Repeat Step 1a using the structuring elements of Fig. 19(c) and 19(d) on the residual graph obtained at the end of step 1a. Note that these two structuring

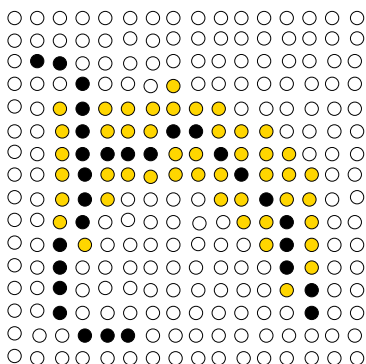


Fig. 15. Example Image [15] used to illustrate construction of a skeleton using ultimate erosion with structuring elements. The skeleton nodes are in black.

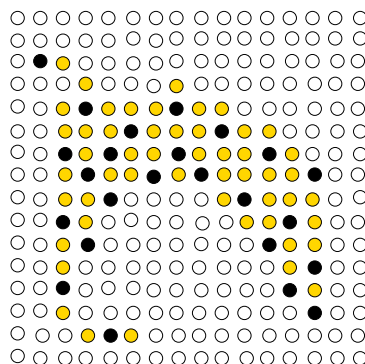


Fig. 16. Skeleton of the image with the MIDS nodes (in black) representing the skeleton of the image

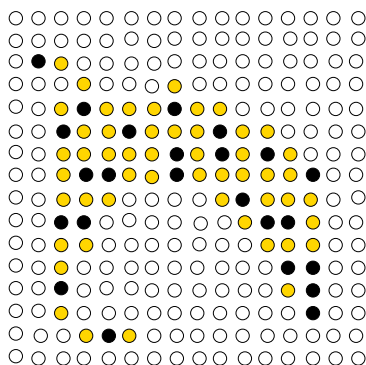


Fig. 17. Skeleton of the image with the opt.approx. MDS nodes (in black) representing the skeleton of the image

elements are applied in parallel to the residual graph of step 1a) and not in sequence. Any isolated nodes left at the end of applying each structuring element are also added to the dominating set.

- (c) We, now, construct the overlay graph, X , of the dominating nodes. For this, we add an edge $e_{u,v}$ between dominating nodes u and v , if they or their dominated nodes are adjacent to each other along the grid.

An example image is given in Fig. 18(a). The dominating nodes of the graph representing this image, after the first iteration, are shown in the darker shade. The overlay graph induced by the dominating nodes is shown in Fig. 18(b). The dominating set of this graph is shown in a lighter shade. The overlay graph of the dominating nodes of Fig. 18(b) is shown in Fig. 18(c). Finally, only the top node of the graph in Fig. 18(c) is

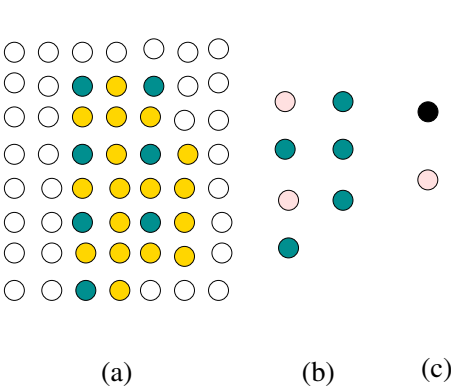


Fig. 18. An example of an image which results in a single cluster after *three* iterations of clustering on overlays of dominating sets. Iteration 1 is shown in (a), iteration 2 in (b) and iteration 3 in (c).

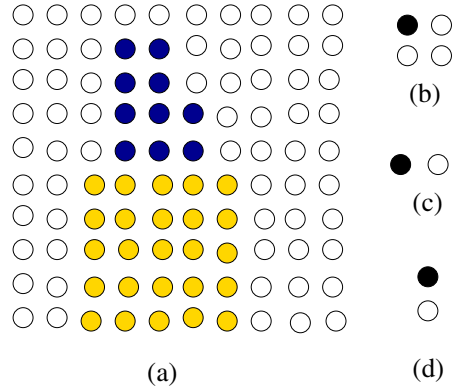


Fig. 19. Hierarchical clustering of the image shown in Fig. 13. We get *two* clusters shown in different shades in (a). The 4-node SE is shown in (b), the 2-node SEs are shown in (c) and (d).

the dominating node and the clustering algorithm halts. As the MIDS at the last overlay has only one node, it implies that the entire graph is a single cluster. Another example is shown in Fig. 19 which shows the clusters formed for the image in Fig. 13 using this algorithm. In this case, there are two disjoint dominating nodes after *four* iterations leading to two clusters in the given image. These are represented by different shades for the two clusters. It may be seen that the MIDS clustering algorithm requires neither the number of clusters nor seed points as input. It starts by operating on 1-hop nodes and then proceeds to more and more distant nodes as it travels up the overlay graphs. In this sense, it is analogous to the algorithms that use a distance threshold for clustering.

5 Conclusions and Future Work

In this paper, we extended the dilation and erosion operators defined on graphs in [11] to use structuring elements. Using the ordered set of structuring elements, we have designed algorithms to compute Minimum Dominating Set (MDS) and Minimum Independent Dominating Set (MIDS). We find that the cardinality returned by our algorithms is similar to the best heuristics in the literature. We applied the graph morphological operators on various tasks in image analysis and showed that they provide a sufficiently interesting alternative to the methods proposed in literature. In future, we plan to explore the use of dominating sets in other aspects of image analysis. We also wish to define granulometries for more intuitive approaches to hierarchical clustering.

References

1. Acton, S.T.: Fast algorithms for area morphology. *Digital Signal Processing* 11(3), 187–203 (2001)
2. Basagni, S.: Distributed clustering for ad hoc networks. In: *Proceedings of ISPAN 1999 International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 310–315 (1999)

3. Bloch, I.: Lattices of fuzzy sets and bipolar fuzzy sets and mathematical morphology. *Information Sciences* 181, 2002–2015 (2011)
4. Bloch, I., Bretto, A.: *Mathematical Morphology on Hypergraphs: Preliminary Definitions and Results*. In: Domenjoud, E. (ed.) *DGCI 2011*. LNCS, vol. 6607, pp. 429–440. Springer, Heidelberg (2011)
5. Bondy, A., Murty, U.: *Graph Theory*. Springer, Berlin (2008)
6. Chen, Y., Dougherty, E.R.: Texture Classification by Gray-Scale Morphological Granulometries. In: Maragos, P. (ed.) *Visual Communications and Image Processing 1992*. SPIE, vol. 1818, pp. 931–942 (1992)
7. Chen, Y.P., Liestman, A.L., Liu, J.: Clustering algorithms for ad hoc wireless networks. *Ad Hoc and Sensor Networks*, 145–164 (2006)
8. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. *Discrete Mathematics*, 165–177 (1990)
9. Comer, M.L., Delp, E.J.: Morphological operations for color image processing. *J. Electronic Imaging* 8(3), 279–289 (1999)
10. Cousty, J., Bertrand, G., Couprie, M., Najman, L.: Collapses and Watersheds in Pseudomanifolds. In: Wiederhold, P., Barneva, R.P. (eds.) *IWCIA 2009*. LNCS, vol. 5852, pp. 397–410. Springer, Heidelberg (2009)
11. Cousty, J., Najman, L., Serra, J.: Some Morphological Operators in Graph Spaces. In: Wilkinson, M.H.F., Roerdink, J.B.T.M. (eds.) *ISMM 2009*. LNCS, vol. 5720, pp. 149–160. Springer, Heidelberg (2009)
12. Dias, F., Cousty, J., Najman, L.: Some morphological operators on simplicial complex spaces. In: *Proceedings of the 16th IAPR International Conference on Discrete Geometry for Computer Imagery, DGCI 2011*, pp. 441–452 (2011)
13. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. *Int. J. Comput. Vision* 59(2), 167–181 (2004)
14. Felzenszwalb, P.F., Zabih, R.: Dynamic programming and graph algorithms in computer vision. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(4), 721–740 (2011)
15. Fisher, R.: Image processing learning resources, <http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm>
16. Garey, M., Johnson, D.S.: *Computers and Tractability, A guide to the theory of NP-Completeness*. Freeman and Company, New York (1979)
17. Géraud, T., Talbot, H., Droogenbroeck, M.V.: Algorithms for mathematical morphology. In: Najman, L., Talbot, H. (eds.) *Mathematical Morphology*, pp. 323–353. John Wiley and Sons Inc. (2010)
18. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*. Addison Wesley, New York (1993)
19. Heijmans, H., Vincent, L.: Graph morphology in image analysis. In: Dougherty, E. (ed.) *Mathematical Morphology in Image Processing*, pp. 171–203. Marcel Dekker, New York (1992)
20. Maragos, P.: Pattern Spectrum and Multiscale Shape Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 701–715 (1989)
21. Meyer, F., Stawiaski, J.: Morphology on Graphs and Minimum Spanning Trees. In: Wilkinson, M.H.F., Roerdink, J.B.T.M. (eds.) *ISMM 2009*. LNCS, vol. 5720, pp. 161–170. Springer, Heidelberg (2009)
22. Nieberg, T., Hurink, J.L.: A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs. In: Erlebach, T., Persinao, G. (eds.) *WAOA 2005*. LNCS, vol. 3879, pp. 296–306. Springer, Heidelberg (2006)
23. Popev, A.T.: Morphological operations on fuzzy sets. In: *Proceedings of Fifth International Conference on Image Processing and its Applications*, pp. 837–840 (1995)
24. Sartor, L.J., Weeks, A.R.: Morphological operations on color images. *J. Electron. Imaging* 10, 548–559 (2001)

25. Serra, J.: *Image Analysis and Mathematical Morphology*. Academic Press, New York (1982)
26. Shen, C., Li, T.: Multi-document summarization via the minimum dominating set. In: *Proceedings 23rd International Conference on Computational Linguistics (Coling 2010)*, pp. 984–992 (August 2010)
27. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, pp. 731–737 (1997)
28. Sternberg, S.R.: Grayscale morphology. *Comput. Vision Graph. Image Process.* 35(3), 333–355 (1986)
29. Vincent, L.: Morphological algorithms. In: Dougherty, E. (ed.) *Mathematical Morphology in Image Processing*, pp. 255–288. Marcel Dekker, New York (1992)
30. Wattenhoffer, R.: Distributed dominating set approximation,
[http://www.disco.ethz.ch/lectures/ss04/distcomp/
lecture/chapter12.pdf](http://www.disco.ethz.ch/lectures/ss04/distcomp/lecture/chapter12.pdf)

Binary Image Reconstruction from Two Projections and Skeletal Information

Norbert Hantos, Péter Balázs, and Kálmán Palágyi

Department of Image Processing and Computer Graphics
University of Szeged
Árpád tér 2. H-6720, Szeged, Hungary
{nhantos,pbalazs,palagyi}@inf.u-szeged.hu

Abstract. In binary tomography, the goal is to reconstruct binary images from a small set of their projections. However, especially when only two projections are used, the task can be extremely underdetermined. In this paper, we show how to reduce ambiguity by using the morphological skeleton of the image as a priori. Three different variants of our method based on Simulated Annealing are tested using artificial binary images, and compared by reconstruction time and error.

Keywords: Binary tomography, Reconstruction, Morphological skeleton, Simulated annealing.

1 Introduction

Binary tomography [7] aims to reconstruct binary images from their projections. In the most common applications of this field, e.g., electron tomography [1,2] and non-destructive testing [3], usually just few projections of the object can be measured, since the acquisition of the projection data can be expensive or damage the object. Moreover, the physical limitations of the imaging devices make it sometimes impossible to take projections from numerous angles. Owing to the small number of projections the binary reconstruction can be extremely ambiguous. A common way to reduce the number of solutions of the reconstruction task is to assume that certain geometrical properties (e.g., convexity and/or connectedness) are satisfied.

In this paper we investigate a new kind of prior information, the skeleton of the image to be reconstructed. Skeleton is a region-based shape descriptor which represents the general form of binary objects [6]. One way of defining the skeleton of a 2-dimensional continuous object is as the set of the centers of all maximal inscribed (open) disks [5]. A disk is maximal inscribed if it is included in an object, but it is not contained by any other inscribed disk. The skeleton of a discrete binary image can be characterized via morphological operations [6], where disks are approximated by successive dilations of the selected structuring element that represents the unit disk. An interesting property of the morphological skeleton is that the original binary image can be exactly reconstructed from the skeletal subsets. In this work, we deal with the reconstruction problem

in which the entire morphological skeleton (instead of the individual skeletal subsets) and two projections of the original image are known.

In the reconstruction process the prior knowledge is often incorporated into an energy function, thus the reconstruction task is equivalent to a function minimization problem. There are various methods to solve that kind of problems [4,9,11]. In this paper, we show how to use Simulated Annealing (SA) for the binary reconstruction problem using two projections and the morphological skeleton. We show that, although theoretically the problem is non-unique, under some circumstances an acceptable image quality can be achieved. We propose three variants of a method to solve the above problem, based on parametric SA reconstruction.

The paper is structured as follows. In Section 2 we introduce the binary reconstruction problem, and show how to describe it as an energy minimization task. The morphological skeleton as an additional information to the reconstruction is presented in Section 3. In Section 4 we describe the problem of using skeletal information in the reconstruction and introduce the proposed algorithms to solve this task. In Section 5 we present experimental results and provide an explanation of them. Finally, we summarize our work and mention some of its possible extensions in Section 6.

2 The Two-Projection Binary Reconstruction Problem

In binary tomography the task is to reconstruct a two-dimensional binary image from a set of projections. The image can be represented by a binary matrix, and its *horizontal and vertical projection* can be defined as the vector of the row and column sums, respectively, of the image matrix. The task is now to reconstruct the binary image F from its horizontal and vertical projections, $\mathcal{H}(F)$ and $\mathcal{V}(F)$, respectively. Throughout this paper – without loss of generality – we assume square images of size $n \times n$.

The first method to solve the above problem was published in [10]. In the same work it was also showed that the solution is not always uniquely determined. Furthermore, in practical applications noisy projection data also complicates the reconstruction. A common way to overcome those problems is to transform the original task to a function minimization problem

$$f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \alpha \cdot g(\mathbf{x}) \rightarrow \min, \quad (1)$$

where $\|\cdot\|_2$ stands for the Euclidean norm, \mathbf{x} is an $n^2 \times 1$ binary vector representing the unknown image in a vector form using row-by-row traversal; $\mathbf{b} = (\mathcal{H}(F), \mathcal{V}(F))^T$ is a $2n \times 1$ vector containing the projections and \mathbf{A} is a $2n \times n^2$ binary matrix, where $a_{ij} = 1$ if and only if the pixel x_i is in relation with the j -th projection ray, 0 otherwise. The function $g(\mathbf{x})$ provides additional information, such as shape, connectivity, perimeter, etc. The lower value it takes the closer the reconstructed image to the expected one. It is multiplied by the weighting parameter $\alpha > 0$. In this paper we show how to use morphological skeleton as additional information.

3 Morphological Skeleton

The morphological skeleton $S(F, Y)$ of a discrete set of points $F \subset \mathbb{Z}^2$ determined by a structuring element $Y \subset \mathbb{Z}^2$ consists of the centers of all maximal inscribed discrete disks of radius k ($k = 0, 1, \dots$) [6]. With this approach, the structuring element Y is assumed to be the unit disk (i.e., a disk of radius 1) and the discrete disk Y^k of radius k is derived from Y by successive dilations:

$$Y^k = (\dots (\underbrace{\{\mathcal{O}\} \oplus Y \oplus Y \oplus \dots}_{k\text{-times}}) \oplus Y), \tag{2}$$

where \mathcal{O} and “ \oplus ” denote origin and the fundamental morphological operation called dilation [6], respectively.

The morphological skeleton $S(F, Y)$ is defined by

$$S(F, Y) = \bigcup_{k=0}^K S_k(F, Y) = \bigcup_{k=0}^K (F \ominus Y^k) - [(F \ominus Y^{k+1}) \oplus Y], \tag{3}$$

where “ \ominus ” denotes the erosion (i.e., a morphological operation that is dual to dilation) [6], and K is the radius of the largest inscribed disk. In other words,

$$K = \max\{ k \mid F \ominus Y^k \neq \emptyset \}. \tag{4}$$

According to the formulation defined by Eq. 3, the morphological skeleton is the union of the disjoint skeletal subsets, where $S_k(F, Y)$ contains the centers of all maximal inscribed disks of radius k ($k = 0, 1, \dots, K$). An interesting property of the morphological skeleton is that a set F can be exactly reconstructed from the skeletal subsets:

$$F = \bigcup_{k=0}^K S_k(F, Y) \oplus Y^k = \bigcup_{p \in S(F, Y)} p \oplus Y^{k_p}, \tag{5}$$

where k_p is a unique value for each p such that $p \in S_{k_p}(F, Y)$.

From now we assume that structuring element Y corresponds to the 4-neighbors of the origin:

$$Y = \{ (-1, 0), (0, -1), (0, 0), (0, 1), (1, 0) \}. \tag{6}$$

Figure 1 shows an example of morphological skeleton by that Y .

4 Problem Setting and the Proposed Method

Let $H \in \mathbb{R}^n$ and $V \in \mathbb{R}^n$ be two vectors, and $S \subset \mathbb{Z}^2$ be a finite set of points. Our task is to reconstruct an image F for which $S(F, Y) = S$, and which (at least approximately) satisfies $\mathcal{H}(F) = H$ and $\mathcal{V}(F) = V$ (see Fig. 2). Unfortunately, the problem is underdetermined, as the following lemma states.

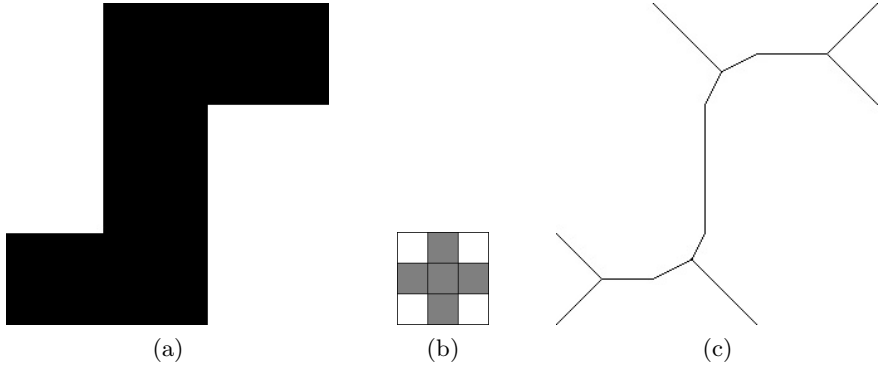


Fig. 1. Example of morphological skeleton. Original image F (a), the enlarged version of the considered structuring element Y (b), and the morphological skeleton $S(F, Y)$ (c).

