Studies in Computational Intelligence 554

Anis Koubâa
Abdelmajid Khelil *Editors*

Cooperative Robots and Sensor Networks 2014

Springer

Studies in Computational Intelligence

Volume 554

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

For further volumes:
http://www.springer.com/series/7092

About this Series

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution, which enable both wide and rapid dissemination of research output.

Anis Koubâa · Abdelmajid Khelil
Editors

Cooperative Robots and Sensor Networks 2014

Springer

Editors
Anis Koubaa
Prince Sultan University, Riyadh,
Saudi Arabia
CISTER Research Unit, Portugal

Abdelmajid Khelil
Dept. of Computer Science
(Fachbereich Informatik)
TU Darmstadt
Darmstadt
Germany

ISSN 1860-949X ISSN 1860-9503 (electronic)
ISBN 978-3-642-55028-7 ISBN 978-3-642-55029-4 (eBook)
DOI 10.1007/978-3-642-55029-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014936017

Printed on acid-free paper

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

Preface

This book is the second volume on Cooperative Robots and Sensor Networks. The primary objective of this book is to provide an up-to-date reference for cutting-edge studies and research trends related to mobile robots and wireless sensor networks, and in particular for the coupling between them.

Indeed, mobile robots and wireless sensor networks have enabled great potentials and a large space for ubiquitous and pervasive applications. Robotics and wireless sensor networks have mostly been considered as separate research fields and little work has investigated the marriage between these two technologies. However, these two technologies share several features, enable common cyber-physical applications and provide complementary support to each other.

The book consists of ten chapters, organized into four parts.

The first part of the book presents three chapters related to localization of mobile robots using wireless sensor networks. Two chapters presented new solutions based Extended Kalman Filter and Particle Filter for localizing the robots using range measurements with the sensor network. The third chapter presents a survey on mobility-assisted localization techniques in wireless sensor networks.

The second part of the book deals with cooperative robots and sensor networks applications. One chapter presents a comprehensive overview of major applications coupling between robots and sensor networks and provides real-world examples of their cooperation. Two other chapters present applications for underwater robots and sensor networks.

The third part of the book is concerned with system engineering, where a first chapter proposes the integration of wireless sensor nodes with the robotic operating system (ROS) framework, which provides a hardware abstraction layer for robots and sensor nodes programming. A second chapter presents a comprehensive overview of mobile sensing and robotic platforms.

The fourth and last part presents two chapters about mobility management. The first chapter proposes a method in which one or more robots repair a WSN by deploying nodes at specific locations. The proposed method is capable of identifying node locations to ensure not only certain WSN connectivity properties but also a certain level of redundancy. The second chapter uses a fuzzy logic

approach to build an efficient mobility controller that aids sensor mobile entities to decide whether they have to trigger the handoff procedure and perform the handoff to a new connection position or not.

List of Reviewers

Abdelmajid Khelil
Anis Koubâa
Ramiro Martinez
Andreas Willig
Ye-Qiong Song
Andrea Zanella
Adel Ben Mnaouer
Enrico Natalizio
Geoffrey Hollinger
Carlos Sagues
Jiong Jin
Fumin Zhang
Rongxing Lu

Contents

Part III: System Engineering

Part IV: Mobility Management

Part I

Robots and Sensor Networks Localization

Simultaneous Localization of Robots and Mapping of Wireless Sensor Nodes

Andrea Zanella and Emanuele Menegatti

Department of Information Engineering, University of Padova, Via G. Gradenigo 6/B, 35131 Padova, Italy
{zanella,emg}@dei.unipd.it

Abstract. This chapter presents the use of a mobile robot to solve the problem of node localization in Wireless Sensor Network (WSN). The algorithms we propose are inspired by the algorithms developed in robotics to solve the robot localization problem exploiting landmarks in the environment. The robotics community developed algorithms of Simultaneous Localization and Mapping (SLAM), in which the robot pose is estimated while simultaneously mapping the position of the landmarks in the environment. Similarly, we simultaneously estimate the robot pose with the position of the nodes of a WSN using range measurements. The assumption is that a mobile robot can estimate the distance to nearby nodes of the WSN by measuring the Radio Signal Strenght (RSS) of the received radio messages. The intrinsic variability of RSS measurements due to interferences and reflections of radio signals, however, makes the ranging measure very noisy, thus limiting the accuracy of simple localization techniques. We first present a SLAM technique based on an Extended Kalman Filter (EKF-SLAM) to integrate RSS measurements from the different nodes over time, while the robot moves in the environment. Successively, we show that combining the EKF-SLAM algorithm with an initialization phase based on a Delayed Particle Filter (DPF) can greatly improve the performance of the algorithm. We then discuss possible extensions of the approach by using advanced RSS measurement techniques, and multidimensional scaling localization. Finally, we compare the different approaches on the same experimental testbed, both for indoor and outdoor scenarios.

1 Introduction

The problem of nodes localization in Wireless Sensor Network (WSN) has been since long recognized as an important and challenging issue and lot of research has been carried out in this context. Many solutions assume the presence of a limited number of nodes, called *beacons* or *anchors*, that know their own position and are used by other nodes to locate themselves through triangulation techniques. Many of these schemes make use of the Radio Signal Strenght (RSS) to determine a rough estimate of the distance between transmitter and receiver, an operation referred to as *ranging*. This approach offers the advantage of being readily employable in any radio device, since the RSS is supported by basically all radio transceivers. Another advantage of RSS-based ranging is that it does not require the node to be in line of sight with the beacon nodes, since the radio signal passes through obstacles, as persons, furniture or even

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014,*
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_1,

walls. Unfortunately, the range estimate based on RSS measurements is unreliable and subject to random fluctuations due to a number of environmental factors. Therefore, the accuracy that can be obtained with RSS-based localization techniques in indoor environments is rather poor, with errors of the order of 1 to 6 meters, depending on the number of beacons and the characteristics of the environments.

The presence of the robot, however, can drastically enhance the performance of the localization techniques. For instance, the robot, which is fairly well localized by virtue of the on-board odometers and navigation system, can act as a sort of mobile beacon drastically augmenting the number of reference signals to be used in classical localization algorithms [1]. However, the presence of robots opens the way to much more advanced and sophisticated localization methods. The chapter is organized as in the followings: In Sec. 2, we describe our approach, originally proposed in [2], based on the Simultaneous Localization and Mapping (SLAM) algorithm [3] realized by means of an Extended Kalman Filter (EKF) that merges the information provided by the robot's odometers with the RSS samples provided by the surrounding motes to simultaneously track the motion of the robot in the environment and refine the mapping of the motes in the area. At the end of the section, we observe that the accuracy of the mapping provided by EKF-SLAM is strongly affected by the first estimation of the mote position, which is required at the beginning of the SLAM procedure to initialize the system state Θ, which is a vector containing the current estimate of robot and motes locations. Therefore, in Sec. 3 we propose to couple the EKF-SLAM algorithm with a mote position initialization based on Delayed Particle Filter (DPF) and show, by means of experimental results, that this approach reduces both mean and variance of the final location estimate error with respect to the simple EKF approach, see also [4]. Successively, in Sec. 4 we discuss a method, originally introduced in [5,6] to increase the accuracy of the RSS-based ranging by exploiting the capability of the sensor nodes to operate on different RF channels, and using another localization method, namely the Weighted Multi-Dimensional Scaling (MDS), that is computationally lighter than EKF, and we discuss the extent to which inter-node RSS measurements may ameliorate the motes mapping. Finally, in Sec. 5 we provide an experimental performance comparison among EKF, particle filter with delayed initialization, MDS only, and MDS with inter-node measurements.

2 SLAM with EKF Only

As mentioned, a common and simple way to estimate the distance between the robot and each node of the WSN consists in measuring the RSS of the radio signal received by the sensor node connected to the robot, and inverting the signal power propagation law that describes the signal received power as a function of the distance from the transmitter, as better explained in the next subsection. However, as we anticipated, the range estimate based on RSS measurements is rather unreliable and subject to random fluctuations due to a number of environmental factors, such as temperature, humidity, daytime, presence of metal obstacles, and so on [7]. These problems are exacerbated in an indoor environment. A detailed characterization of the radio signal propagation and of RSS-based ranging can be found in [8].

A good example of mobile node positioning systems based on RSS ranging is RADAR [9], in which, in the set up phase, a map of the RSS received from static beacons by the

mobile node in each possible position of the working environment is built; and then, at the working phase this pre-built RSS map of the environment is used to determine the most likely position of the mobile node. Radar achieves discrete accuracy (i.e. 2 meters in 50% of the cases), but building the RSS map is a very time consuming and sometimes impractical especially in dynamic environments. The mobile node localization problem becomes even more complex when the position of the landmark is also unknown. Kurth, et al [10] have developed a system based on the time of flight of radio signals. The localization error, however, is approximately one meter. In [11] a similar approach based on a Robust Extended Kalman Filter (REKF)-based state estimator is reported, but a careful tuning of the parameters of the communication channel and of the weights associated to the measurements is requested. In the remainder of this section, we will provide a more detailed description of the SLAM algorithm we implemented and present some experimental results that we collected using the testbed described in Appendix 1. Throughout this chapter, we will use the term SLAM to refer to the problem of localizing a mobile node with range measurements obtained through RSS values measured from the radio signals exchanged with static sensor nodes with unknown positions. Thus, our algorithm is at the same time localizing the mobile node and mapping the (initially unknown) positions of the static nodes.

2.1 RSS-Based Range Measurement

To estimate the distance between two motes by measuring the RSS, it is required to know the signal propagation law. Unfortunately, the characteristics of the radio channel are largely affected by the environment and are difficult to capture by a unique model. In this chapter we refer to the standard path-loss model, which is generally considered reasonable for a large number of scenarios [12]. Accordingly, we estimate the distance d_i of a node whose radio signal is received with power P_i as:

$$d_i = d_0 \cdot 10^{\frac{A-P_i}{10 n_p}} \tag{1}$$

where[1]

- P_i radio signal power [dBm]
- d_0 is the far-field reference distance
- A is the nominal received power at the reference distance d_0
- n_p is the path loss coefficient

The power measure P [dBm] can be obtained from the RSS measurements returned by the node transceiver.