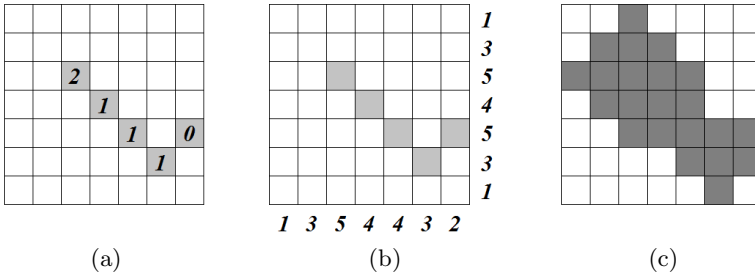


Fig. 2. Examples of two kinds of reconstruction problems. If k_p is known for each $p \in S(F, Y)$, F is uniquely reconstructable by Eq. 5 (a), the considered problem is to reconstruct F from $S(F, Y)$ and the two projections (b), image F to be reconstructed (c).

Lemma 1. *There may be some images with the same projections and morphological skeleton (i.e., the considered reconstruction problem is ambiguous).*

Proof. An example is given in Fig. 3. □

We know that for each point $p \in S(F, Y)$ there is a unique k_p value such that $p \in S_{k_p}(F, Y)$. Thus, the image F can be uniquely represented by a vector $K(S(F, Y)) = (k_{p_1}, k_{p_2}, \dots, k_{p_{|S(F, Y)|}}) \in \mathbb{Z}^{|S(F, Y)|}$. Using the notions of Eq. 1 and given a set of points S , our goal is to find a $K^*(S) = (k_{p_1}^*, k_{p_2}^*, \dots, k_{p_{|S|}}^*)$ which corresponds to the image F^* generated by Eq. 5 such that $f(\mathbf{x}^*) = \|\mathbf{Ax}^* - \mathbf{b}\|_2^2$ is minimal. Here, \mathbf{x}^* is the column vector representing F^* . Figure 4 shows an example. Note that even if there is no F such that $S = S(F, Y)$ and the function value of f is zero (e.g. in case of noisy data), it is still possible to give a solution, whose projections are close to the required ones.

The following lemma gives an upper bound for each element of $K(S(F, Y))$ of an arbitrary binary image F .

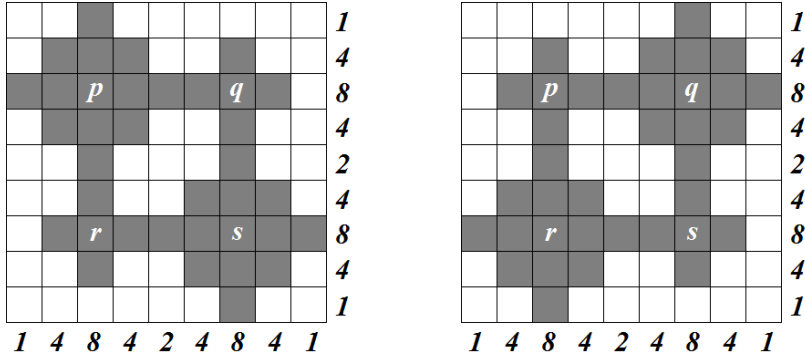


Fig. 3. Two different images F_1 and F_2 having the same projections and morphological skeleton, where $S(F_1, Y) = S(F_2, Y) = \{p, q, r, s\}$

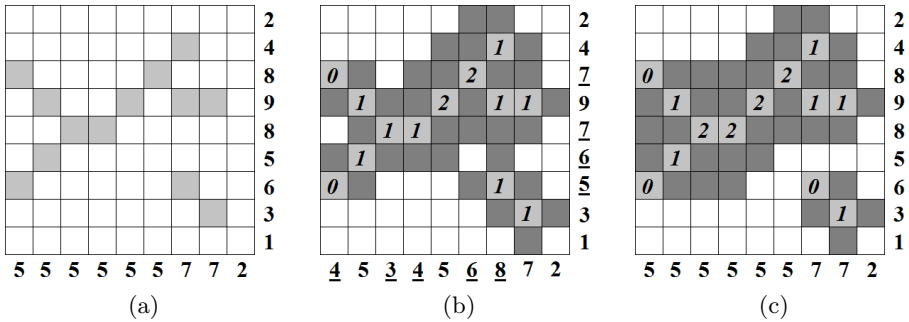


Fig. 4. Example of the studied reconstruction problem. The skeleton S and the required projections H and V (a), a possible solution F with $\mathcal{H}(F)$ and $\mathcal{V}(F)$ given by some $K(S)$ (b), the optimal solution F^* with $H = \mathcal{H}(F^*)$ and $V = \mathcal{V}(F^*)$ given by $K^*(S)$ (c). Projection elements that differ from the required ones are shown underlined.

Lemma 2. Let F be a binary image of size $n \times n$ and $K(S(F, Y)) = (k_{p_1}, k_{p_2}, \dots, k_{p_{|S(F, Y)|}}) \in \mathbb{Z}^{|S(F, Y)|}$. Then $k_{p_i} \leq n/2$ for each $i = 1, \dots, |S(F, Y)|$.

Proof. From Eq. 4 we know that the maximum value of $K(S(F, Y))$ is $\max\{k \mid F \ominus Y^k \neq \emptyset\}$. Since the size of the structuring element Y is 3×3 , it follows that $F \ominus Y^{n/2} = \emptyset$. Thus, the possible maximum value in $K(S(F, Y))$ is $n/2$. \square

Since the size of the image is known, the searching space is bounded by Lemma 2. The following lemma defines a sharper upper bound.

Lemma 3. For any skeletal set of points S and for each $p = (i, j) \in S$ with the corresponding $k_{i,j} \in K(S)$

$$k_{i,j} \leq \min \left\{ i - 1, j - 1, n - i, n - j, \frac{h_i}{2}, \frac{v_j}{2} \right\},$$

where h_i and v_j is the corresponding horizontal and vertical projection value, respectively.

Proof. It is trivial due to the size of the image and the size of $Y^{k_i,j}$. □

Lemma 2 and Lemma 3 define a unique maximum value for each $k_p \in K(S(F, Y))$. Additionally, we can use the following theorem for further reducing the searching space.

Theorem 1. *Let $S(F, Y)$ be the morphological skeleton of F generated with the structuring element Y defined by Eq. 6. Let $p, q \in S(F, Y)$, where $\|p - q\|_2 \leq \sqrt{2}$ (i.e., p and q are 8-adjacent to each other) and $p \in S_i(F)$, $q \in S_j(F)$ defined by Eq. 3. Then $|i - j| \leq 1$.*

Proof. Indirectly assume that $|i - j| > 1$, e.g., $i \geq j + 2$. We know from Eq. 3 that $q \in (F \ominus Y^j)$ but $q \notin (F \ominus Y^{j+1})$. Similarly $p \in (F \ominus Y^i)$. However, in that case $p \in (F \ominus Y^{j+2})$. Since p and q are 8-adjacent to each other, this also means that $q \in (F \ominus Y^{j+1})$, which is a contradiction. The case $j \geq i + 2$ can be seen analogously. □

Informally, if two skeletal points, p and q are 8-adjacent, then $|k_p - k_q| \leq 1$, if the structuring element is Y mentioned before. This can significantly reduce the searching space if the skeleton contains numerous pairs of 8-adjacent points.

There are numerous methods for solving Eq. 1. Since the function f is discrete and has many local minima, we propose to use Simulated Annealing (SA) [8]. Perhaps the most important advantage of the SA over the competitive methods is that it can guarantee a near optimal solution in a reasonable time. On the other hand, one serious drawback of the method is that one has to fine-tune many parameters to achieve an acceptable approximation of the global minimum of f . See Alg. 1 for the pseudo-code for SA.

The energy function f is simply $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$, where \mathbf{x} is defined by F . The goal is to find $K^*(S)$ which describe an image \mathbf{x}^* where $f(\mathbf{x}^*)$ is minimal, i.e., it has the lowest energy. We know that if $f(\mathbf{x}_1) < f(\mathbf{x}_2)$, then the image F_1 is better than F_2 in the sense that its projections are closer to the required ones, therefore function $f(\mathbf{x})$ is a proper energy function. $T(t)$ is the temperature function or the cooling schedule, such that $T(0)$ is positive, and $T(t) \rightarrow 0$ as $t \rightarrow \infty$.

We choose the following exponential function

$$T(t) = T_0 \cdot \left(\frac{T_s}{T_0}\right)^{t/M},$$

where t denotes time, so the temperature will decrease over time, T_0 is the chosen value for the starting temperature and T_s is a technical parameter controlling the shape of the cooling schedule. We empirically established the starting temperature $T_0 = 10$ and the technical parameter $T_s = 0.001$. In each iteration step the time t is increased by 1. The process terminates when reaching M the maximal number of allowed iterations or zero energy.

Algorithm 1. Simulated Annealing on the Introduced Problem

Input: projections H and V , set of skeletal points S and starting position $K_0(S)$
Output: $K(S)$
 $K(S) \leftarrow K_0(S)$
 $t \leftarrow 0$
repeat
 $K'(S) \leftarrow \text{MODIFY}(K(S))$

 Calculate x' and x from $K'(S)$ and $K(S)$, respectively

if $f(\mathbf{x}') < f(\mathbf{x})$ **or** $\text{RAND} < \exp\left(\frac{f(\mathbf{x}) - f(\mathbf{x}')}{T(t)}\right)$ **then**
 $K(S) \leftarrow K'(S)$
end if
 $t \leftarrow t + 1$
until the termination criterion is satisfied

RAND is a floating point number taken in each iteration from a uniform random distribution ($0 \leq \text{RAND} \leq 1$). With the function MODIFY we alter a state to another one simply by choosing a $k_p \in K(S)$ randomly and updating its value between the corresponding bounds. For the initial solution we choose the k_p -s such that the initial image satisfies Theorem 1 and its projections are close to the required ones. We developed three different strategies for the reconstruction:

1. *No Vase Constraint* (NVC): In the SA modification step, we choose a k_p randomly, and change it randomly between its bounds, omitting Theorem 1.
2. *Dynamic Vase Constraint* (DVC_C): We apply Theorem 1 in the following way: in each step, we modify a randomly chosen k_p by defining its new value such that $|k_p - k_q| \leq C$ holds for each q 8-adjacent to p . If $C = 1$, we allow only those differences that mentioned in Theorem 1. Because it also means slow convergence during iterations, we allow higher C values in the beginning of the reconstruction, and decrease C through time. For that we use a function $C(t)$, which is similar to the cooling schedule:

$$C(t) = \left\lceil C_0 \cdot \left(\frac{C_s}{C_0}\right)^{t/M} \right\rceil,$$

where $\lceil \cdot \rceil$ denotes the ceil function, C_0 is the starting parameter, so $C(0) = C_0$, C_s is a technical parameter established to 0.15 explicitly. Note that $C(t) \rightarrow 1$ as $t \rightarrow M$, so we force SA to search a solution that satisfies Theorem 1 as much as possible.

3. *Combined Energy Function* (CEF_α): We incorporate the constraints of Theorem 1 by using an extended energy function:

$$f(\mathbf{x}) = \alpha \|\mathbf{Ax} - \mathbf{b}\|_2^2 + (1 - \alpha)g(\mathbf{x}),$$

where α is a weighting parameter ($0 \leq \alpha \leq 1$),

$$g(\mathbf{x}) = \sum_{\|p-q\|_2 \leq \sqrt{2}} h(k_p, k_q) \quad (p, q \in S, k_p, k_q \in K(S)),$$

and

$$h(k_p, k_q) = \begin{cases} 0 & \text{if } |k_p - k_q| \leq 1 \\ |k_p - k_q|/2 & \text{otherwise.} \end{cases}$$

Note that if a solution F satisfies Theorem 4, then $g(\mathbf{x}) = 0$. In case of $\alpha = 1$, this method is equivalent to the No Vase Constraint method (i.e. $\text{CEF}_1 = \text{NVC}$).

5 Results

5.1 Implementation Details

For testing our proposed methods we developed a general reconstruction framework. For initialization, one has to specify the initial temperature T_0 , the technical parameter T_s , the maximal number of allowed iterations and the initialization method. Some of the solving methods could have additional parameters, such as α or C_0 . Certain parameters were fixed, since they are not really relevant to the efficiency of the methods, such as C_s or the structuring element Y . We also fixed the cooling schedule, which is empirically established. The test was running under Windows 7 on an Intel Core 2 Duo T2520 of 1.5 GHz PC with 2GB of RAM.

5.2 Experimental Results

We tested our algorithm on many images, in this paper we show eight samples of them. Six of our test samples have one point thin morphological skeleton consisting of few 8-connected components. However, we also show two other images which have more complex skeletons. All of the test images have the size of 256×256 .

Since SA is a randomized algorithm, we performed each test 5 times and measured the mean CPU time and errors of the reconstruction. For the numerical evaluation of the quality of the reconstructed images, we calculated



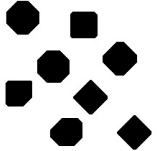


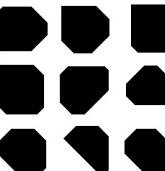

$$E = \|\mathbf{b} - \mathbf{b}'\|_2 ,$$

where \mathbf{b} and \mathbf{b}' are the projection vectors of the original and the reconstructed image, respectively. For all tests, we set $T_0 = 10$, $T_s = 0.001$ and $M = 50\,000$.

First, we tested the images containing just one convex object (see the first row of Table 1). An example for the reconstruction is shown in Fig. 5. We found that the results were mostly smooth enough, and 50 000 iterations were more than enough to converge to such a reconstructed image. All three methods provided good results, and DVC turned out to be the best choice. In one case, with certain parameters we could even perfectly reconstruct the original image in all 5 runs, using only 21 220 iterations on average.

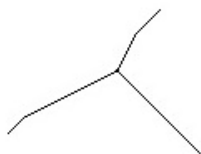
In the second turn we studied the images of convex objects arranged in a 2×2 and a 3×3 array (second row of Table 1). We observed that the initial state misled the DVC algorithm in one of the images. The main reason is that the initial image is very dissimilar to the original one, and DVC converges very slowly.

Table 1. Reconstruction results. CPU values are in milliseconds and E values are rounded to integers. Best results are highlighted.

Image	Method	CPU	E
	NVC	3842	1060
	DVC ₁₀	4030	98
	DVC ₅	4116	97
	DVC₁	4563	18
	CEF _{0.3}	4358	2468
	CEF _{0.5}	4415	1675
	CEF _{0.7}	4435	1305
	NVC	3784	3405
	DVC₁₀	3038	1291
	DVC ₅	3164	4288
	DVC ₁	3566	5307
	CEF _{0.3}	5412	5665
	CEF _{0.5}	5387	4829
	CEF _{0.7}	5328	3212
	NVC	1666	1341
	DVC₁₀	1215	292
	DVC ₅	1234	314
	DVC ₁	1302	294
	CEF _{0.3}	2904	2534
	CEF _{0.5}	2827	1950
	CEF _{0.7}	2851	1732
	NVC	3537	2530
	DVC ₁₀	2852	9154
	DVC ₅	2981	13138
	DVC ₁	3226	67493
	CEF _{0.3}	6380	5183
	CEF _{0.5}	6367	4102
	CEF _{0.7}	6343	3029
	NVC	2757	4034
	DVC ₁₀	2304	4523
	DVC ₅	2467	7472
	DVC ₁	2430	13096
	CEF _{0.3}	8884	6663
	CEF _{0.5}	8856	5012
	CEF _{0.7}	8959	4407
	NVC	4346	6136
	DVC ₁₀	4733	1066145
	DVC ₅	4609	1722350
	DVC ₁	4926	3302481
	CEF _{0.3}	7308	14371
	CEF _{0.5}	7243	8896
	CEF _{0.7}	7222	7402
	NVC	2165	2709
	DVC ₁₀	1713	6042
	DVC ₅	1724	7962
	DVC ₁	1910	6360
	CEF _{0.3}	4123	5688
	CEF _{0.5}	4131	4178
	CEF _{0.7}	4114	3346



(a)



(b)



(c)

Fig. 5. A test image (a), its morphological skeleton (b), one of the reconstructed images with CEF_{0.5} (c)

The third group of test data contained images consisting of convex objects forming random groups (third row of Table II). For the first image, the results are similar to the first group’s results, even if there are more skeletal points now which yields a bigger searching space. However, for the second image NVC produced the best results.

Finally, we examined some images that have many skeletal points with few connections (fourth row of Table II). An example reconstruction result can be seen in Fig. 6. One of the reasons for the poor results could be the skeleton, which contains many isolated pixels. It makes the method slow and ambiguous due to the large searching space. Here, NVC proved to be the best choice, since it does not use Theorem I, yielding the most robust approach of all. Although even this method could reach just a rough approximation of the original object, the result is quite promising – regarding that just two projections were used.

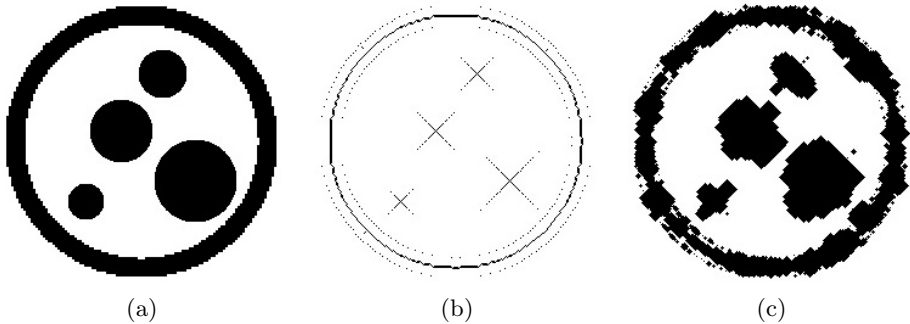


Fig. 6. A test image (a), its morphological skeleton that contains numerous isolated pixels (b), and one of the reconstructed images with NVC (c)

6 Conclusions and Further Work

We proposed three variants of a method based on Simulated Annealing to reconstruct binary images from their horizontal and vertical projections and their morphological skeleton. Without assuming 8-connected morphological skeletons, a rough reconstruction is always possible in a short time and a small number of iterations. With additional restrictions the result will be smoother, however, the convergency of the method becomes slower. The No Vase Constraint method provides overall satisfactory results, however, the Dynamic Vase Constraint creates smoother results in most cases, but needs more iterations to converge. The Combined Energy Function method is just slightly worse than the first method, but much slower. Beside that, in all the three considered methods we found that the result is much more dependent on the number of the skeletal points, rather than on the size of the image.