We observe that the distance estimate given by (1) will be affected by an error due to the noisy RSS readings. To limit these effects, we employ a simple pre-filter for the RSS measurements taking into account the robot odometry information. If the robot moved a short distance between two consecutive measurements, the RSS measured in the second robot position cannot change much from the first measurement. However,

[1] The channel model parameters considered in this section are $d_0 = 1.071467$ m, $A = -45$ dBm, $n_p = 2$.

being unknown the relative position of the robot and of the beacon, it is not possible to know whether the robot is approaching or moving away from the sensor node. However, the maximum variation in the RSS with respect to the previous measure cannot be bigger than either the one registered when the robot is directly moving toward the node or the one when the robot is directly moving away from the node.

To exemplify the concept, let us consider the case depicted in Fig. 1. Here, the sensor node is in the known position D, while the starting position of the robot is A. Now, suppose the robot moves to position B, along a straight path of length $\overline{AB}$. Projecting the distance $\overline{AD}$ over the line $\overline{BD}$, one finds the new point C. The length of the segment $\overline{CD}$ is the distance from the node D before the motion to B. After the motion, the two extreme cases are to end in C' (i.e. $\overline{CD}-\overline{AC}$) or to end in C'' (i.e. $\overline{CD}+\overline{AC}$). The actual displacement will be within these two extremes. Therefore, we even out the actual RSS measures received in C to the maximum expected RSS (i.e. the one corresponding to C') or to the minimum expected RSS (i.e. the one corresponding to C''). The remaining noise error is taken into account in the filtering matrices of the KF, as detailed below.

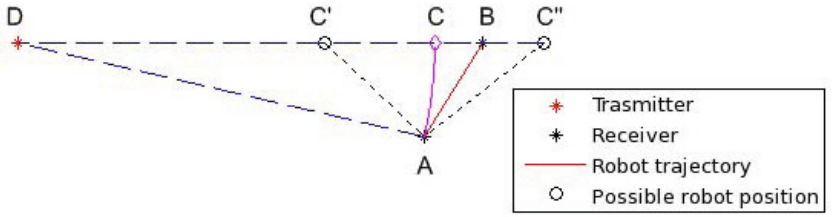


Fig. 1. A sketch of the two extreme situation of the robot total approach and of total moving away with respect to a transmitting node to understand the pre-filter to even out the measured RSS

2.2 SLAM Algorithm

The SLAM algorithm we propose is based on Extended Kalman Filter (EKF) and is very similar to the one presented in [13], so we will use the same formalism. EKF SLAM is a well-known technique that recursively solves the online SLAM problem where the map is feature-based [3,14]. It estimates at the same time the position of the robot and the position of the features: in our case the features are the motes.

The robot state (pose and heading) at time k is $q_k = [x_k, y_k, \theta_k]$. The motion model of the system is:

$$q_{k+1} = \begin{bmatrix} x_k + \Delta D_k \cos(\theta_k) \\ y_k + \Delta D_k \sin(\theta_k) \\ \theta_k + \Delta\theta_k \end{bmatrix} + \nu_k = f(\hat{q}_k, u_k) + \nu_k \tag{2}$$

where ν_k is a noise vector, while ΔD_k and $\Delta\theta_k$ are the odometric distance traveled by the robot in an update step and the heading change, respectively, both measured by the

robot's odometers. The term $u_k = [\Delta D_k, \Delta\theta_k]$ in the right-most expression accounts for the last input from the odometry. The system state matrix $A(k)$ is given by the Jacobian:

$$A(k+1) = \left.\frac{\partial f}{\partial q_k}\right|_{q=\hat{q}_k} = \begin{bmatrix} 1 & 0 & -\Delta D_k \cos(\theta_k) \\ 0 & 1 & \Delta D_k \sin(\theta_k) \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The input matrix $B(k)$ is given by the Jacobian:

$$B(k+1) = \left.\frac{\partial f}{\partial u}\right|_{q=\hat{q}_k} = \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

When a new input from the odometry u_k is available, the robot state is updated through the motion equation (2). The standard equations of the EKF updates the covariance matrices. When the robot receives a message from a sensor node i, it can extract the RSS and estimate a measure of range between itself in the state q_k and the sensor node in the position (x_k^i, y_k^i), this can be expressed as:

$$h\left(q_k, \left[x_k^i, y_k^i\right]^T\right) = \sqrt{\left(x_k - x_k^i\right)^2 + \left(y_k - y_k^i\right)^2}. \tag{5}$$

By linearizing with the Jacobian we can write:

$$H(k) = \left.\frac{\partial h}{\partial q}\right|_{q=\hat{q}_{k+1}} = \left[\frac{x_k - x_k^i}{\sqrt{\left(x_k - x_k^i\right)^2 + \left(y_k - y_k^i\right)^2}}, \frac{y_k - y_k^i}{\sqrt{\left(x_k - x_k^i\right)^2 + \left(y_k - y_k^i\right)^2}}, 0\right] \tag{6}$$

The above formulas are for the robot localization only, but in the case of SLAM also the wireless node must be localized so the new vector of state of the system is:

$$q_k = \left[x_k, y_k, \theta_k, x_k^1, y_k^1, \ldots, x_k^n, y_k^n\right] \tag{7}$$

where n is the total number of sensor nodes in the area. Also the matrices $A(k)$ and $B(k)$ must be modified an they become:

$$A(k+1) = \left.\frac{\partial f}{\partial q_k}\right|_{q=\hat{q}_k} = \begin{bmatrix} 1 & 0 & -\Delta D_k \cos(\theta_k) & 0 \cdots & 0 \\ 0 & 1 & \Delta D_k \sin(\theta_k) & 0 \cdots & 0 \\ 0 & 0 & 1 & 0 \cdots & 0 \\ 0 & 0 & 0 & 1 \cdots & 0 \\ & & & & \ddots\ 0 \\ 0 & 0 & 0 & 0 \cdots & 1 \end{bmatrix} \tag{8}$$

$$B(k+1) = \left.\frac{\partial f}{\partial u_k}\right|_{q=\hat{q}_k} = \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \tag{9}$$

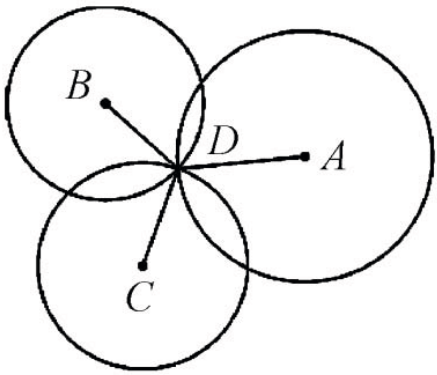


Fig. 2. An example of trilateration. *D* is the node with unknown position. *A*, *B*, and *C* are robot positions along the path of the robot where the distance robot-node was estimated using the RSS.

Now, while the time update remains as in (2), the measurement update requires that the Jacobian $H(k)$ is calculated considering the update of the wireless nodes position. Note that the only non-zero terms in $H(k)$ are those regarding the position of the robot and that of the sensor node involved in the last range measurement.

At the start-up, the EKF needs to be initialized with a *first guess* position of the nodes to be located. A possible way top obtain this first guess estimate is by using trilateration. Trilateration estimates the position of an object *D* as the intersection of the circumferences centered on (at least three) reference points (not lying on the same line), with radii equal to the distances between the object *D* and each reference point, as exemplified in Fig. 2. With noisy distance measurements, as in the case of RSS-based ranging, the circumferences may actually not intersect in a single point. In this case, the estimate of *D*'s position is obtained by applying a least-square method. This first estimate of a new node's position is then inserted in the EKF and updated in the following steps. Note that the covariance matrix is expanded by inserting the current robot covariance summed to the mean covariance of the measurements used in the trilateration.

Fig. 3 sketches the data flow in the SLAM algorithm considered in this work. First the RSS is measured, then the RSS values are evened out by the pre-filter. These values and the odometry are fed into the EKF, which is initialized for every node with the estimation provided by the trilateration algorithm. The output of the SLAM algorithm is a combined estimation of the positions of the robot and of the wireless nodes.

2.3 Experiments

Two kinds of experiments were performed to validate this approach: (i) the simultaneous estimation of the location of the robot and of the locations of the nodes of the WSN, *without any prior knowledge of the node locations*; (ii) the simultaneous estimation of the location of the robot and of the nodes *starting from a rough guess close to the real position*. This second experiment relates to a scenario in which a human operator (or a

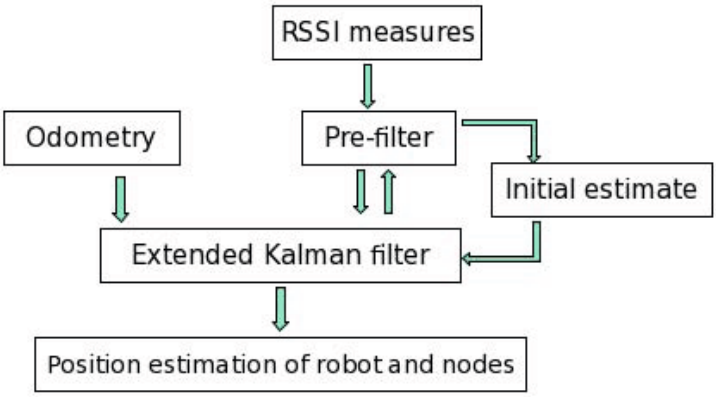


Fig. 3. The data flow in the implemented SLAM algorithm

Fig. 4. The robot and the WSN deployed in the test environment

loosely localized robot, like the helicopter reported in [15]) drops wireless nodes along his/its path and records a first localization that needs to be further improved.

The experiments were performed in the corridor depicted in Fig. 4, which is 30 m long and 6 m wide. The WSN composed by 8 nodes were deployed in a 6×3 meter area that was transversed by the robot.

2.4 Discussion

As can be seen in Fig. 5 and Fig. 6, the proposed approach can achieve a mean error on the position of the wireless sensor nodes between 0.5 m and 1.0 m. This is obtained with an on-line algorithm that updates the SLAM state variables every time a new measure is received.