This paper is just an introduction of a novel approach and there are many open questions in the field. Since SA is rather sensitive to the initial state, in a

further work, we want to apply further strategies for choosing a starting image, e.g., by using Ryser's algorithm to obtain an initial solution. Beside that, we try to find a more sophisticated function minimizer, or redefine our energy function in a way that it could be managed with deterministic mathematical tools – however, this seems to be a hard task. We assume that the problem is NP-hard. We also plan to examine the efficiency of the methods using more projections and other prior information, such as smoothness on the boundary. Finally, we also intend to study the robustness of the reconstruction when the projections are corrupted by noise.

Acknowledgements. This work was supported by the European Union and the European Regional Development Fund under the grant agreements TÁMOP-4.2.1/B-09/1/KONV-2010-0005 and TÁMOP-4.2.2/B-10/1-2010-0012. The research was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the OTKA PD100950 project of the National Scientific Research Fund.

References

1. Aert, S.V., Batenburg, K.J., Rossell, M.D., Erni, R., Tendeloo, G.V.: Three-dimensional atomic imaging of crystalline nanoparticles. *Nature* 470, 374–377 (2011)
2. Batenburg, K.J., Bals, S., Sijbers, J., Kuebel, C., Midgley, P.A., Hernandez, J.C., Kaiser, U., Encina, E.R., Coronado, E.A., Tendeloo, G.V.: 3D imaging of nanomaterials by discrete tomography. *Ultramicroscopy* 109(6), 730–740 (2009)
3. Baumann, J., Kiss, Z., Krimmel, S., Kuba, A., Nagy, A., Rodek, L., Schillinger, B., Stephan, J.: Discrete tomography methods for nondestructive testing. In: Herman, G.T., Kuba, A. (eds.) *Advances in Discrete Tomography and Its Applications*, pp. 303–331. Birkhäuser, Basel (2007)
4. Di Gesù, V., Lo Bosco, G., Millonzi, F., Valenti, C.: A Memetic Algorithm for Binary Image Reconstruction. In: Brimkov, V.E., Barneva, R.P., Hauptman, H.A. (eds.) *IWCIA 2008. LNCS*, vol. 4958, pp. 384–395. Springer, Heidelberg (2008)
5. Giblin, P., Kimia, B.B.: A formal classification of 3D medial axis points and their local geometry. *IEEE Trans. Pattern Analysis and Machine Intelligence* 26(2), 238–251 (2004)
6. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*, 3rd edn. Prentice Hall (2008)
7. Herman, G.T., Kuba, A. (eds.): *Advances in Discrete Tomography and Its Applications*. Birkhäuser, Boston (2007)
8. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. *Science* 220, 671–680 (1983)
9. Nagy, A., Kuba, A.: Parameter settings for reconstructing binary matrices from fan-beam projections. *Journal of Computing and Information Technology* 14(2), 100–110 (2006)
10. Ryser, H.J.: Combinatorial properties of matrices of zeros and ones. *Canad. J. Math.* 9, 371–377 (1957)
11. Schüle, T., Schnörr, C., Weber, S., Hornegger, J.: Discrete tomography by convex-concave regularization and D.C. programming. *Discrete Applied Mathematics* 151, 229–243 (2005)

Energy-Minimization Based Discrete Tomography Reconstruction Method for Images on Triangular Grid

Tibor Lukić¹ and Benedek Nagy²

¹ Faculty of Technical Sciences, University of Novi Sad, Serbia
tibor@uns.ac.rs

² Faculty of Informatics, University of Debrecen, Hungary
nbenedek@inf.unideb.hu

Abstract. In this paper binary tomography on the triangular grid is addressed. We use three and six natural projections according to the structure of the grid. We propose an energy-minimization method based on the simulated annealing algorithm to reconstruct the original images. Our method is shown in four regular hexagon shaped test images of approximately 4000 pixels.

Keywords: Discrete tomography, Triangular grid, Energy-minimization method, Simulated annealing algorithm.

1 Introduction

Tomography deals with recovering images from a number of projections. From the mathematical point of view, the object corresponds to a function and the problem posed is to reconstruct this function from its integrals or sums over subsets of its domain. In general, the tomographic reconstruction problem may be continuous or discrete. In *Discrete Tomography* (DT) [9,10] the range of the function is a finite set, in practice, DT often deals with reconstructions of digital images that consist of only few number of gray levels. In addition to other, DT has a wide range of application in medical imaging [6], for example within Computer Tomography (CT), Positron Emission Tomography (PET) and Electron Tomography (ET). A special case of DT, which is called *Binary Tomography* (BT), deals with the problem of the reconstruction of a binary image.

In digital geometry and in digital image processing the digital plane/space is addressed with integer coordinates. By the widespread use of the Cartesian coordinate system usually the square (rectangular) and the cubic grid is used in two and three dimensions, respectively. However there are some important theoretical results that show the legacy of other regular grids. In the plane the hexagonal and the triangular grids are valid candidates of image processing methods. The hexagonal grid (hexagonal pixels) has some advantages, it is very simple; it has only one widely used neighborhood criterion. It is connected to the most densest packing of same size circles of the plane. The hexagonal grid can be described by

three coordinates using its symmetry. The pixels are addressed with zero-sum triplets [8]. The triangular grid has a similar symmetry (rotations by $2\pi/3$ moves the grid to itself) therefore three coordinate axes seem to be appropriate in its description (see, e.g., [18]).

BT has a wide literature on the square grid using two [4,3] three [23,24] four [2] or even more directions of projection (see e.g. [21]). The projection on two directions (i.e., by rows and columns) is the most basic problem, the extension to four direction (using diagonal directions) is also frequently used. In hexagonal grid there are three natural directions of projection [15]. On the triangular grid, the grid allows to use three basic directions [17] that can be extended to six directions by the geometry of the grid. Based on this fact in this paper we use three and six directions of projections.

Tomography is not an easy task. Actually, for two projections the reconstruction problem of discrete tomography is usually undetermined, while for more than two projection directions, the problem is NP-hard (see [7]) therefore various stochastic methods, e.g., genetic algorithms, brunch and bound, simulated annealing [19] can effectively be used ([14][17][22]) to compute acceptable solutions in reasonable time.

The structure of the paper is as follows. In the next section we recall the description of the triangular grid in a brief form. After this, we formalize the BT

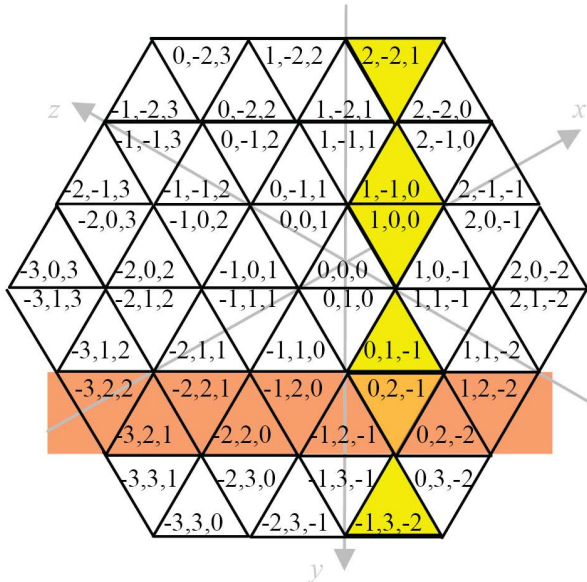


Fig. 1. Coordinate system for the triangular grid with a lane ($y = 2$) and a projection direction parallel to axis y

problem. In Section 3 we present our energy-minimization based method, while in Section 4 some experimental results are shown. Concluding remarks are given in Section 5.

2 The Triangular Grid

We use a symmetric coordinate system for the triangular grid addressing every pixel by a coordinate-triplet. The three coordinate axes have angles $2\pi/3$, the coordinate values are not independent from each other, since we are in the two dimensional plane: the sum of the coordinate values is 0 or 1 for each pixels. The pixels with zero-sum are called even pixels, they have \triangle shape, while the triangle pixels having coordinate values with one-sum are odd and their shape is ∇ . A so-called lane is obtained by fixing a coordinate value (see Fig. [1](#), red color represents the lane with $y = 2$). Every lane consists of even and odd triangle pixels alternately.

In this paper we use three projections to orthogonal to the coordinate axes. In this way the sum of the 1's by lanes are counted. Using six projections, we use directions also parallel to the coordinate axes. The pixels counted in such a projection that sums the value (i.e. 1's) of the set containing points that fulfill the equation that two of the coordinate values have a fixed difference, e.g., $\{(p(1), p(2), p(3)) | p(1) - p(3) = 1\}$ (yellow color on Fig. [1](#)).

We use images of regular hexagonal shape. A hexagon with size l contains $n = 6l^2$ triangles (for example, in Fig. [1](#) l is equal to 3).

3 Proposed Reconstruction Method

We consider a BT image reconstruction problem where the imaging process is represented by the following linear system of equations

$$Ax = b, \quad A \in R^{m \times n}, \quad x \in \{0, 1\}^n, \quad b \in R^m.$$

The matrix A is a so called projection matrix, whose each row corresponds to one projection ray, the corresponding components of vector b contain the detected m projection values, while binary-vector x represents the unknown image to be reconstructed. Each row entries $a_{i,j}$ of A represent the length of the intersection of the pixel and the projection ray passing through it. The projection value measured by a projection ray is calculated as a sum of products of the pixel's intensity and the corresponding length of the projection ray through that pixel. Projections are taken from different directions. Due to the symmetry and the distinguished directions of the triangular grid, as we already mentioned, three and six projection directions are used in this paper.

We reformulate the BT problem into an energy-minimization problem given by

$$\min_{x \in \{0, 1\}^n} E(x), \tag{1}$$

where the energy/objective function is defined by

$$E(x) = \frac{1}{2} \left(\|Ax - b\|^2 + \lambda \sum_i \sum_{j \in \mathcal{Y}(i)} (x_i - x_j)^2 \right). \quad (2)$$

The first term in (2) is called as *data fitting* term and measures the accordance of a solution with a projection data. The second term is the so called *smooth regularization* term and its rule is to enforce the coherency of the solution. Its application is based on the the prior knowledge that the original image is composed from compact regions of pixels with homogeneous intensities. By $\mathcal{Y}(i)$ we denote the set of indices of 3-neighborhood pixels of x_i , defined by the following way: if x_i is an even pixel (\triangle shape), then the $\mathcal{Y}(i)$ contains indices of neighborhood pixels in directions parallel to x - and y coordinate axes; in other case, when x_i is an odd pixel (∇ shape), the $\mathcal{Y}(i)$ contains indices of neighborhood pixels in direction parallel to z - axis. The parameter $\lambda > 0$ is the balancing parameter between data fitting and smoothing terms. For the minimization task in (II) we adapt the Simulated Annealing (SA) algorithm.

Algorithm 1. SA Algorithm

Parameters supplied by the user:

$T_{start} > 0$ {start temperature},

$T_{min} > 0$ {minimum temperature},

$T_{factor} \in (0, 1)$ {multiplicative factor for reducing the temperature},

$NoChgLimit \in \mathbb{N}$ {number of required successively reduced temperature levels without accepted change attempts}.

Initial settings:

$x = [0, 0, \dots, 0]^T$, $T = T_{start}$,

$NoChg = 0$, $E_{current} = E(x)$.

while $(T \geq T_{min}) \wedge (NoChg <= NoChgLimit)$

for $i = 1$ to $sizeof(x)$,

 choose a random position j in the vector x ;

$\tilde{x} = x$; $\tilde{x}_j = 1 - x_j$;

$E_{attempt} = E(\tilde{x})$;

$\Delta E = E_{attempt} - E_{current}$;

$z = rand(U(0, 1))$;

if $(\Delta E < 0) \vee (\text{Exp}(-\Delta E/T) > z)$, **then**

$x = \tilde{x}$; {accept change}

$E_{current} = E_{attempt}$;

$NoChg = 0$;

end if

end for

$T = T * T_{factor}$;

$NoChg = NoChg + 1$;

end while.

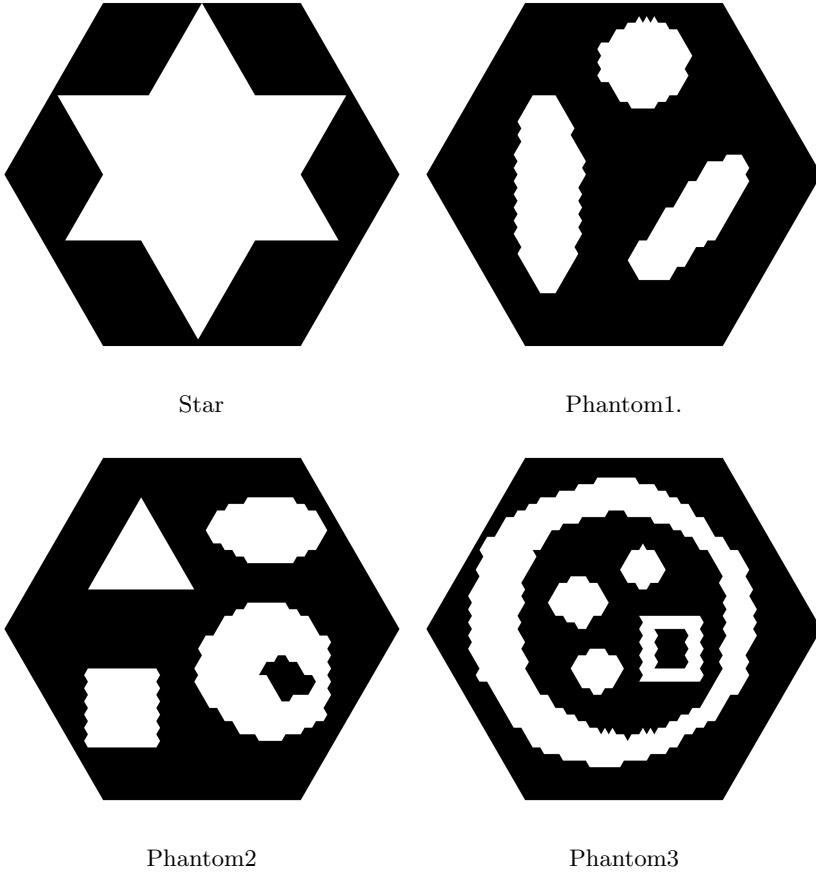


Fig. 2. Original images used in experiments. All images has the same size of 26 by 26 (regular hexagons), i.e. 4056 pixels/triangles.

Simulated annealing (SA) is a stochastic optimization algorithm based on the simulation of physical process of slow cooling of the material in a heat bath. Based on the ideas from a paper published by Metropolis et al.(1953) [16] the SA optimization algorithm is introduced by Kirkpatrick et. al.(1983) [11]. The SA algorithm starts from an arbitrary high initial temperature (control parameter) and initial solution with initial energy. The initial solution is perturbed by a small random move to a neighboring solution and the resulting change in energy, ΔE is computed. If ΔE is negative the new solution is accepted. In a case when ΔE is positive (worse attempt), the new solution is accepted with the probability given by the Boltzmann probability factor [12]:

$$e^{-\Delta E/T}.$$

This process is repeated sufficiently many times until the equilibrium state is achieved at the current temperature, T . The criterium for equilibrium state is often interpreted as a sufficiently large number of iterations taken. The temperature is then reduced and the algorithm is run on a lower temperature value. The reduced temperature value decreases the probability of the worse attempts acceptance. The entire process is repeated until the frozen state, that is, the stopping criterium is reached, which is often determined by the final temperature. The final temperature is commonly zero or close to zero.

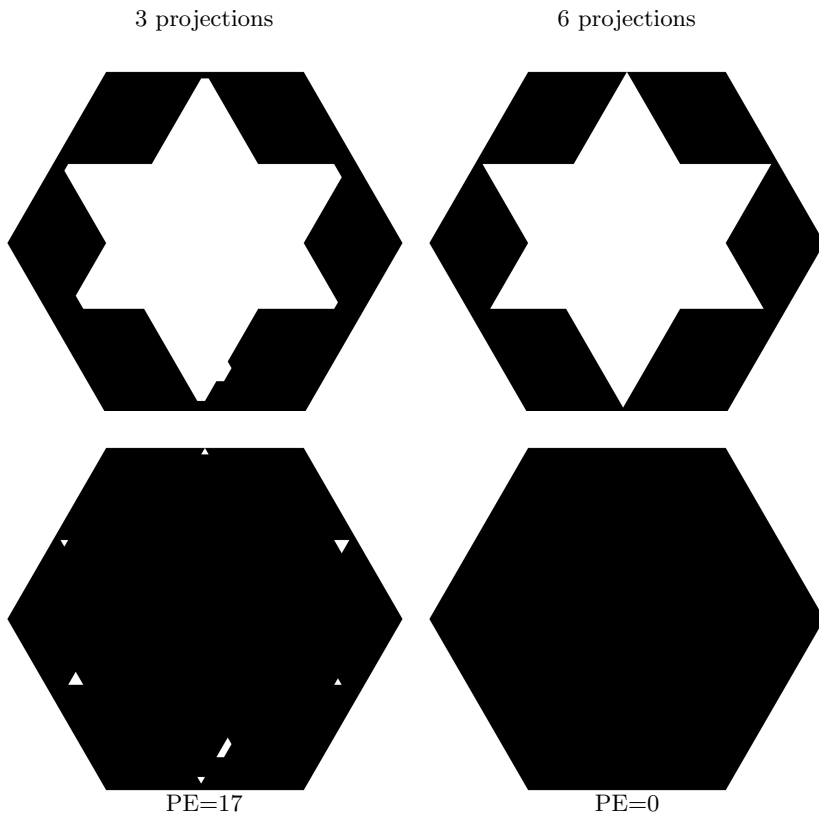


Fig. 3. First row: reconstructions of Star test image from 3 and 6 projections. Second row: difference images of reconstructions and the original image. PE denotes the number of wrongly reconstructed pixels.

Our interpretation of the SA algorithm for solving the minimization problem (II) is described in Algorithm 1. We note, that applications of the SA algorithm in similar problems [13,14,22] show its good performance. This gave us motivation to the application of SA for the minimization problem considered in this paper.

4 Experimental Results

In this paper we use test binary images of size 26 by 26 by 26 (regular hexagons). An image of this size has 4056 pixels, and therefore the size of such an image is very close to the size of a 64 by 64 square image in terms of the used pixels. The original test images are presented in Fig. 2.