The performance of this approach is however strongly affected by the initial guess of the node position. This is already evident comparing the final estimation of the node

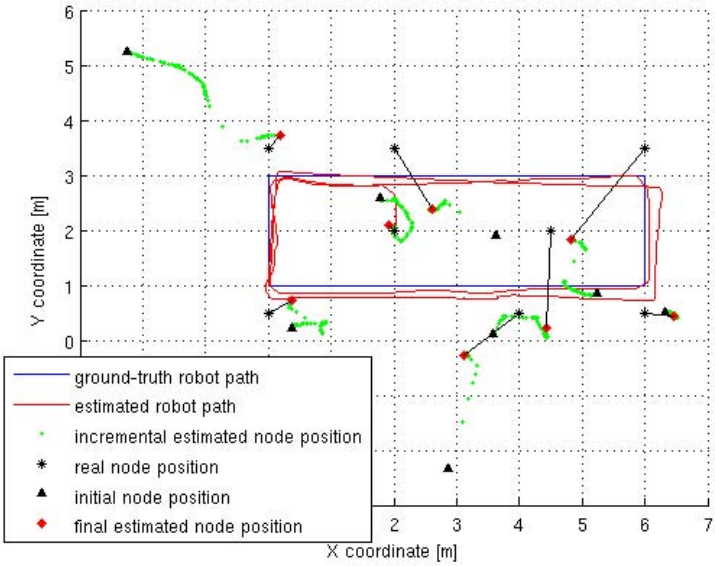


Fig. 5. The reconstruction of the robot path ad of the wireless nodes' positions starting from no knowledge

position obtained in the first experiment, in which the initial position of the sensor nodes is completely unknown and the EKF is initialized with the guess obtained by the trilateration algorithm, see Fig. 5, and the node position estimated when the SLAM algorithm is initialized with a coarse position of the nodes, see Fig. 6. In the latter case, the residual error is much smaller and the convergence is faster and better.

In order to gain better insights on the most critical parameters for the system, we performed repeated experiments and compared the residual error on the node positions and on the robot localization as reported in Fig. 7.

With respect to Fig. 7, `ATE` and `XTE` are respectively the average error in the robot position along the robot path (i.e., *Along-Track Error*) and in the direction perpendicular to the robot path (i.e. *Cross-Track Error*); `Cartesian` is the mean error between the estimated robot position and its ground-truth (*Cartesian distance*); while `Nodes` is the mean residual error between the final estimated nodes' positions and their ground-truth. The series `1,2,3` in Fig. 7 are those referring to the experiments with the triangulation algorithm to initialize the position of the newly encountered nodes along the path in the SLAM. The series `7,8,9` in Fig. 7 are those referring to the experiments in which a *good* first guess close to the real position of the node is given to the SLAM system. To generate a *good* first guess for every node, we chose to randomly draw a Cartesian point in a 2 meter range from the actual node position. The variance associated to this initial *good* guess is calculated as the average error for that distance given by the RSS curve.

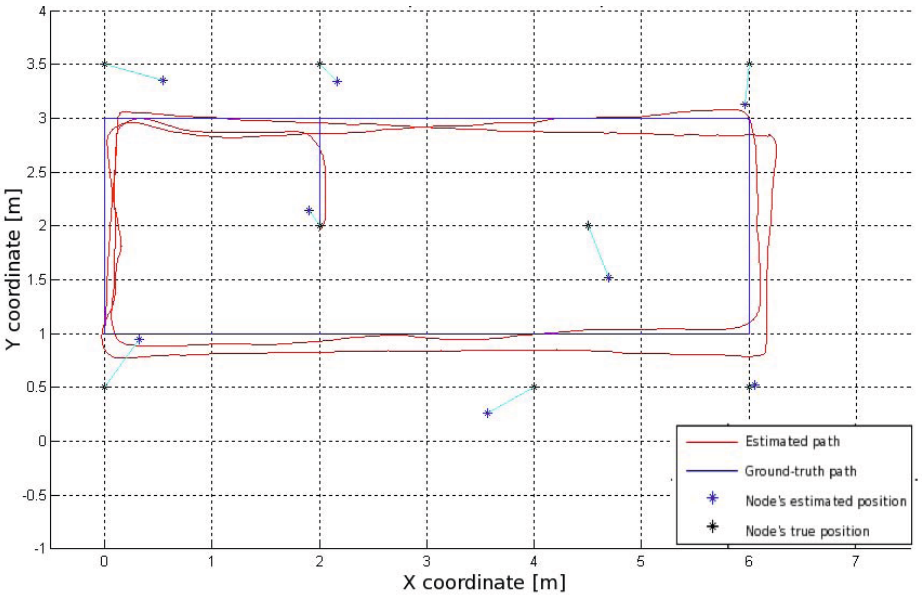


Fig. 6. The reconstruction of the robot path and of the wireless nodes' positions starting from a rough initialization of the nodes' position

The most evident result in Fig. 7 is that the residual error for the sensor nodes when a good guess is used is almost half of that obtained with initialization given by trilateration. Therefore, a more robust system to estimate the first guess on the node position is, then, needed.

It can also be noted that the standard deviation associated to the residual error on the nodes' positions (the lines over the bars of the histogram) is quite large. This is because there is a big difference in the initial guess of the node position among the different nodes. Some were initialized with a position error of more than 4 m, other nodes where initialized with less than 1 m error. The reason is that the first position guess given by trilateration strongly depends on the robot's path and on the structure of the environment.

The differences within the two series of Fig. 7 come from the management of multiple measurements collected by the robot when standing or turning on a certain spot. In the two series `1,7`, only the first three measurements in chronological order of reception are inserted in the SLAM algorithm. In series with index `2,8`, only the three measurements with highest RSS are inserted in the SLAM algorithm. Obviously, the SLAM state is updated again when the robot changes position. In series `3,9` a fully off-line approach is tested and only the 40 measures with the highest RSS for each node are inserted in the SLAM algorithm. There are no substantial differences among the three cases, so that the simplest approach (i.e., the first) is adopted.

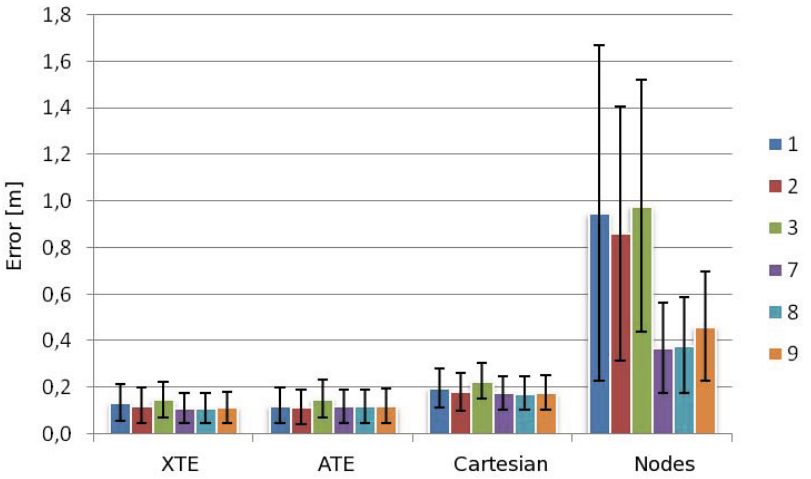


Fig. 7. The comparison between the average residual error on the robot and nodes' positions obtained by the SLAM algorithm in the two situations: (i) starting from no prior knowledge on the node position (1, 2, and 3 in the legend), (ii) starting from an initial coarse localization close to the real position (7, 8, and 9 in the legend)

Summing up, results obtained by using SLAM are quite good if one consider they were obtained: (i) in an indoor environment, (ii) without any calibration of the receiving/transmitting systems on the nodes, (iii) without a specific model of the actual communication channel of the environment in which the experiments where performed, (iv) exploiting only the RSS measures to calculate the robot-node distance, (v) no node-to-node communication has been used.

Much better results on the experiments without any *a priori* knowledge could be obtained with a better initial guess of the wireless nodes' positions. The next section will further investigate this aspect.

3 SLAM with Position Initialization Based on Delayed Particle Filter

As shown in Section 2, the largest part of the residual error in the estimation of the mote positions is due to a wrong initialization of the motes in the Kalman filter at the beginning of the SLAM procedure. In this section, we propose the use of a particle filter for implementing a delayed initialization of the mote positions.

In fact, the initial guess of motes' position that can be obtained by using trilateration techniques with raw RSS ranging is vert noisy. The technique we adopted to improve the initial guess is to model the distance estimated from an RSS measurements as a Gaussian random variable. Hence, each RSS measurement collected by the robot was associated to an annular ring, centered on the robot position, and representing the Gaussian probability distribution of the distance estimated from that RSS sample. The peak

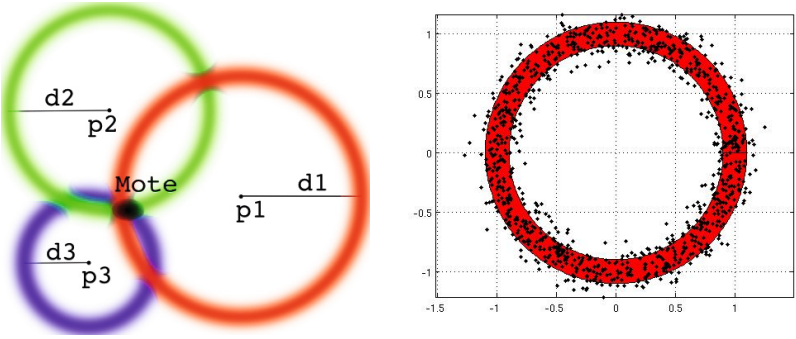
of the distribution was reached along the circle with radius equal to the nominal distance r estimated from the RSS values, as for (1). The standard deviation σ was previously calculated with a calibration table in which at each measured distance r is associated a standard deviation. Fig. 8 depicts the situation after three measurements (like in the case of Fig. 2). The estimated mote position now lays somewhere in the intersection of the three annular rings (or better in the intersection of the three Gaussian distributions associated to the three distances), see the grayish ellipse in Fig. 8(a). We modeled all this by instancing for each mote a particle filter in which the samples are hypotheses on the position of the mote drawn from a normal distribution. In Fig. 8(b), after the reception of the first message from a mote, the robot calculates the mote-robot distance r from the RSS. The samples are uniformly distributed around a ring of radius r and standard deviation σ. The weight associated to every particle is given by the Gaussian distribution.