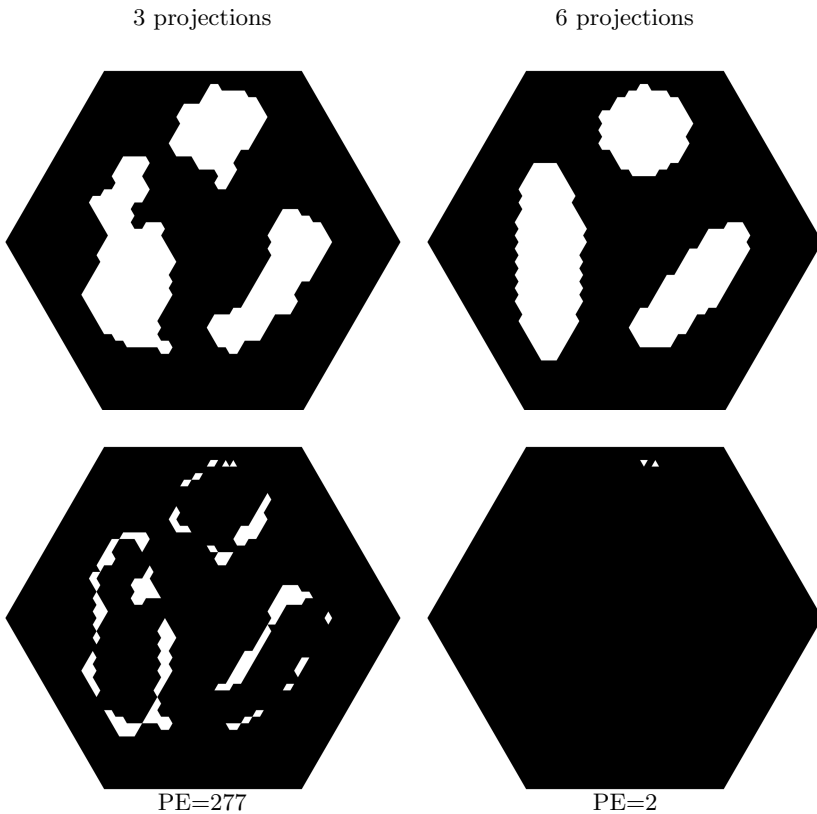


Fig. 4. Reconstruction analysis of Phantom1 test image. The layout is the same as in Fig. 3.

For SA algorithm we use the following parameter settings: $T_{start} = 4$, $T_{min} = 0.001$, $T_{factor} = 0.97$ and $NoChgLimit = 10$. The algorithm were performed without restarting. The parameter λ in energy function (2) is empirically derived and set as 5.

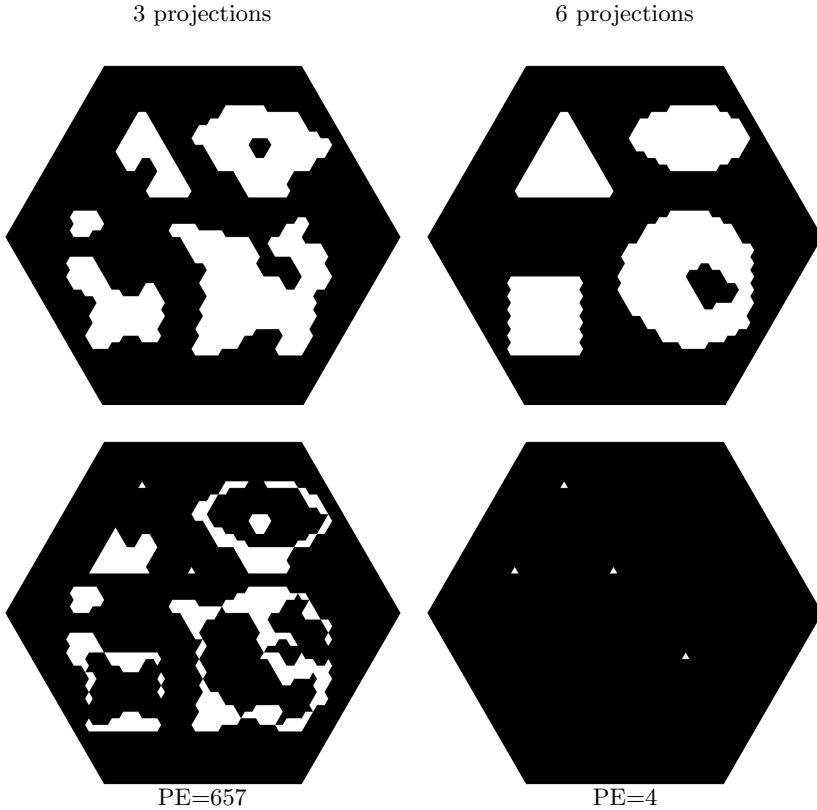


Fig. 5. Reconstruction analysis of Phantom2 test image. The layout is thee same as in Fig. 3

Reconstructions, obtained by the proposed method, are presented in Figs. 3, 4, 5 and 6. We note here that using only three projections there are various ‘switching components’ [17]. Due to them the reconstruction of a larger image almost always differ from the original projected image even if the projection values has no error. In our experiments the most complex test images, Phantom2 and Phantom3, have unacceptable bad reconstructions, see Figs. 5 and 6. However, reconstructions from 6 projection directions provide always quality reconstructions.

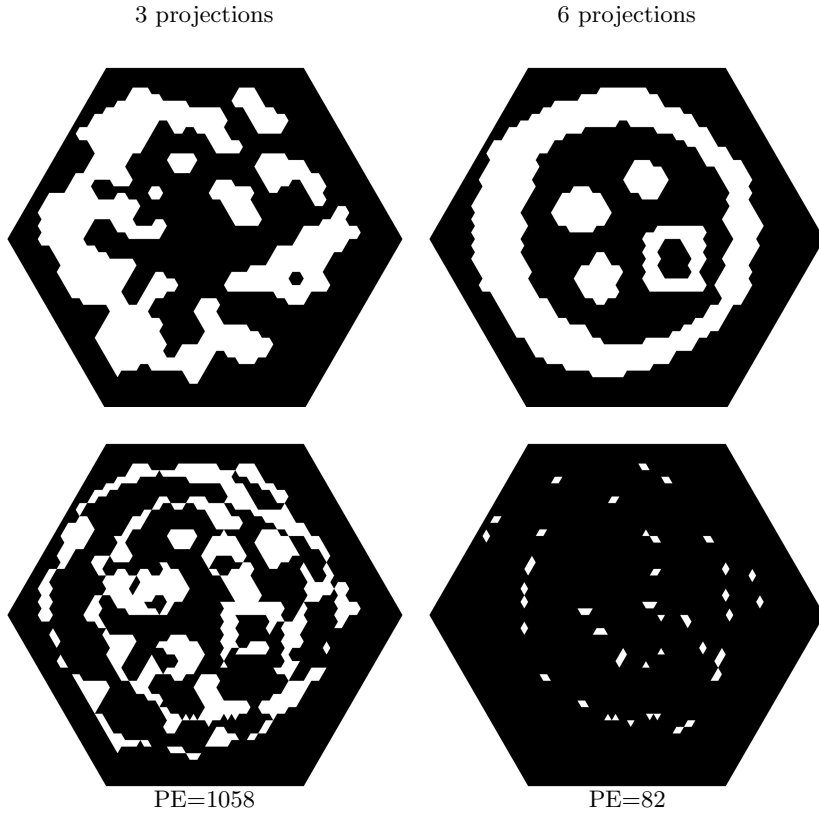


Fig. 6. Reconstruction analysis of Phantom3 test image. The layout is the same as in Fig. 3.

5 Conclusions

We have successfully developed an energy-minimization method based on the SA optimization algorithm for binary tomography reconstruction problem on triangular grid. We analyzed reconstructions from 3 and 6 projection directions. Experimental results on four phantom images confirm a capability of the proposed method for providing quality reconstructions. Reconstructions from 3 projections, using lane directions, do not provide always accurate results, but, depending on the goals and the complexity of the original, can satisfy specific requirements. It is obvious that data from 3 projection can not provide enough information for fully reconstruction. Our other conclusion, based on the obtained experimental results, is that the used 6 projection directions provides perfect or reconstructions with very small number of wrong pixel positioning.

6 Future Work

The energy function (2) of the proposed method is differentiable and convex. This can be a motivation for further work, where the nondeterministic Simulated Annealing algorithm can be replaced with a much faster, deterministic and gradient based minimization method. As possible approaches for this purpose we mention the *convex-concave regularization* [20] and *spectral gradient* based algorithms [5,13,14].

Acknowledgments. Tibor Lukić acknowledges the Ministry of Education and Science of the Republic of Serbia for support through the projects OI-174008 and III-44006. He also acknowledges support received from the Hungarian Academy of Sciences via Domus HTMT project. The work of Benedek Nagy is supported by the TÁMOP 4.2.1/B-09/1/KONV-2010-0007 project. The project is implemented through the New Hungary Development Plan, co-financed by the European Social Fund and the European Regional Development Fund.

References

1. Balázs, P.: Discrete tomography reconstruction of binary images with disjoint components using shape information. *Int. Journal of Shape Modeling* 14, 189–207 (2008)
2. Balázs, P., Gara, M.: An Evolutionary Approach for Object-Based Image Reconstruction Using Learnt Priors. In: Salberg, A.-B., Hardeberg, J.Y., Jensen, R. (eds.) SCIA 2009. LNCS, vol. 5575, pp. 520–529. Springer, Heidelberg (2009)
3. Batenburg, K.J.: A Network Flow Algorithm for Binary Image Reconstruction from Few Projections. In: Kuba, A., Nyúl, L.G., Palágyi, K. (eds.) DGCI 2006. LNCS, vol. 4245, pp. 86–97. Springer, Heidelberg (2006)
4. Batenburg, K.: An evolutionary algorithm for discrete tomography. *Discrete Applied Mathematics* 151, 36–54 (2005)
5. Birgin, E.G., Martínez, J.M., Raydan, M.: Spectral Projected Gradient Methods. In: *Encyclopedia of Optimization*, pp. 3652–3659 (2009)
6. Bronzino, J.D.: *The biomedical engineering handbook*, 3rd edn. 3 Volume Set. CRC Press (2006)
7. Gritzmann, P., Prangenberg, D., de Vries, S., Wiegelmann, M.: Success and failure of certain reconstruction and uniqueness algorithms in discrete tomography. *Int. J. Imag. Syst. Technol.* 9, 101–109 (1998)
8. Her, I.: Geometric Transformations on the Hexagonal Grid. *IEEE Transactions on Image Processing* 4, 1213–1222 (1995)
9. Herman, G.T., Kuba, A.: *Discrete Tomography: Foundations, Algorithms and Applications*. Birkhäuser (1999)
10. Herman, G.T., Kuba, A.: *Advances in Discrete Tomography and its Applications*. Birkhäuser (2006)
11. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. *Science* 220(4598), 671–680 (1983)
12. Kittel, C., Kroemer, H.: *Thermal Physics*. Freeman Co., New York (1980)
13. Lukić, T.: Discrete Tomography Reconstruction Based on the Multi-well Potential. In: Aggarwal, J.K., Barneva, R.P., Brimkov, V.E., Koroutchev, K.N., Korutcheva, E.R. (eds.) IWCIA 2011. LNCS, vol. 6636, pp. 335–345. Springer, Heidelberg (2011)

14. Lukić, T., Lukity, A.: A Spectral Projected Gradient Optimization for Binary Tomography. In: Rudas, I.J., Fodor, J., Kacprzyk, J. (eds.) *Computational Intelligence in Engineering*. SCI, vol. 313, pp. 263–272. Springer, Heidelberg (2010)
15. Matej, S., Herman, G.T., Vardi, A.: Binary tomography on the hexagonal grid using Gibbs Priors. *International Journal of Imaging Systems and Technology* 9, 126–131 (1998)
16. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21(6), 1087–1092 (1953)
17. Moisi, E., Nagy, B.: Discrete tomography on the triangular grid: a memetic approach. In: *Proc. of 7th International Symposium on Image and Signal Processing and Analysis (ISPA 2011)*, pp. 579–584. Dubrovnik, Croatia (2011)
18. Nagy, B.: Distances with neighbourhood sequences in cubic and triangular grids. *Pattern Recognition Letters* 28, 99–109 (2007)
19. Russel, S.J., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall (2002)
20. Schüle, T., Schnörr, C., Weber, S., Hornegger, J.: Discrete Tomography by Convex-concave regularization and D.C. Programming. *Discrete Appl. Math.* 151, 229–243 (2005)
21. Varga, L., Balázs, P., Nagy, A.: Direction-Dependency of a Binary Tomographic Reconstruction Algorithm. In: Barneva, R.P., Brimkov, V.E., Hauptman, H.A., Natal Jorge, R.M., Tavares, J.M.R.S. (eds.) *CompIMAGE 2010*. LNCS, vol. 6026, pp. 242–253. Springer, Heidelberg (2010)
22. Weber, S., Nagy, A., Schüle, T., Schnörr, C., Kuba, A.: A Benchmark Evaluation of Large-Scale Optimization Approaches to Binary Tomography. In: Kuba, A., Nyúl, L.G., Palágyi, K. (eds.) *DGCI 2006*. LNCS, vol. 4245, pp. 146–156. Springer, Heidelberg (2006)
23. Weber, S., Schnörr, C., Hornegger, J.: A linear programming relaxation for binary tomography with smoothness priors. *Electronic Notes in Discrete Mathematics* 12, 243–254 (2003)
24. Weber, S., Schüle, T., Hornegger, J., Schnörr, C.: Binary Tomography by Iterating Linear Programs from Noisy Projections. In: Klette, R., Žunić, J. (eds.) *IWCIA 2004*. LNCS, vol. 3322, pp. 38–51. Springer, Heidelberg (2004)

Skin Lesion Feature Vector Space with a Metric to Model Geometric Structures of Malignancy for Classification

Mutlu Mete¹, Ye-Lin Ou², and Nikolay Metodiev Sirakov^{1,2}

¹ Texas A&M University Commerce, Computer Science
{Mutlu.Mete,Nikolay.Sirakov}@tamuc.edu

² Texas A&M University Commerce, Department of Mathematics
Yelin.Ou@tamuc.edu

Abstract. This paper develops an approach aiming to identify a pattern defined by malignant dermoscopic skin lesions presented as lesion feature vectors (LFVs) in a 4D Riemannian manifold. The manifold consists of all 4-tuples and its metric is defined on the basis of the Total Dermoscopy Score (TDS) formula used in the ABCD diagnosis rule. Tools in Riemannian geometry including distance functions, directional angles, polar, cylindrical and spherical coordinates are used to study the geometric structures of a sample space defined by malignant 4D LFVs. To explore the geometric structures and the distribution pattern of the malignant LFVs, we find methods to visualize the objects in a 4D manifold by projecting them onto a 2D or 3D space via polar, generalized cylindrical and spherical coordinates. To observe malignancy identification structures in the newly constructed manifold a data-set of 70 lesion images with a ground truth were used to generate LFVs. To build a surface separating the benign and malignant LFVs a linear TDS-based and a non-linear support vector machine (SVM) classifiers were applied. The SVM is build with a polynomial kernel, whose degree and parameters were suggested by a geometric structure observed in a 3D space. A statistical comparison, on the base of experimental results, showed that the polynomial SVM has a better f-measure accuracy than the linear TDS based one.

Keywords: Skin lesion classification, Riemannian manifold, Support vector machines (SVM).

1 Introduction

The skin disease melanoma is one of the most common malignancies in the United States [1]. The problem of diagnosing skin lesion malignancy is that the entire present process heavily depends on the investigator. Currently the ABCD (Asymmetry, Boundary, Colors number, Dermoscopy structures) rule [2,3] is used in majority of the dermoscopic clinical practices, in which components of the rule are connected with a linear function, called total dermoscopy score (TDS).

In practice the TDS components are found by naked-eye investigation, which is time-consuming, operator dependent, and error-prone. In this sense, automatic, intelligent, and unbiased extraction and measurement of the components of the ABCD rule will greatly improve their assessments and help generate feature vectors for classification of benign and malignant lesions.

This paper is a natural continuation of a sequence of works presented in [4,5,11,12]. In [12] a shrinking active contour model (S-ACES) is designed. In [4,5,11] the model is applied to extract the ABCD rule [2] related features: lesion boundary; abrupt boundary; boundary of color regions; number of colors in a lesion. The boundary is used in [11] for lesion asymmetry calculation, in [5] for abrupt boundary definition and color regions asymmetry calculation. In [11] a 2D LfV space is generated while [5] extends the vectors to 4D. The study in [4] proved that the active contour presented in [12] possesses the necessary accuracy to extract lesion's boundaries.

The present work develops an approach that 1) views the 4D LfVs generated in [5,11] as points in a 4D Riemannian manifold [7,9]; 2) uses the generalized cylindrical and spherical coordinates to project 4D LfVs onto a 3D space for a purpose of visualization; 3) applies SVM and selects its kernel using knowledge derived from the 3D space to separate the geometric structures formed by malignant and benign vectors.

This paper is organized as follows. An overview of the lesion feature extraction and measurement approaches are given in Section 2. The 4D Riemannian manifold that we constructed, the basics of its geometry, as well as 2D and 3D visualizations are given in Section 3. Experimental dataset and results of experiments are presented in Section 4. Finally, Section 5 concludes this study and sheds a light on future directions.

2 Lesion Features Extraction

The first step of our framework is to extract, from images, the ABCD [8] related lesion features used to generate the LfVs. The skin lesion features subjects of extraction are: lesion boundary; lesion colored regions; number of colored regions; and boundary used to calculate abrupt lesion endings. To extract the above features the shrinking active contour model S-ACES [12] is applied.

Note that for the accuracy of skin lesion delineation, colored regions and abrupt boundary extraction are important requirements for the proper LfVs generation. To prove the S-ACES model is up to the job its accuracy has been investigated in [4] and a set of experiments has been performed in [4,11]. The boundaries of 51 skin lesions were extracted from images and the statistical properties recall, precision, accuracy and border error have been measured and compared to a ground truth.

The evolution equation of S-ACES is formulated with the following parametric vector function [12]:

$$r(s, t) = e^{as-4a^2t}(C_1 \cos(c.a.s), C_2 \sin(c.a.s)) \quad (1)$$

In Eq. (1), s is a space parameter that describes a particular curve, t is time parameter, which describes a family of curves for $t \in [0, \infty)$, $a = |ds|/2$, C_1, C_2 , and c are real constants such that $r : \mathbb{R}^2 \rightarrow \mathbb{R}^2$. The ds denotes the rate of change of s .

The initial curve circumscribe the entire image by using the following initial conditions (IC) [5][11][12]:

$$a^2t = 0.001, c = 1000, R = C_1 = C_2 = \sqrt{(nc)^2 + (nr)^2} \tag{2}$$

where, nc denotes the number of columns in the image, nr the number of rows. One may tell that Eq. (1) describes a circle with a radius $R \rightarrow 0$ if $a^2t \rightarrow \infty$. In practice R becomes a point (pixel) if $a^2t = 2.5$.

In order to extract lesion’s boundary and boundary used to measure abrupt lesion endings the following boundary conditions (BC) is applied [5]:

$$r(s, t) = r(s, t + \partial t) \text{ if } \frac{\partial f}{\partial t}(r(s, t)) > \epsilon, \text{ for } 2.5 > ta^2 > 0.001 \tag{3}$$

To define color regions and extract their boundaries, the following BC is applied along with evolution Eq. (1) and IC (2):

$$\begin{aligned} r(s, t) &= r(s, t + dt) \text{ if } f(r(s, t + dt)) \in [\epsilon_1, \epsilon_2] \\ r(s, t) &\neq (s, t + dt) \text{ otherwise.} \end{aligned} \tag{4}$$

The values of ϵ_1 and ϵ_2 represent image intensities and are selected by the user on the base of the image database under consideration, whereas $f(x, y)$ is the image function. For more details about the values selected for ϵ_1, ϵ_2 and ϵ please see [5]. Also, BCs and (3) and (4) will detect the boundaries in a noiseless images. An improvement of the boundary conditions that lets the active contour pass through noise, which area may be inscribed in a rectangle with $n \times k$ pixels, is developed in [5].



Fig. 1. Upper left to down right: image of malignant lesion; masks of light brown, blue, gray, dark brown, black, and red colors. Their boundaries were extracted by S-ACES.

In [11] the extracted lesion boundaries were used to measure the lesion’s asymmetry and calculate the lesion’s area. Thus a set of 2D LFVs was generated in this paper. Further BC (3) was applied in [5] along with an area based approach

to calculate the lesions abrupt endings, and a noise surpassing version of BC (4) to extract the boundaries of color regions.

Thus, our approach automatically extracts and measures the components of 4D LFVs $X^i = (A_B^i, A_C^i, B^i, C^i)$, where A_B denotes boundary asymmetry, A_C colors asymmetry, B abrupt boundary, and C the number of colors in a lesion, respectively. For example, the LFV for the lesion in Fig. 1 is $(0, 0.3, 5, 5)$. The totality of these LFVs $S_n = \{X^i | i = 1, 2, \dots, n\}$ is our sample space, where n is the numbers of images in our experiments.

In this paper we present the sample space S_n as a subspace of a 4D Riemannian manifold and study the geometric structures and distribution pattern of the sets defined by malignant and benign LFVs.

2.1 Asymmetry

Using the extracted lesion boundary and color regions boundary we calculate asymmetry of each lesion and the asymmetry of its color regions (A_B, A_C). For this purpose we used an axial asymmetry measure developed in [6] and modified in [5,11]. The asymmetry approach starts with finding major and minor axes, vertical and horizontal dash lines, of the lesion boundary [6] shown in Fig. 2.

Note that in the clinical practice a dermatologist manually determines the major and minor axes of the lesion [8]. In the present study we use the boundary points and calculate all distances defined by them in time $O(n^2)$, where n is the number of boundary points extracted by S-ACES. Further the maximum and minimum distances are found as major and minor axes, again in time $O(n^2)$. In the light of the ABCD instructions, we calculate asymmetry with respect to both axes and select the minimum value.

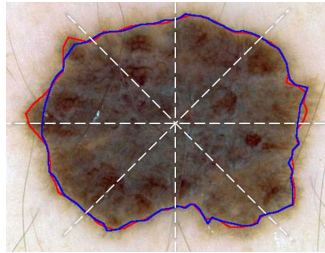


Fig. 2. Lesion boundary S_a (blue, dark in grayscale) extracted by S-ACES with BC (3) and the union of color regions S_b (red, light in grayscale)

In order to quantify asymmetry [5,11] with respect to an axis, the area of the lesion mask, R , is flipped around this axis to find overlapping areas of the regions, called true-symmetry and denoted by (T_s) ; and to find the area of the

non-overlapping regions, called false-symmetry and denoted by F_s . We calculate asymmetry by F_s/R for the major and minor axes and use a threshold to assign 1, for each axis, if symmetry value is above the threshold and 0 otherwise. Thus asymmetry is determined for the whole lesion to find A_B which takes values either 0 or 1 or 2. The criterion that applies for this feature is the larger asymmetry, the higher likelihood for melanoma.

Analogously asymmetry of each color region is calculated. The LFV component A_C is determined as the mean of the asymmetry of all color regions. For more detailed calculation of the components A_B and A_C please see [5]. Applying the approach, described above, on the boundaries extracted from the image shown in Fig. 1 we found that $A_B = 0$ and $A_C = 0.3$.

2.2 Abrupt Edges

In regard to component B , from the LFV, we developed a technique to model the necked eye abrupt lesion edges assessment, performed by dermatologists (ABDC [8]). Thus a lesion is divided into eight sectors using axial segments as shown in Fig. 2. In each segment, the method looks for a sharp, abrupt ending of lesion at the periphery. The quantification of this feature is binary-wise and is given in details in [5]. If an abrupt edge is detected, this sector contributes a score of one, otherwise zero. Therefore, the maximum abrupt border score is eight, and the minimum score is zero.

The lesion boundary is extracted by S-ACES using BC (3) (Fig. 2 - blue (dark) line). The area bounded by this line is denoted by S_a . The area enclosed by the red (light) line is denoted by S_b and determined as the union of the color regions extracted by S-ACES with BC (4).

Further, for each segment we test if $1 - \tau \leq S_a/S_b \leq 1 + \tau$ holds [5], where τ is an empirically found threshold. If this test holds for a section, we add one point to the total abruptation score.

As stated in ABCD rule, total border score in nevi is very low and in melanomas is between three and eight. Based on this calculation, for instance, the value of abrupt boundary is five for the lesion in Fig. 2.

2.3 Number of Colors

Since the presence of many colors in a lesion increases the likelihood of melanoma, the last component C of the LFVs regards number of colors seen in a lesion. As indicated by ABCD rule, major colors of the lesion are white, red, light-brown, dark-brown, blue-gray, and black. White color in the lesion should be considered significant if the area is lighter than the neighboring skin. If all six colors are found the maximum color score would be six. The minimum score is one. The lesion colors are extracted by S-ACES using evolution Eq. 1, IC (2), and BC (4), where the thresholds ϵ_1 and ϵ_2 are selected by an expert as shown in [5]. Melanomas are usually characterized by presence of three or more colors. Five colors were extracted from the lesion shown in Fig. 2.

3 The 4D Manifold and Its Metric

The main idea of this study is to view the LFBVs sample space S_n as a subspace of the Riemannian manifold (\mathbb{R}^4, g) , so that we can use the notion of distance and other tools of Riemannian geometry to study the geometric structures of the subsets defined by malignant and benign points in S_n .

We start with the weighted Euclidean metric $g(X, Y) = 1.3X_1Y_1 + 1.3X_2Y_2 + 0.1X_3Y_3 + 0.5X_4Y_4$, where the choice of the weights is suggested by the TDS formula in the ABCD diagnosis rule [8].

3.1 The Geometry of (\mathbb{R}^4, g)

We define the norm of $X = (X_1, X_2, X_3, X_4)$ by the expression

$$|X| = (g(X, X))^{\frac{1}{2}} = [1.3X_1^2 + 1.3X_2^2 + 0.1X_3^2 + 0.5X_4^2]^{\frac{1}{2}}, \tag{5}$$

which gives the distance of the terminal point of the vector X to the origin. The angle between two vectors $X = (X_1, X_2, X_3, X_4)$ and $Y = (Y_1, Y_2, Y_3, Y_4)$ in the 4D Manifold is defined by

$$\theta = \cos^{-1} \frac{g(X, Y)}{|X||Y|} \tag{6}$$

Thus, the directional angle of a vector $X = (X_1, X_2, X_3, X_4)$ is computed by

$$\theta_i = \cos^{-1} \frac{g(X, e_i)}{|X|}, \quad i = 1, 2, 3, 4, \tag{7}$$

where $e_1 = \frac{1}{\sqrt{1.3}}(1, 0, 0, 0)$, $e_2 = \frac{1}{\sqrt{1.3}}(0, 1, 0, 0)$, $e_3 = \frac{1}{\sqrt{0.1}}(0, 0, 1, 0)$, $e_4 = \frac{1}{\sqrt{0.5}}(0, 0, 0, 1)$ form an orthonormal basis of the Riemannian manifold (\mathbb{R}^4, g) .

3.2 Visualization of LFBVs in 2D and 3D Spaces

In our efforts to visualize the distribution of malignant and benign lesions in 4D Riemannian manifold we compute the distance function $\rho(X) = |X|$ of LFBVs and obtain the graph of the distance function as shown in Fig. 3.

Note that most malignant LFBVs lie in the interval $[3.5, 3.7] \cup [4.2, 4.7]$. This image result shows that the distance function is not discriminative condition because many benign vectors appear in the same intervals as given above. Therefore, the development of another tool will be an effort to geometrically distinguish between malignant and benign vectors.

A point X in xy -plane can be located by its Cartesian coordinates $X = (x, y)$ or its polar coordinates $X = (\rho, \theta)$. The relation between the Cartesian and the polar coordinates are given by:

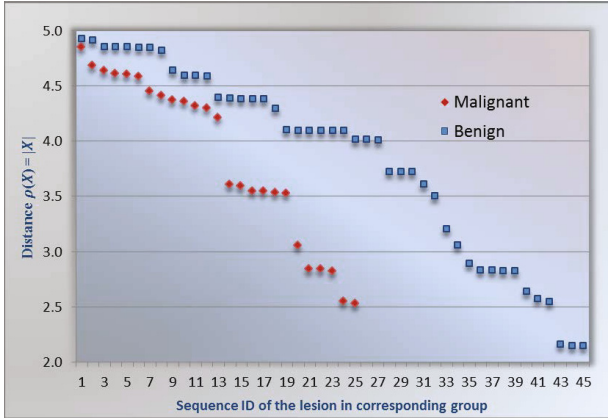


Fig. 3. Plot of the distance functions of the LFVs, $|X|$

$$\begin{aligned} x &= \rho \cos \theta, & \rho &= \sqrt{x^2 + y^2} \\ y &= \rho \sin \theta, & \theta &= \tan^{-1}\left(\frac{y}{x}\right) \end{aligned} \tag{8}$$

Further, we use the mean

$$\theta_m = \frac{(\theta_1 + \theta_2 + \theta_3 + \theta_4)}{4} \tag{9}$$

of the four directional angles of a LFV $X = (A_B, A_C, B, C)$, as its polar angle. Further we apply the distance $\rho(X) = |X|$ of X , defined by Eq. 5, to the origin in order to plot the graph of the LFVs in a generalized polar coordinate system (ρ, θ_m) . Here, the polar coordinates of a point $X = (X_1, X_2, X_3, X_4)$ are given by $X = (x, y) = (|X| \cos(\theta_m), |X| \sin(\theta_m))$, and we obtained Fig. 4.

Although many malignant points are already evident in the upper right part of Fig. 4, benign points are present there as well. To better identify malignant LFVs, we map them to a 3D space by using the generalized cylindrical coordinates $X \rightarrow (\rho(X) \cos(\theta_m), \rho(X) \sin(\theta_m), \theta_\sigma(X))$, where $\theta_\sigma(X) = \left[\frac{1}{4} \sum_{i=1}^4 (\theta_m(X) - \theta_i(X))^2 \right]$ is the variance of the directional angles. One advantage of this generalized cylindrical coordinates representation is that it allows us to plot points of a 4D space in a 3D space as seen in Fig. 5. One may observe that most of the malignant LFVs are close to the plane defined by the points (3, 0, 0) (0, 2.5, 0) and (0, 0, 0.2). The region where the malignant points are located, around the plane, is defined by the intervals $x \in [3.5, 5], y \in [1.3, 2.5]$ and $z \in [0.1, 0.25]$. This parallel piped region contains more and better separated malignant points but still benign are presented there. Except that the malignant points are located around a plane, no geometric structure is observed so far.

Thus we represent the LFVs in a spherical coordinate system

$$X \rightarrow (\rho(X) \sin \theta_m \cos \theta_\sigma, \rho(X) \sin \theta_m \sin \theta_\sigma, \rho(X) \cos \theta_m). \tag{10}$$

in order to find a geometric structure or pattern defined by malignant and/or benign vectors in the 3D space. This may prove helpful in predicting the geometric structures in the 4D manifold. The shape of this structure may be used to suggest a SVM kernel to be applied for the purpose of vectors separation to malignant and benign.

For example, as we mention above, using polar coordinates in 2D, we observed that all LFVs are located around a line. In 3D generalized cylindrical coordinates, we observed that the vectors are located around a plane. Further, a mapping of the same vectors in the 3D spherical coordinate system still kept them around a plane, where no geometric structure was formed yet. But a presentation on the unit sphere showed a quasi-polynomial curve on the sphere (see Fig. 6 (right)). Thus we observe that the curve defined by red (light gray) points have three picks, which may suggest that a fourth degree polynomial may be used as a kernel of a SVM to separate the malignant vectors from the benign ones on the unit sphere.

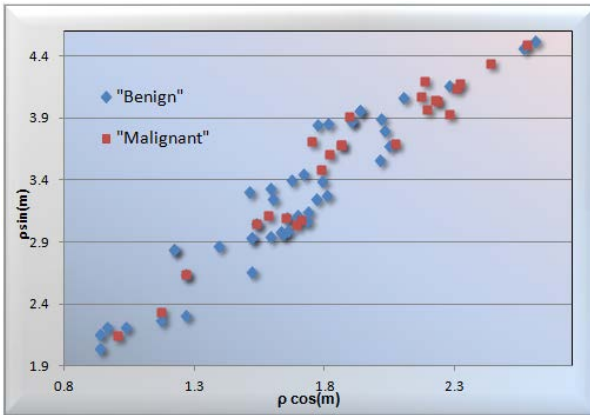


Fig. 4. LFVs in a generalized polar coordinate system

4 Experiments and Discussion

A data set of 70 lesion images selected from [3] was used for these experiments. Ground truth delineation of the lesions and diagnosis is available for each lesion. Of 70, 45 and 25 images are identified as benign and malignant respectively with the help of ground truth labels. The size of the images varies between 1891 x 1261 and 707 x 484 in pixel. In a few cases, frame black strip is removed from the images to facilitate the S-ACES evolution.

To separate malignant LFVs from benign ones we used SVM [14] in the 3D space. On the other hand, for comparison, predictions of TDS model are found in the 4D Manifold. Remember that TDS is already capable of result labels

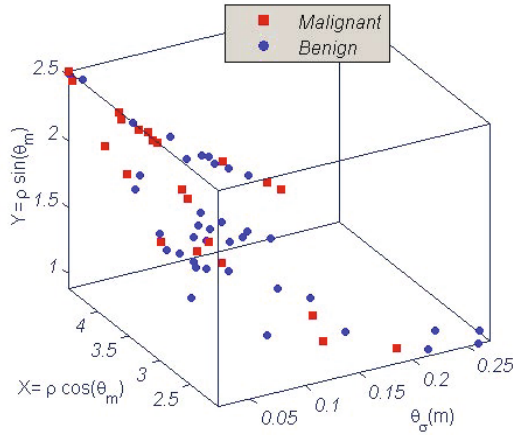


Fig. 5. LFVs in a cylindrical system

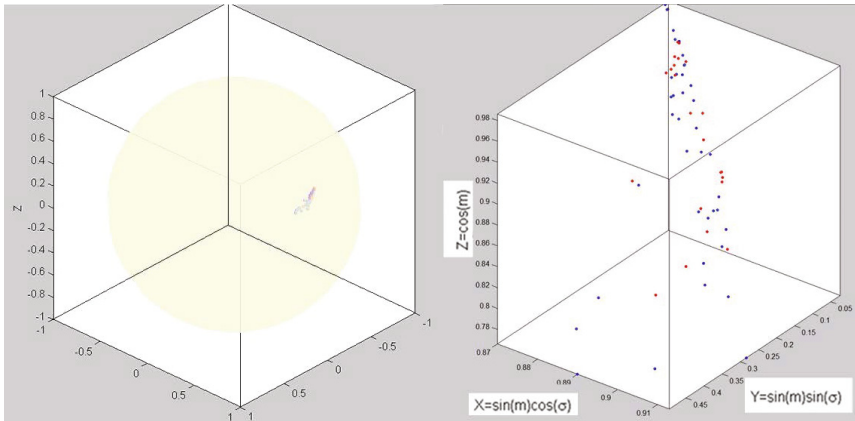


Fig. 6. (left) The 70 LFVs mapped on a unisphere; (right) zoom of the part of the unisphere where the vectors are located

for lesions. In the experiments, we compare the methods over f-measure of the model classification, which is harmonic mean of precision and recall. Recall is the percentage of positive (malignant) labeled instances that were predicted as positive and found by $TP/(TP + FN)$. Precision is defined as the percentage of positive predictions that are correct, and calculated by $TP/(TP + FP)$ ¹.

TDS applied in this study uses the feature set, A_B, A_C, B, C and the weights [1.3, 1.3, 0.1, 0.5] suggested by [8]. To calculate the values of each feature we selected the following thresholds to maximize the classification accuracy: 0.91 for lesion asymmetry, 0.88 for color asymmetry and 0.06 for abrasion. Within a

¹ TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative.

grid search over possible intervals (asymmetry $[0.7, 1]$, and abruptness threshold $[0, 2]$), the best accuracy of 0.68 was obtained with the threshold of 5.9, meaning that a lesion will be classified as malignant if $(1.3A_B + 1.3A_C + 0.1B + 0.5C) \geq 5.9$, or benign vice versa.

In the 3D space, $(\rho(X), \theta_m(X), \theta_\sigma(X))$, we employed ν -SVM [10] to build up a surface that will separate malignant from benign LFVs. ν -SVM is also a soft margin classifier using ν -parameterizations. In this approach, the classification problem is solved by ν -SVM, which is

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{H}, \xi \in \mathbb{R}^m, \rho \in \mathbb{R}} &= \tau(\mathbf{w}, \xi, \rho) = \frac{1}{2} \|\mathbf{w}^2\| - \nu\rho + \frac{1}{m} \sum_{i=1}^m \xi_i \\ \text{subject to } &y_i k(x_i, x') \geq \rho - \xi_i, \quad i = 1, \dots, m \\ &\text{and } \xi_i \geq 0, \rho \geq 0 \end{aligned} \quad (11)$$

In the above equation \mathbf{w} is weight vector in a Hilbert space \mathcal{H} , slack variables ξ_i are used to relax constraints subject functions, $\nu \in (0, 1)$ and ρ are parameterization variables. In our case, dimensionality of the problem is $m = 3$. Also, $k(x_i, x')$ is the kernel, which may be based on Gaussian, polynomial, and sigmoid function. In case of polynomial function $k(x_i, x') = \gamma[(x_i \cdot x') + b]^d$, where x_i is a vector from the training set, x' is the vector to be classified, b is a constant and d is the power of the polynomial.