When the robot receives new messages from the mote and calculates new estimations of the mote distance, these correspond to new annular probability distributions centered in the corresponding robot position. We apply a Sampling Importance Resampling algorithm each time a new distance estimation is available, making the particle to condensate toward the real mote position. To solve the problems of the frequent outliers that can arise with RSS-based ranging, especially in the first steps of the initialization, we also distributed uniformly in the environment a certain percentage of the particles (5% in our experiments). This enables the particle filter to recover from a totally wrong estimation as in the case of the well-known "*kidnapped robot problem*" [14].

We successfully tested this delayed initialization strategy with some dataset where the range-only SLAM approach described in Sec. 2 failed due to a wrong initial guess of the mote positions. With the proposed delayed strategy based on a particle filter, the initial mote position is estimated over several consecutive range measures, filtering out in a probabilistic way the outliers. An example of SLAM convergence with the proposed approach is depicted in Fig. 9. Before adding a new mote into the map, the robot maintains and updates for some steps the corresponding particle filter. When uncertainty on the mote position given by the particle filter drops below a certain threshold, the node is initialized and it is inserted in the SLAM process described in the previous section as initialization of the EKF. This approach proved to be effective also with large noise in the RSS range measurements. The Delayed Particle Filter approach makes it possible to have a *good* guess for initializing the EKF of the SLAM process.

4 SLAM with Multichannel RSS Measurements and MDS Localization

In this section we consider a simple technique to improve RSS-ranging accuracy by exploiting the capability of the sensor nodes to operate on different RF channels. Then, we propose a different localization method, namely the Weighted Multi-Dimensional Scaling (MDS), that is computationally lighter than EKF. We then extent the method to include inter-object RSS measurements.



(a) (b)

Fig. 8. (a) An example of trilateration. P_1, P_2, and P_3 are robot positions along the path of the robot where the corresponding distances robot-node d_1, d_2, and d_3 were estimated using the RSS. (b) The annular ring and the samples drawn from the underlying Gaussian distribution associated to a distance estimated by the robot measuring the RSS of the messages received from a mote.

4.1 Multichannel RSS-Based Ranging

As observed in [16], the variability of the RSS in indoor environment is mainly due to the self-interference among multiple copies of the transmitted signal that reach the receiver following different paths, and hence with random phase shifts, as resulting from multiple reflections on floor, ceiling and close-by objects. At the frequency of 2.4 GHz, which is typically used for short range wireless transmissions, displacements of few centimeters of transmitter and/or receiver can result in a totally different combination of the signal reflections at the receiver, with a significant variation of the received signal power.

Now, the same effect can be obtained by changing the carrier frequency without moving the nodes. As an example, by increasing the carrier frequency of the radio signal from 2.4 GHz to 2.45 GHz, the phase shift between the direct signal and a copy that follows a path 3 meters longer (e.g., ceiling reflection) will be $\approx \pi$. This suggest that it is possible to reduce the variability of the RSS-based ranging by collecting multiple RSS samples on different RF channels and, then, using their mean value in the ranging equation [8].

Clearly, to gather RSS measurements on different channels, nodes need to coordinate in order to concordantly change the carrier frequency. A possible, basic algorithm to perform multi-channel RSS harvesting is described below.

The process occurs in successive rounds. Each round is initiated by the robot that broadcasts an `RSS_GET` message. This message contains the list of sensors that are required to collect RSS samples, and the transmission order of the nodes. Channel access occurs according to a Time Division Multiple Access (TDMA) scheme: time is partitioned in transmission slots of constant duration (slightly longer than the transmission time of a full data packet), and each node is assigned to a single slot in an exclusive