We performed a grid search using the following values for $\gamma = \{0.001, 0.005, 0.01, 0.015, 0.02, 0.01\}$; $d = \{1, 2, 3, 4\}$; $\nu = \{0.4, 0.5, 0.6\}$ and applying the following kernels - linear, polynomial, Gaussian based, radial basis, and sigmoid function. We found that the maximum f-measure accuracy of 0.89 was obtained with the polynomial kernel [14] of $(1/70)(x_i \cdot x') + 6)^4$ with $\nu = 0.5$ and $\gamma = 0.015$.

Recall that selecting the right parameters and kernel, for the SVM, is a challenging task in the high dimensional spaces especially because of visualization difficulties. Thus for the 4D manifold, the observations in section 3.2 suggest us that most likely the kernel which provides higher accuracy would be the fourth degree polynomial kernel. This conclusion was validated in [11] where a grid search over multiple kernels and parameters using 51 LFVs determined that the best average accuracy in the 4D Manifold is obtained again by using a polynomial kernel of degree four. A similar kernel is also reported in [5] as an optimum one for classification of 64 skin lesions.

As stated above the experiments performed in the 3D space with the set of 70 LFVs applying SVM with polynomial, Gaussian based, radial basis function, and sigmoid kernels provided that the best f-measure classification was obtained with a polynomial of degree four. This result complies with the suggestion derived in Section 3.2, that the best f-measure accuracy is obtained with a fourth degree polynomial kernel.

Therefore the LFVs geometric patterns observed in the 3D space may be used to make prediction about the kernel and its parameters to be used by the SVM in 3D and 4D manifolds. Note that there is no reliable method, in the literature, capable of finding these SVM parameters. A combinatorial reasoning tells that the number of test runs will dramatically increase if we increase the size of each used set with a few more elements.

Note that similar results can be obtained with other soft margin SVM, however ν -SVC is preferred in our study since in practice, choosing a ν is easier than parameter C of C-SVM. The results show that ν -SVM applied in the 3D space provides a better f-measure accuracy for classification than the TDS applied in the 4D space.

5 Conclusion

The contribution of this study can be summarized as follows. We used 4D LFVs generated in [4,5,11] on the base of a modified version of the ABCD rule and the idea of measuring TDS to build a 4D Riemannian manifold as a model to study skin lesion identification. In an attempt to characterize the geometric structure and explore the distribution of the malignant lesions, we found methods to project 4D manifold onto 2D and 3D spaces which provide visualizations of the 4D LFVs in 2D polar and 3D cylindrical and spherical spaces.

Recall that the classification by SVM depends on the kernel and its parameters. Also, there are no reliable approaches in the literature to select the right kernel for a particular classification problem. Thus, the projection to 3D spherical space was used to provide an observation of the LFVs distribution, which allowed us to select the kernel and its parameters, which provide the most accurate classification for malignant and benign points. As a result of the presentation of the LFV on a unit sphere we observed that the malignant vectors are most probably located on a fourth degree polynomial curve which lies on the unit sphere. This observation suggests that polynomial kernel of degree four may be used to build up a malignancy separating surface applying ν -SVM in the 3D spherical space.

Thus, we are able to suggest a kernel for SVM learning. Therefore, an advantage of this approach is that helps the user avoid tedious and time consuming grid search on SVM kernels and parameters. As shown in the experimental section, results of the search using different kernels complies with our observation in [3,2].

Speaking of the opportunity of kernel and parameters prediction for the SVM most accurate f-measure classification we have to note the followings:

- The observation is performed in a 3D space. Therefore it would be accurate for prediction made in this dimension which is relatively low.
- Bringing the conclusion to a higher dimension may not be that much accurate. This statement holds because adding a single dimension to the 3D may lead to the mapping of a single point in 3D to multiple points in 4D. But this property still allows us to reject the use of some kind of kernels, which is sufficient to reduce the calculation complexity two or three times.

For example, observing a polynomial of degree greater than one in 3D rejects the opportunity for use a linear, or sigmoid or Gaussian kernel in the 4D manifold.

Another advantage of the present study is a better f-measure of malignancy identification than TDS rule used by dermatologists. Also, although we are focusing on a 4D manifold, for the model in this paper, our approach can be extended

to higher dimensions as we can add more features to the LFV. Moreover our method is automatic whereas the one used by clinicians is manual and carried out through naked eyes.

A drawback of this method is the subjective observation of a relatively small amount of LFVs selected from a single collection. Also, there is a number of parameters that should be correctly selected in order to provide accurate LFVs. Hence, this study could be extended with a new image collection to increase sample space and a work on automated parameters selection.

This work will continue with additional manifolds generalizations to higher dimension and development of a new metrics to provide more accurate separation of LFVs. Level set method [13] will be applied to extract additional lesion features such as dots and streaks.

Acknowledgment. Authors thank anonymous reviewers for their suggestions and notes, which helped us to improve quality of this paper. This study is partly supported by Texas A&M University-Commerce Research Grant.

References

1. American Cancer Society. Cancer Facts & Figures (2010), <http://www.cancer.org/> (accessed July 26, 2010)
2. Argenziano, G., Fabbrocini, G., Carli, P., De Giorgi, V., Sammarco, E., Delfino, M.: Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions. Comparison of ABCD rule of dermatoscopy and a new 7-point checklist based on pattern analysis. *Arch. Dermatol.* 134, 1563–1570 (1998)
3. Argenziano, G., Soyer, H.P., De Giorgi, V., et al.: Interactive atlas of dermoscopy. EDRA Medical Pub., Milan (2000)
4. Mete, M., Sirakov, N.M.: Application of active contour and density based models for lesion detection in dermoscopy images. *BMC Bioinformatics* 11(suppl. 6), S23 (2010), doi:10.1186/1471-2105-11-S6-S23
5. Mete, M., Sirakov, N.M.: Dermoscopic diagnosis of melanoma in a 4D feature space constructed by active contour extracted features. *Journal of Medical Imaging and Graphics* 36, 572–579 (2012)
6. Mulchrone, K., Choudhury, K.: Fitting an ellipse to an arbitrary shape: implications for strain analysis. *J. of Structural Geology* 26(1), 143–153 (2004)
7. O’Neill, B.: Semi-Riemannian geometry with applications to relativity. Academic Press, New York (1983)
8. Nachbar, F., Stolz, W., Merkle, T., et al.: The ABCD rule of dermatoscopy: High prospective value in the diagnosis of doubtful melanocytic skin lesions. *J. of the American Academy of Dermatology* 30(4), 551–559 (1994)
9. Petersen, P.: Riemannian geometry. Graduate Texts in Mathematics, vol. 171. Springer, New York (1998)
10. Scholkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New Support Vector Algorithms. *Neural Comput.* 12, 1207–1245 (2000)
11. Sirakov, N.M., Mete, M., Nara, S.C.: Automatic boundary detection and symmetry calculation in dermoscopy images of skin lesions. In: *IEEE ICIP 2011, Brussels*, pp. 1637–1640 (2011)

12. Sirakov, N.M., Ushkala, K.: An Integral Active Contour Model for Convex Hull and Boundary Extraction. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnaç o, M.L., Silva, C.T., Coming, D. (eds.) ISVC 2009, Part II. LNCS, vol. 5876, pp. 1031–1040. Springer, Heidelberg (2009)
13. Thieu, Q.T., Luong, M., Rocchisani, J.-M., Viennet, E.: A Convex Active Contour Region-Based Model for Image Segmentation. In: Real, P., Diaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W. (eds.) CAIP 2011, Part I. LNCS, vol. 6854, pp. 135–143. Springer, Heidelberg (2011)
14. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)

Segmentation by a Local and Global Fuzzy Gaussian Distribution Energy Minimization of an Active Contour Model

Quang Tung Thieu¹, Marie Luong¹, Jean-Marie Rocchisani²,
Nikolay Metodiev Sirakov³, and Emmanuel Viennet¹

¹ L2TI Laboratory, Paris 13 University, 93430 Villetaneuse, France
{quangtung.thieu,marie.luong,emmanuel.viennet}@univ-paris13.fr

² Avicenne University Hospital, 93000 Bobigny, France
and LAGA Laboratory, Paris 13 University, 93430 Villetaneuse, France
rocchisani@univ-paris13.fr

³ Dept. of Computer Science and Info Systems, Dept. of Mathematics,
Texas A&M University Commerce, Commerce, TX 75429, USA
Nikolay.Sirakov@tamuc.edu

Abstract. We propose a novel region-based active contour model, which incorporates the image local and global information into a fuzzy energy function, providing a robust and accurate segmentation while accounting for intensity inhomogeneity. In this model, image intensities are assumed to have a Gaussian distribution with different means and variances. While the local information contributes to dealing with intensity inhomogeneity, the global information allows improving the performance of the model in the case of very noisy or blurred images. Another interesting property is that the energy function of the proposed model is convex with respect to the variable used to determine the contour. This makes the accuracy of the result invariant with respect to the position of the initial contour and more suitable for an automatic segmentation. Moreover, the energy function of the proposed model is minimized in a computationally efficient way by calculating the fuzzy energy alterations directly. Experiments are carried out to validate the capabilities of the proposed model. Comparisons are provided with ground truth and other methods in the field to underline the superiority of our method in terms of accuracy.

Keywords: Segmentation, Active Contour, Fuzzy energy function, Medical images.

1 Introduction

Segmentation is a basic domain in image analysis and processing. Generally, it consists of partitioning a given image into objects sharing the same image properties and background. In medical imaging, segmentation is necessary for detection of pathological regions such as tumors or lesions [6,18,20]. With the newest imaging technologies and computer vision, segmentation can help improving the

diagnosing, monitoring and staging of patients. Despite the recent acquisition technologies and performances of reconstruction algorithms, the quality of medical images can be affected by inherent noise or artifacts involving PET (Positron Emission Tomography), MR (Magnetic Resonance) images as well as low dose CT (Computed Tomography) images [17], making it difficult to distinguish the organ structure. This in turn makes the segmentation more difficult [13]. An important obstacle for the effective segmentation is the intensity inhomogeneity, which is due to a typical phenomenon known as shading artifact in medical images. It behaves as a relative variation of the intensity in the object of interest.

A motivation for this study is the development of an automatic tool capable of facilitating the work of the medical experts, avoiding manual delineation for purpose of segmentation. Note that such manual work is tedious, prone to errors, and depends on the qualification of the expert.

A number of segmentation methods have been proposed in the literature. Among them the active contour models (ACM) have gained significant interest due to their accuracy and ability to handle complex images. Since the introduction of the first model by Kass *et al.* [5], ACM have been proven to be among the most successful and accurate methods for segmentation of medical images [1, 4, 18]. The main idea behind the ACM is based on deformation of an initial curve so that it evolves towards object boundaries, under some constraints. The existing ACM can be categorized as: edge-based [2, 5, 11], and region-based [3, 8, 15, 16, 19].

The methods in the first category, where image gradient is used to stop the curve evolution, may fail when the gradient of the object boundaries is not well defined, e.g., noisy, blurred or even discontinuous edges. To overcome this drawback, other image information is used including the properties of regions between the contour, in order to efficiently evolve and stop the evolving curve. This characterizes the region-based ACM, including the popular CV model proposed by Chan and Vese [3], by assuming that image regions are homogeneous. The CV model is efficient in segmenting images with noisy and weak boundaries, but inefficient while dealing with intensity inhomogeneity, due to the above assumption. Other methods consist of combining ACM with fuzzy logic to enhance the ability of segmentation [4, 7], allowing more robustness to weak boundaries and initial condition. In [7], Krinidis *et al.* proposed an effective ACM based on fuzzy energy to segment objects whose boundaries are very smooth and discontinuous. However, these models fail to segment images with intensity inhomogeneity by assuming that each region is homogeneous, as the CV model does.

To tackle the problem of intensity inhomogeneity, the local intensity information is taken into account in the energy function [8, 14], where a local energy term using a weighting function and local intensity means are applied. In [8], the weighting function is a Gaussian kernel used to account for localization property, while local intensity means are approximated by smooth functions. By using the local information, these models are proven to be efficient to handle the intensity inhomogeneity. However, the localization and the non-convexity of the energy function of these models make them dependent on the initial contour. To improve

the robustness with respect to the position of the initial contour, some other approaches, namely the LGIF [15] and the LCV [16] models, take into account the global information in the energy function. These models are formulated using level set function [9] which is an implicit method and allows automatic change of topology. However, the inconvenience induced from this formulation is the non-convexity of the energy function. Thus, the influence of the initial contour is only reduced by global information, but still remains unavoidable.

In order to cope with the intensity inhomogeneity and the dependence on the initial condition of the ACM, in this paper we propose a novel region-based ACM which incorporates both local and global information in a fuzzy energy function. The model takes advantage of both approaches as well as of the fuzziness of the energy. In the present method, intensity information is described by Gaussian distributions with different means and variances. The local information approach is inspired by the LGDF model of Wang *et al.* [14] which utilizes the local information distribution. This allows our model to handle the intensity inhomogeneity which is not the case of the FEBAC [7] model. In addition, using a fuzzy membership function to determine the evolution process of the contour, our energy function is convex, while the LGDF model is not. The convexity of our model implies that its accuracy is independent from the initial contour. Furthermore, the energy function of the proposed model is minimized in a computationally efficient way by utilizing the fast algorithm described in [7,12] to calculate the fuzzy energy alterations directly. This calculation allows us to avoid issues related to numerical stability constraints. Hereafter, we will refer to our model as LGFGD (local and global fuzzy Gaussian distribution).

The rest of this paper is organized as follows. In Section 2, the description of our model and its fuzzy energy are introduced along with a proof of the convexity of the model, followed by the numerical implementation, an illustration, and the computational complexity of our method. Experimental results and comparison with existing methods are given in Section 3. Finally, Section 4 presents the conclusions and the directions of the future work.

2 The Proposed Method

In this section, we first present the LGFGD energy function. Then, this energy function is described in a similar way as in the level-set formulation, which is referred also as pseudo-level formulation. Then, using appropriate values of the weighting exponent on the degree of fuzzy membership, we will prove that the energy function of our model is convex. Further, the local information is taken into account to tackle the intensity inhomogeneity, while the global information is used to handle the objects whose boundaries are not well defined by the gradient. To exploit the local image intensities, we use Gaussian distributions with different means and variances at each pixel in a region and the Gaussian kernel to define the local property of the energy function. We assume that the global image intensities are characterized by Gaussian distributions with the same means and variances for all the pixels in a region. Moreover, we combine the local and

global energy functions with the fuzzy logic to benefit from this technique in clustering, which provides more accurate detection. This allows us to obtain a convex model capable of handling efficiently the intensity inhomogeneity.

Let $\Omega \subset \mathbb{R}^2$ be an image domain, and $I : \Omega \rightarrow \mathbb{R}^+$ denotes the image. The proposed ACM is based on the minimization of the following fuzzy energy function:

$$F(M) = \mu|C| + \lambda F_1(M_1) + (1 - \lambda)F_2(M_2) \tag{1}$$

where $M = (C, c_1, c_2, \epsilon_1^2, \epsilon_2^2, f_1, f_2, \sigma_1^2, \sigma_2^2, u)$, $M_1 = (c_1, c_2, \epsilon_1^2, \epsilon_2^2, u)$, and $M_2 = (f_1, f_2, \sigma_1^2, \sigma_2^2, u)$, $\mu \geq 0$ is a constant to control the length $|C|$ of the contour C , $0 < \lambda < 1$ is a constant to control the influence of the global term F_1 and the local term F_2 , c_1 and c_2 are the global intensities described by the standard deviations of Gaussian distributions ϵ_1 and ϵ_2 , respectively inside and outside the contour C , f_1 and f_2 are the local intensities described by the standard deviations of Gaussian distributions σ_1 and σ_2 , respectively inside and outside of C . The global and local terms are defined as follows:

$$F_1(M_1) = - \int_{\Omega} \log p(I(y), \epsilon_1)[u(y)]^m dy - \int_{\Omega} \log p(I(y), \epsilon_2)[1 - u(y)]^m dy \tag{2}$$

$$F_2(M_2) = - \int_{\Omega} \left[\int_{\Omega} K_{\sigma}(x - y) \log p_x(I(y), \sigma_1)[u(y)]^m dx \right] dy - \int_{\Omega} \left[\int_{\Omega} K_{\sigma}(x - y) \log p_x(I(y), \sigma_2)[1 - u(y)]^m dx \right] dy \tag{3}$$

where x, y are pixels, K_{σ} is a Gaussian kernel with a standard deviation σ , $u(x) \in [0, 1]$ and $1 - u(x) \in [0, 1]$ are the degrees of memberships of $I(x)$ inside and outside of C respectively, $m \geq 2$ is a weighting power on the degree of membership, $p(I(y), \epsilon_i)$ and $p_x(I(y), \sigma_i)$, $i = 1, 2$, are the Gaussian distributions:

$$p(I(y), \epsilon_i) = \frac{1}{\sqrt{2\pi}\epsilon_i} \exp \left(- \frac{(c_i - I(y))^2}{2\epsilon_i^2} \right) \tag{4}$$

$$p_x(I(y), \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left(- \frac{(f_i(x) - I(y))^2}{2\sigma_i^2} \right) \tag{5}$$

2.1 Pseudo Level Set Formulation

In this section, we formulate the fuzzy energy function by using a pseudo level set formulation based on the membership values of u as defined in [7]. The curve C in Ω is represented by the pseudo zero level set of u such that $C = \{x \in \Omega : u(x) = 0.5\}$, $in(C) = \{x \in \Omega : u(x) > 0.5\}$, $out(C) = \{x \in \Omega : u(x) < 0.5\}$.

Now, we change the contour length $|C|$ in (1) by $\int_{\Omega} |\nabla u| dx$ for the following goal: When our energy function is minimized, the values of u for the pixels inside the contour C are different from the values of u for the pixels located outside the contour. However, the values of u for the pixels located inside the contour C are similar. This is the same for the pixels located outside the contour. Then, energy function (1) is transformed to:

$$F(M) = \mu \int_{\Omega} |\nabla u| dx + \lambda F_1(M_1) + (1 - \lambda)F_2(M_2) \tag{6}$$

2.2 Convexity of Our Model with Respect to u

First, we recall the definition of a convex function (Definition 1). Then, we will prove that our energy function is convex with respect to u (Theorem [11](#)).

Definition 1. A real valued function $k : D \rightarrow \mathbb{R}$ defined on a convex set D is called convex if for $\forall x, y \in D$ and $\forall t \in [0, 1]$, the following inequality holds:

$$k(tx + (1 - t)y) \leq tk(x) + (1 - t)k(y) \tag{7}$$

Note that the variable u of our energy function is in $BV_\Omega(0, 1)$, which is the set of bounded functions in $[0, 1]$. So, we will prove that $BV_\Omega(0, 1)$ is convex. It means that for $\forall u, v \in BV_\Omega(0, 1)$ and $\forall t \in [0, 1]$, we have to prove $tu + (1 - t)v \in BV_\Omega(0, 1)$. Indeed, consider $f(x, t) = tu(x) + (1 - t)v(x) = (u(x) - v(x))t + v(x)$, this is a linear function with respect to $t \in [0, 1]$. It follows that the max and the min of $f(x, t)$ are at the end points 0 and 1: $f(x, 0) = v(x)$ and $f(x, 1) = u(x)$. Since $u(x), v(x) \in [0, 1]$, we have $f(x, t) \in [0, 1]$, $\forall x \in \Omega$ and $\forall t \in [0, 1]$. Thus, $f \in BV_\Omega(0, 1)$ and also $tu + (1 - t)v \in BV_\Omega(0, 1)$. So, $BV_\Omega(0, 1)$ is a convex set.

Then, we formulate the following theorem.

Theorem 1. The energy function F in Eq. [11](#) is convex with respect to u .

Proof. Utilizing $|a + b| \leq |a| + |b|$, $\forall a, b$, and the linearity of the gradient operator, we apply the definition of convex function to prove that $\int_\Omega |\nabla u| dx$ is convex with respect to u .

To prove the convexity of F_1 with respect to u , we consider the function:

$$G(z) = az^m + b(1 - z)^m \tag{8}$$

where $z \in [0, 1]$, $m \geq 2$, $a, b > 0$. We will show that $G(z)$ is convex with respect to t by proving that $G''(z) \geq 0$. Indeed, we have:

$$G''(z) = m(m - 1)az^{m-2} + m(m - 1)b(1 - z)^{m-2} \tag{9}$$

Recall that $z \in [0, 1]$, $m \geq 2$, $a, b > 0$. Therefore, we have $G''(z) \geq 0$, which implies that $G(z)$ is convex with respect to z . Since p is a Gaussian distribution, its values are in $(0, 1)$. Thus, $\log p < 0$. Then, we can consider $a = -\log p(I(y), \epsilon_1)$, $b = -\log p(I(y), \epsilon_2)$, and $z = u(y)$, where $y \in \Omega$. Then, the function:

$$F_1^y(u) = -\log p_1(I(y), \epsilon_1)[u(y)]^m - \log p_2(I(y), \epsilon_2)[1 - u(y)]^m \tag{10}$$

is a convex function with respect to u . It means that if $\forall y \in \Omega$, $\forall t \in [0, 1]$, $\forall u, v \in BV_\Omega(0, 1)$, we have the following inequality:

$$F_1^y(tu + (1 - t)v) \leq tF_1^y(u) + (1 - t)F_1^y(v) \tag{11}$$

Then, by integrating inequality [11](#) on Ω , we obtain:

$$\int_\Omega F_1^y(tu + (1 - t)v) dy \leq t \int_\Omega F_1^y(u) dy + (1 - t) \int_\Omega F_1^y(v) dy \tag{12}$$

Note that, $F_1 = \int_{\Omega} F_1^y dy$. So, we have:

$$F_1(tu + (1 - t)v) \leq tF_1(u) + (1 - t)F_1(v) \tag{13}$$

$\forall t \in [0, 1], \forall u, v \in BV_{\Omega}(0, 1)$. Therefore, F_1 is convex with respect to u .

Taking into account that $K_{\sigma} > 0$, we prove in a similar fashion that F_2 is convex with respect to u as well. Finally, considering that $\mu, \lambda, 1 - \lambda$ are positive numbers, we prove that the energy function F is convex with respect to u . \square

2.3 Numerical Implementation

The energy function could be solved by the gradient descent method derived from the Euler-Lagrange equation which converges relatively slowly. For our model, we apply the fast numerical scheme proposed by Song *et al.* [12] and developed by Krinidis *et al.* [7]. Instead of solving the Euler-Lagrange equation, this method calculates the energy directly and verifies if the energy decreases when the intensity membership on the image changes.

Before presenting the algorithm to solve our model, we compute each variable fixing the others. For simplicity, without losing the generality, Eq. (6) has been considered without the length term ($\mu = 0$).

Applying calculus of variation we can see that if we fix u , the minimization of the energy in Eq. (6) with respect to $c_1, c_2, \epsilon_1^2, \epsilon_2^2, f_1, f_2, \sigma_1^2, \sigma_2^2$ implies:

$$c_i = \frac{\int_{\Omega} I(y)[a_i(y)]^m dy}{\int_{\Omega} [a_i(y)]^m dy}, \quad \epsilon_i^2 = \frac{\int_{\Omega} (c_i - I(y))^2 [a_i(y)]^m dy}{\int_{\Omega} [a_i(y)]^m dy} \tag{14}$$

$$f_i(x) = \frac{\int_{\Omega} K_{\sigma}(x - y)I(y)[a_i(y)]^m dy}{\int_{\Omega} K_{\sigma}(x - y)[a_i(y)]^m dy} \tag{15}$$

$$\sigma_i^2(x) = \frac{\int_{\Omega} K_{\sigma}(x - y)(f_i(x) - I(y))^2 [a_i(y)]^m dy}{\int_{\Omega} K_{\sigma}(x - y)[a_i(y)]^m dy} \tag{16}$$

where $i = 1, 2, a_1(\cdot) = u(\cdot), a_2(\cdot) = 1 - u(\cdot)$.

Moreover, if we fix $c_1, c_2, \epsilon_1^2, \epsilon_2^2, f_1, f_2, \sigma_1^2, \sigma_2^2$, the minimization of the energy F with respect to u , by calculus of variation, allows us to compute u as follows:

$$u(y) = \frac{1}{1 + \left[\frac{\lambda \log p(I(y), \epsilon_1) + (1 - \lambda) \int_{\Omega} K_{\sigma}(x - y) \log p_x(I(y), \sigma_1) dx}{\lambda \log p(I(y), \epsilon_2) + (1 - \lambda) \int_{\Omega} K_{\sigma}(x - y) \log p_x(I(y), \sigma_2) dx} \right]^{\frac{1}{m-1}}} \tag{17}$$

Now, given a point y_0 in Ω , the intensity value of point y_0 is $I(y_0)$, and the corresponding degree of membership for $I(y_0)$ is u_{y_0} . Assume that the degree of membership of $I(y_0)$ changes to the new value u_{ny_0} . Denote by ΔF the difference between the new and old energies calculated at the new and old degree of membership of $I(y_0)$. Assuming that the changes of $\epsilon_1, \epsilon_2, \sigma_1$ and σ_2 at a point are very small, ΔF is calculated as follows:

$$\Delta F = \mu \Delta l + \lambda \Delta F_1 + (1 - \lambda) \Delta F_2 \tag{18}$$

where Δl is the change of the contour length. The rate of change of the global fuzzy energy ΔF_1 and the local fuzzy energy ΔF_2 are calculated as follows:

$$\begin{aligned} \Delta F_1 = & -\log \frac{1}{\sqrt{2\pi}\epsilon_1} \Delta u_m + \frac{s_1 \Delta u_m}{s_1 + \Delta u_m} \frac{(I(y_0) - c_1)^2}{2\epsilon_1^2} \\ & -\log \frac{1}{\sqrt{2\pi}\epsilon_2} \Delta v_m + \frac{s_2 \Delta v_m}{s_2 + \Delta v_m} \frac{(I(y_0) - c_2)^2}{2\epsilon_2^2} \end{aligned} \quad (19)$$

$$\begin{aligned} \Delta F_2 = & -K_\sigma * \left(\log \frac{1}{\sqrt{2\pi}\sigma_1} \right) (y_0) \Delta u_m - K_\sigma * \left(\log \frac{1}{\sqrt{2\pi}\sigma_2} \right) (y_0) \Delta v_m \\ & + \sum_{x \in \Omega} \frac{s_3(x) K_\sigma(x - y_0) \Delta u_m}{s_3(x) + K_\sigma(x - y_0) \Delta u_m} \frac{(I(y_0) - f_1(x))^2}{2\sigma_1^2(x)} \\ & + \sum_{x \in \Omega} \frac{s_4(x) K_\sigma(x - y_0) \Delta v_m}{s_4(x) + K_\sigma(x - y_0) \Delta v_m} \frac{(I(y_0) - f_2(x))^2}{2\sigma_2^2(x)} \end{aligned} \quad (20)$$

where $s_1 = \sum_{y \in \Omega} [u(y)]^m$, $s_2 = \sum_{y \in \Omega} [1 - u(y)]^m$, $s_3 = K_\sigma * [u]^m$, $s_4 = K_\sigma * [1 - u]^m$, $\Delta u_m = [u_{ny_0}]^m - [u_{y_0}]^m$, $\Delta v_m = [1 - u_{ny_0}]^m - [1 - u_{y_0}]^m$.

In summary, the algorithm for the active contour evolved by the fuzzy energy function consists of the following steps:

- **Step 1.** Initial partition: setup $u > 0.5$ for the interior of the contour, and $u < 0.5$ for the exterior.
- **Step 2.** For each iteration, the following sub-steps are performed:
 - (i) Compute $c_1, c_2, \epsilon_1, \epsilon_2, f_1, f_2, \sigma_1, \sigma_2$ by fixing u and using (14-16).
 - (ii) For each pixel $y_0 \in \Omega$, we are given u_{y_0} as the degree of membership of $I(y_0)$ in the previous iteration. The new degree of membership u_{ny_0} is calculated by using (17) where the fixed $c_1, c_2, \epsilon_1, \epsilon_2, f_1, f_2, \sigma_1, \sigma_2$ are calculated in the previous step. Then, we use (18-20) to calculate the difference between the new and old energy ΔF at the pixel y_0 . If $\Delta F < 0$, then we change u_{y_0} with u_{ny_0} ; otherwise, we keep the old value u_{y_0} .
 - (iii) Repeat step (ii) for all the pixels $y_0 \in \Omega$ to obtain all new degree of membership u_{ny_0} . Then, we compute the total energy F of the image.
- Repeat step 2 until the rate of change of F becomes zeros.
- **Step 3.** We use the following criterion: if $u(x) > 0.5$, then x is in the object. Otherwise, x is in the background.

2.4 Illustration of the Method

In this section, we illustrate our method on a real image of rices. We present the evolution of the contour (Figs. 1(a) - 1(d)) where the initial contour is in red (dark gray) and the evolution contours are in yellow (light gray); the evolution of the energy function F (Fig. 1(e)). Fig. 1(e) shows that our energy function converges to the minimum at the 12^{nd} iteration. However, we see that the contours at the 5^{th} (Fig. 1(c)) and 25^{th} (Fig. 1(d)) iterations are visually similar. This can be explained by using the values of u at two points p_1 and p_2 reported

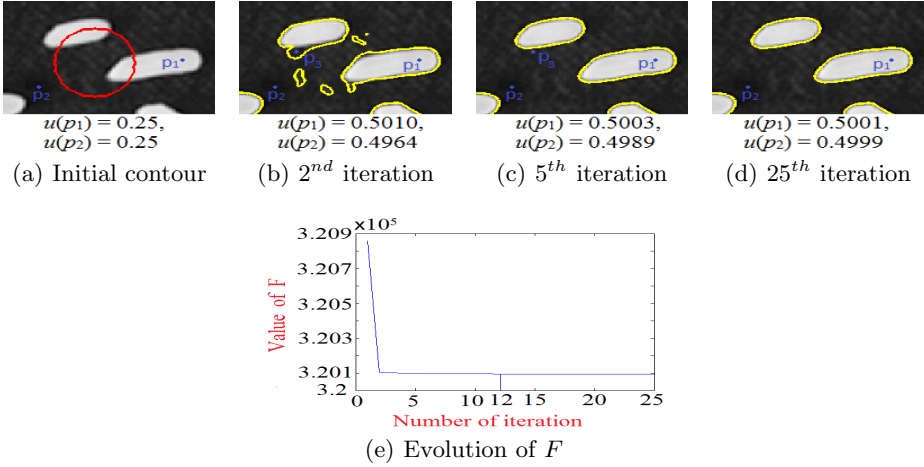


Fig. 1. Illustration for our method on a real image of rices

Table 1. Values of $f_1, f_2, \sigma_1, \sigma_2$ and u for the pixel p_3 reported in Fig. [1\(b\)](#)[1\(c\)](#)

	f_1	f_2	σ_1	σ_2	u
2^{nd} iteration	63.5085	44.5250	2537.4	922.3	0.5003
5^{th} iteration	51.1039	51.2402	1564.9	1579.1	0.4986

in each figure as follows: at the 5^{th} iteration, $u(p_1) = 0.5003, u(p_2) = 0.4989$. As $u(p_1) > 0.5$, then p_1 is a pixel corresponding to the object. As $u(p_2) < 0.5$, then p_2 is a pixel from the background. However, the value of our energy function is not minimized at this iteration. Therefore, the values of u must be changed. But, this change does not affect the fact that p_1 and p_2 are the pixels of the object and background, respectively. Moreover, Table [1](#) shows the evolution of the pixel p_3 , reported in Figs. [1\(b\)](#)[1\(c\)](#), through the values of $f_1, f_2, \sigma_1, \sigma_2$, and u which show that p_3 is the pixel in the background. At the 2^{nd} iteration, as $u(p_3) = 0.5003 > 0.5$, p_3 is a pixel in the object. We see also at this iteration: $f_1 > f_2, \sigma_1 < \sigma_2$. However, at the 5^{th} iteration, as $u(p_3) = 0.4986 < 0.5$, p_3 is a pixel in the background, we have also the changes of f_1, f_2, σ_1 and σ_2 : $f_1 < f_2$ and $\sigma_1 > \sigma_2$.

2.5 Computational Complexity

The computational complexity of the LGFGD method is in the order of $O(N_{xyG})$, where $N_{xyG} = N_x \cdot N_y \cdot N_{G_x} \cdot N_{G_y}$, N_x and N_y are the column and row numbers of the considered image, respectively; N_{G_x} and N_{G_y} are the column and row numbers of the Gaussian kernel K_σ , respectively. To validate the above complexity, we test the LGFGD method on an image with size 65×68 (Fig. [2\(a\)](#)) and its

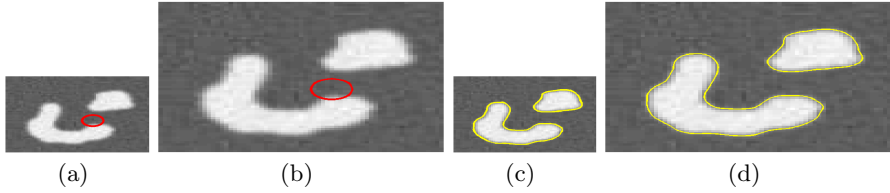


Fig. 2. (a) and (b) Images with sizes 65×68 and 130×136 , respectively. (c) and (d) Segmentation results applying the LGFGD on (a) and (b), respectively.

zoom-in image with double size for each dimension 130×136 (Fig. 2(b)). As it can be observed in Figs. 2(a) - 2(b), the initial contours are at the same positions and the obtained results are also the same. The CPU times to obtain results in Figs. 2(c) - 2(d) are 1.44 second and 5.52 second, respectively. We can see that, since the size of Fig. 2(b) is four times the size of Fig. 2(a), the CPU time to obtain Fig. 2(d) is approximately four times the CPU time to obtain Fig. 2(c).

3 Experimental Results and Comparison

We present results of our model on variety of synthetic and real images, namely PET, CT, X-Ray and MR images. A comparative evaluation has been performed to demonstrate the advantages of our method over similar contemporary methods such as the LGIF [15], the LCV [16] and the FEBAC [7]. The code of the LGIF model is obtained by incorporating the global term in the code of the LBF [8] model, which is obtained from the home page of the author [8]. The code of the LCV model is provided by the author of [16]. The code of the FEBAC model is implemented based on the algorithm published in [7]. To provide a fair comparison, all the methods used one and the same initial contour, which is painted in red (dark gray), while the yellow (light gray) curves indicate the final contours. We use the MATLAB r2008b to implement our algorithm on a computer with Intel Core 2 Duo CPU 2.93GHz and 4GB RAM. The images used for the experiments vary in size between 65×68 and 452×348 .

In Fig. 3, we compare our model with the FEBAC model on two synthetic images. A synthetic image (top row) including two homogeneous objects with similar intensities is tested. One may observe that the results of FEBAC and our models are accurate and similar. However, for the synthetic image exhibiting three homogeneous objects with different intensities (bottom row), the FEBAC model detected the two objects with dark intensities on a light background, while the object with a quite different intensity, from those objects, is missed. Note that the missed object and the background have quite similar intensities. Thus, the global model FEBAC fails when there is not significant variation of intensities between the object and the background. On the contrary, our new model LGFGD successfully extracted the three objects, as show in Fig. 3, bottom row.

Fig. 4 shows the results of the FEBAC and our models for the case of noisy images with Gaussian distribution. The Gaussian noise is added with standard

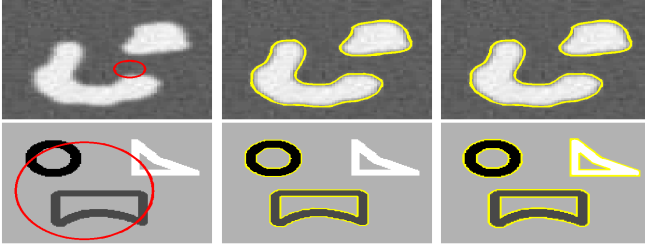


Fig. 3. Test on two synthetic images. Col. 1: Original images with initial contours. Col. 2: Results of the FEBAC model. Col. 3: Results of the LGFGD.

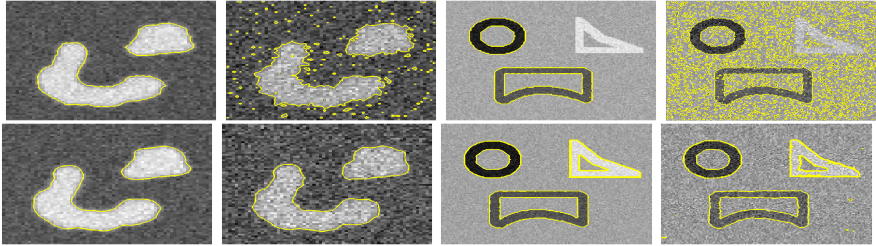


Fig. 4. Results on noisy versions of the original images from Fig. 3 with standard deviations 10 (col. 1&3) and 30 (col. 2&4). Top: results of the FEBAC model. Bottom: results of our model.

deviations 10 and 30 in the images in Fig. 3. One may observe that the results of the FEBAC model, in the case of standard deviation 10, are similar to the results obtained from the original image in Fig. 3 while the result of this model are not accurate for higher noise level. We can see that on images with high noise (standard deviation 30), our results are still as accurate as the results for the original images in Fig. 3.

The results in Figs. 3-4 show that the LGFGD model outperforms the FEBAC model in what concern the segmentation of low contrast and noisy images.

Fig. 5 compares the segmentation capabilities of FEBAC, LGIF, LCV and the newly proposed LGFGD method. Three X-ray images of blood vessels are used for the purpose of comparison. All images are nonhomogeneous. In this experiment, the results provided from the LGIF (4th column) and our (5th column) models are similar and good. The FEBAC does not give accurate results (2nd column) for all of the three images because of its global property. For the LCV model (3rd column), the result of the top image is not accurate, while the result of the middle image is similar to the results of the LGIF and our models. The bottom image is sufficiently well segmented except the right corner part where the contour is missed. The reason why the LGIF, the LCV and our models provide good results on these nonhomogeneous images is that these models utilize the local information in the energy function.

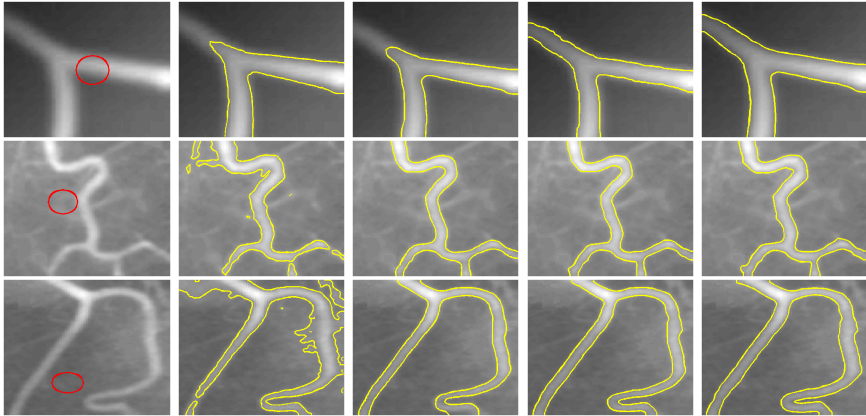


Fig. 5. Results obtained on three blood vessel X-ray images. From left to right columns: original images with initial contours, results of the FEBAC, the LCV, the LGIF and the LGFGD models, respectively.

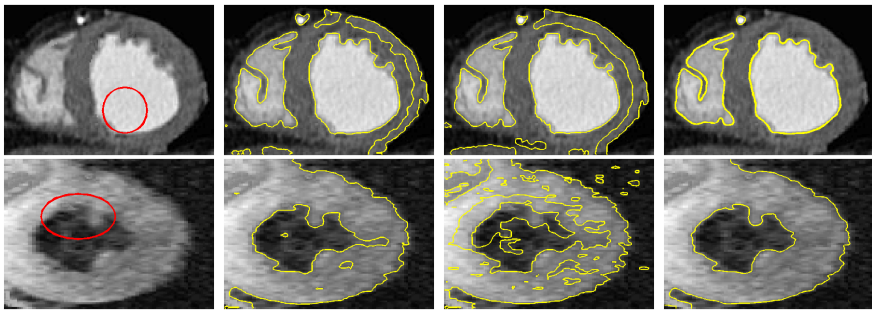


Fig. 6. Results obtained on two heart MR images. Column 1: original images and initial contours. Column 2: results of the LCV model. Column 3: Results of the LGIF model. Column 4: Results of our model, LGFGD.

Further, we compare our model with the LCV and the LGIF models using two heart MR images. The results are illustrated in Fig. 6 where one may observe that the new model LGFGD detects the object boundary in a more accurate fashion than the LCV and the LGIF did. On the top images, the results of the LCV and the LGIF models are good enough, but there is seen also some non-accurate contours. On the bottom image with nonhomogeneous intensity, the result of LGIF model is completely incorrect, while the LCV contour can not be stopped at the exact contour. This is not the case for our result.

Now, we present in Fig. 7 experiments on a noisy 2D CT images, which contains the thorax at the level of the pulmonary arteries (at top left), of 3mm thickness. This image is acquired with standard reconstruction filter and is noise-free (or low noise). Another CT image (at top right) is 2mm thick, acquired at the same position with optimized reconstruction filter, and different values of

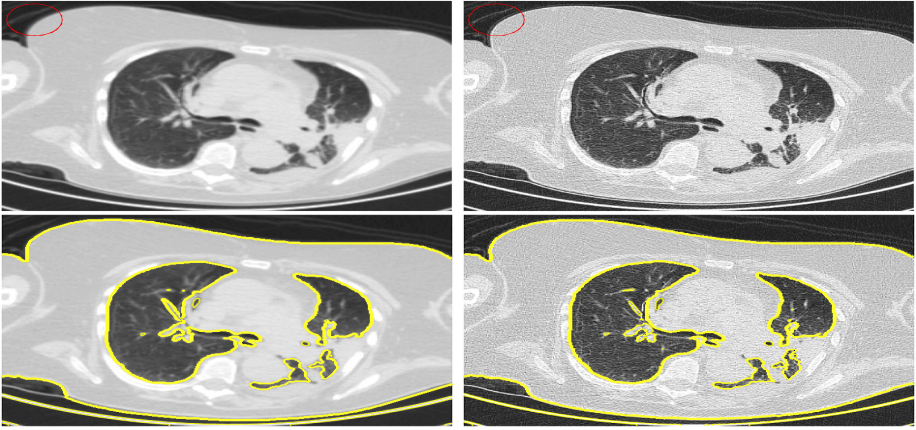


Fig. 7. Results on a thorax CT image and its noisy version. Top row: original images with initial contour. Bottom row: the corresponding segmentation results.

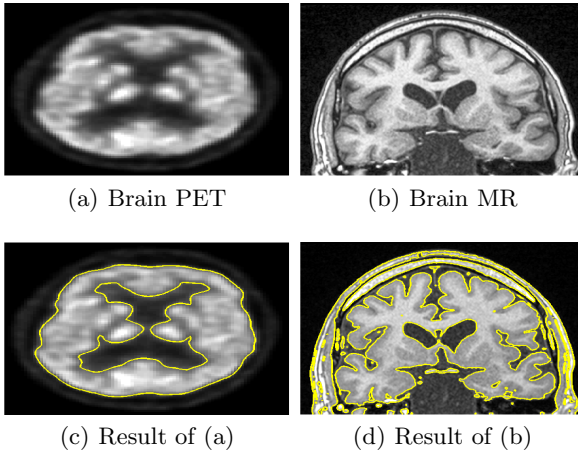


Fig. 8. Results of our method for some medical images

the parameters for tomography reconstruction, hence noisy. The original images and the initial contours are shown in the top row. The segmented images are shown in the bottom. It is not difficult to see that the results are very similar.

We report also on Fig. 8 some medical images which have a clinical importance. We present a segmentation of a brain PET image (Fig. 8(a)) and a brain MR image (Fig. 8(b)) to show the flexibility of our method. The obtained results (Figs. 8(c)-8(d)) are evaluated by our medical expert, as successful in detecting the boundary. One may notice on brain MR image that sulci and gyri are well delineated.

Next, we present also validation results with real expert-segmented thorax CT images and heart MR image. As can be seen in Fig. 9, the interior boundaries of the thorax are accurately extracted, as compared with the contour segmented

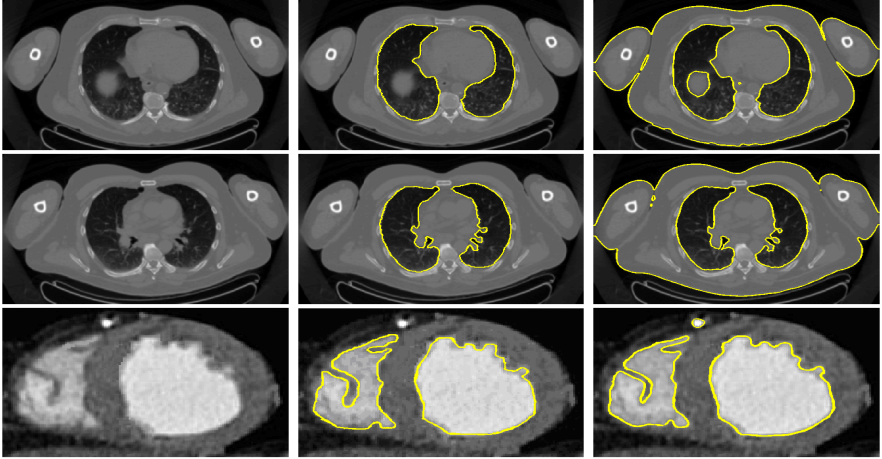


Fig. 9. Comparison with ground truth established by our expert on thorax CT images and heart MR image. Column 1: original image. Column 2: ground truth image. Column 3: our result.

Table 2. DSC values for third column of Fig. 9

		Row 1	Row 2	Row 3
DSC	left part	0.98	0.97	0.94
	right part	0.96	0.98	0.98

by our expert. Moreover, to quantitatively evaluate the accuracy of our results, we use Dice Similarity Coefficient (DSC) [10], which is defined as:

$$DSC = \frac{2N(S_1 \cap S_2)}{N(S_1) + N(S_2)} \quad (21)$$

where S_1 and S_2 represent the obtained segmentation and the ground truth, respectively, $N(\cdot)$ indicates the numbers of pixels in the enclosed set. The closer the DSC value is to 1, the better the segmentation is. Table 2 shows the DSC values of our method. From this table, our results are very close to the ground truth established by expert, since the DSC values LGFGD are very close to 1.

To validate the convexity of our model, proved in Section 2.2, we performed experiments using one of the synthetic images and one of the heart MR images. In every experiment, we placed the initial contour on a different position with a varying contour size. In every instance, the accuracy of segmentation is the same, which validates the robustness of the LGFGD model (see Fig. 10).

As mentioned above, the constant value $0 < \lambda < 1$ is used to control the influence of the global term F_1 and the local term F_2 . If the considered image is homogeneous, the value of λ can be close to 1 which will make the global term more dominant than the local term. On the other hand, if the tested image is

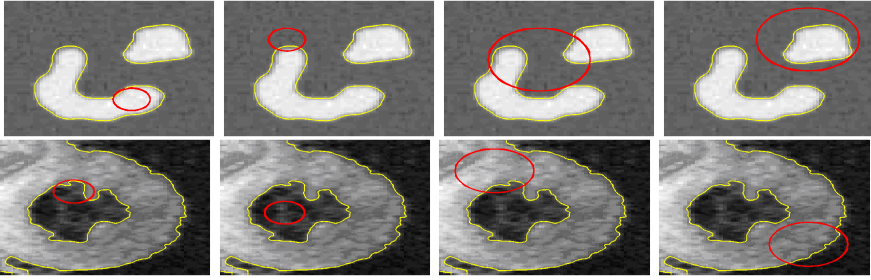


Fig. 10. Results of our model with the different initial contours

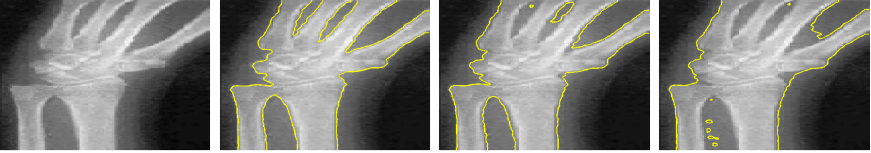


Fig. 11. Result on a hand X-Ray image with the different values of λ . From left to right: original image, results with $\lambda = 0.2, 0.5, 0.9$, respectively.

nonhomogeneous, the value of λ can be selected small enough to make the local term dominant. To support the last statements we applied our method on a hand X-Ray image with intensity inhomogeneity. One may observe that $\lambda = 0.2$ gives accurate result; if $\lambda = 0.5$, a part of the contour on the top of this image is missing; while in the case of $\lambda = 0.9$, when the global term is more dominant than the local term, the segmentation result is incorrect (see Fig. 11).

4 Conclusion and Future Works

This paper presents a novel convex and fuzzy energy based ACM. The model is named LGFGD and outperforms, in terms of accuracy and robustness with respect to noise, contemporary models such as the FEBAC [7], the LCV [16] and the LGIF [15]. By taking into account the local information with a distribution of different means and variances, the model is capable to segment nonhomogeneous regions. Furthermore, global information is also accounted for providing accurate performance in case of noise and weak boundaries. Moreover, our fuzzy energy function is convex, which makes the model invariant with respect to the initial position of the contour while giving the same accuracy and repeatability. The experimental results depend on the size of the Gaussian kernel. In terms of future work, we plan to investigate more effective implementation of the model and extend it to 3D images.

References

1. Boscolo, R., Brown, M., McNitt-Gray, M.: Medical Image Segmentation with Knowledge-guided Robust Active Contours. *Radiographics* 22(2), 437–448 (2002)
2. Caselles, V., Catta, F., Coll, T., Dibos, F.: A Geometric Model for Active contours in Image Processing. *Numerische Mathematik* 66, 1–33 (1993)

3. Chan, T., Vese, L.: Active Contour without Edges. *IEEE Trans. Image Process.* 10(2), 266–277 (2001)
4. Chen, Z., Qui, T., Ruan, S.: Fuzzy adaptive level set algorithm for brain tissue segmentation. In: *Proceedings of International Conference Signal Processing, Beijing, China*, pp. 1047–1050 (2008)
5. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active Contour Models. *Inter. J. Comp. Vi.* 1, 321–332 (1988)
6. Khoo, V.S., Dearnaley, D.P., Finnigan, D.J., Padhani, A., Tanner, S.F., Leach, M.O.: Magnetic resonance imaging (MRI): considerations and applications in radiotherapy treatment planning. *Radiother. Oncol.* 42, 1–15 (1997)
7. Krinidis, S., Chatzis, V.: Fuzzy energy-based active contour. *IEEE Trans. Image Process.* 18(12), 2747–2755 (2009)
8. Li, C., Kao, C., Gore, J., Ding, Z.: Implicit Active Contour Driven by Local Binary Fitting Energy. In: *Proc. IEEE Conf. Comp. Vi. and Pat. Recog.*, pp. 1–7 (2007)
9. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* 79, 12–49 (1988)
10. Shattuck, D.W., Sandor-Leahy, S.R., Schaper, K.A., Rottenberg, D.A., Leahy, R.M.: Magnetic Resonance Image Tissue Classification using a Partial Volume Model. *Neuroimage* 13, 856–876 (2001)
11. Sirakov, N.M., Ushkala, K.: An Integral Active Contour Model for Convex Hull and Boundary Extraction. In: *Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnação, M.L., Silva, C.T., Coming, D. (eds.) ISVC 2009, Part II. LNCS, vol. 5876*, pp. 1031–1040. Springer, Heidelberg (2009)
12. Song, B., Chan, T.: A fast algorithm for level set based optimization. *UCLA CAM Report 02-68* (2002)
13. Tylski, P., Stute, S., Grotus, N., Doyeux, K., Hapdey, S., Gardin, I., Vanderlinden, B., Buvat, I.: Comparative Assessment of Methods for Estimating Tumor Volume and Standardized Uptake Value in 18F-FDG PET. *J. Nucl. Med.* 51(2), 268–276 (2010)
14. Wang, L., He, L., Mishra, A., Li, C.: Active Contours Driven by Local Gaussian Distribution Fitting Energy. *Signal Processing* 89(12), 2435–2447 (2009)
15. Wang, L., Li, C., Sun, Q., Xia, D., Kao, C.Y.: Active contours driven by local and global intensity fitting energy with application to brain MR image segmentation. *Comp. Med. Imag. and Grap.* 33, 520–531 (2009)
16. Wang, X., Huang, D., Xu, H.: An efficient local Chan-Vese model for image segmentation. *Pattern Recognition* 43, 603–618 (2010)
17. Wang, Y., Bock, G.H., Van Klaveren, R.J., Van Ooyen, P., Tukker, W., Zhao, Y., Dorrius, M.D., Proenca, R.V., Post, W.J., Oudkerk, M.: Volumetric measurement of pulmonary nodules at low-dose chest CT: effect of reconstruction setting on measurement variability. *Eur Radiol.* 20(5), 1180–1187 (2009)
18. Xu, C., Pham, D.L.: Prince, J.L.: Medical Image Segmentation Using Deformable Models. *SPIE Handbook on Medical Imaging: Med. Image Anal.*, 29–174 (2000)
19. Zhu, S.C., Lee, T.S., Yuille, A.L.: Region Competition: Unifying Snakes, Region Growing, Energy/Bayes/MDL for Multi-band Image Segmentation. In: *Proceedings of ICCV 1995*, pp. 416–416 (1995)
20. Zijdenbos, A.P., Dawant, B.M.: Brain segmentation and white matter lesion detection in MR images. *Critical Reviews in Biomedical Eng.* 22, 401–465 (1994)

Author Index

- Asano, Tetsuo 90, 103
Attig, Anja 209
- Balázs, Péter 263
Bereg, Sergey 90, 103
Bhagvati, Chakravarthy 249
Bhattacharya, Bhargab B. 1
Bhowmick, Partha 1, 16
Biswas, Arindam 1, 16
Biswas, Soma 45
Buzer, Lilian 103
- Chen, Li 45
- Dutt, Mousumi 1
- Esbelin, Henri-Alex 223
- Geetha, H. 196
Gonzalez, Damien 223
- Hantos, Norbert 263
Herrera-Navarro, Ana Marcela 75
- Jiménez-Hernández, Hugo 75
- Kalyani, T. 196
Kamaraj, Thangasamy 181, 196
Kardos, Péter 128
Karmakar, Nilanjana 16
Kenmochi, Yukiko 234
- Lalitha, D. 166
Lukić, Tibor 274
Luong, Marie 298
- Malgouyres, Rémy 223
Mete, Mutlu 285
- Nagy, Benedek 143, 274
Ngo, Phuc 234
- Ou, Ye-Lin 285
- Palágyi, Kálmán 128, 263
Passat, Nicolas 234
Perner, Petra 209
Potluri, Anupama 249
- Rangarajan, K. 166
Rocchisani, Jean-Marie 298
- Samir, Chafik 223
Sekiya, Fumiki 59
Sirakov, Nikolay Metodiev 285, 298
Šlapal, Josef 115
Subramanian, K.G. 154
Sugimoto, Akihiro 59
- Talbot, Hugues 234
Terol-Villalobos, Iván Ramón 75
Thieu, Quang Tung 298
Thomas, Durairaj Gnanaraj 166, 181, 196
- Veelaert, Peter 31
Venkat, Ibrahim 154
Viennet, Emmanuel 298
- Wiederhold, Petra 154