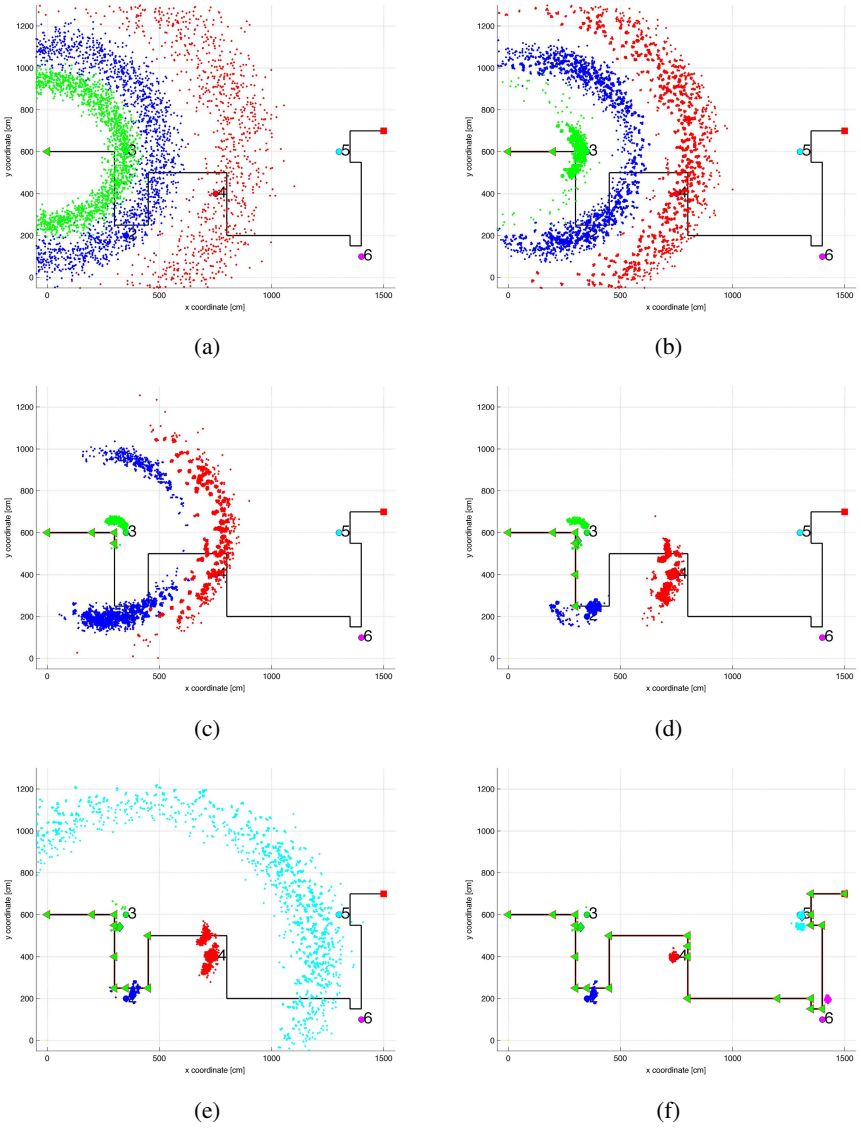


Fig. 9. The delayed initialization with the particle filter in action while the robot (the green triangle) is moving in the environment. (a) The robot received the first distance RSS thus the annular rings are created around the robot and it encompasses the motes; (b) The robot moved; new measures are inserted in the particle filter and the samples cluster around the mote position; (c) the particle filter converged and the node is initialized in the Extended Kalman Filter. (d-f) More complex situation along the robot path with several showing the convergence of multiple particle filters.

manner. Each node listed in the RSS_GET message, then, waits for its assigned slot and, then, broadcasts an RSS_REPORT message that contains the vector of RSS values measured in the previous slots. Furthermore, the RSS_GET and RSS_REPORT messages will also carry an indication of the RF channel that will be used in the following round. In this way, nodes that miss the robot's packet, but overhear a report message can synchronize again in the following round. We observe that the number of RSS samples reported by the nodes is not homogeneous across the round, since the first nodes that transmit have not yet received messages from the others. To overcome this drawback, the robot may permute the transmission order of the nodes in each subsequent rounds. Furthermore, each round may be repeated multiple times, without changing channel.

The communication is unreliable and no acknowledgment mechanism is considered. If a node does not receive *any* message (either from the robot or other nodes) for a interval $T_{timeout}$, it switches back to the default operating frequency.

When the multichannel RSS harvesting is complete, the robot can move into a new location and repeat the full process.

4.2 MDS

In the following we describe the MDS algorithm for nodes mapping that we used in our experiments. We describe the algorithm in its general form, here denoted as *MDS Internode*, which assumes that RSS measurements are available between *any* pair of nodes. In other words, besides RSS measurements between sensors and robot, it also considers RSS measurements performed between different sensors. In many practical cases, however, inter-sensor measurements may not be available. In this case, the sensor-sensor measurements can be neglected by setting to zero their weights (as explained below) and the resulting algorithm is simply referred to as *MDS*.

Let us enumerate from 1 to n the nodes included in the mapping process. Furthermore, let $n+1, n+2, \ldots, n+k$ denote the locations where the robot stopped to collect RSS samples from the surrounding objects. In the following, we denote these positions as *virtual beacon* nodes. Let $\theta_i = (x_i, y_i)^T$ the vector of cartesian coordinates for node i. Our aim is to determine an estimate of θ_i for $i = 1, 2, \ldots, n$ knowing the exact position of the virtual beacons and the ranging values given by (1) (when using either single channel RSS measurements or multi-channel average RSS measurements, depending on the considered case). The MDS approach consists in minimizing the following cost function

$$S(\Theta_k) = \sum_{i=1}^{n} \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} (\hat{d}_{i,j} - d_{i,j}(\Theta_k))^2 + \sum_{j=n+1}^{n+k} 2 w_{i,j} (\hat{d}_{i,j} - d_{i,j}(\Theta_k))^2 \right). \tag{10}$$

where $\Theta_k = [\theta_1, \ldots, \theta_{n+k}]$ is the state vector, $\hat{d}_{i,j}$ is the estimated distance between smart object i and virtual beacon j, whereas $d_{i,j}(\Theta_k)$ is the distance between the same nodes given the state vector Θ_k. Finally, the scalar $w_{i,j}$ accounts for the accuracy of $\hat{d}_{i,j}$ and is given by

$$w_{i,j} = e^{-\bar{P}^2_{rx_{i,j}} / P^2_{th}} \tag{11}$$

where $\bar{P}_{rx_{i,j}}$ and P_{th} are respectively the power received by node i from node j (possibly averaged over different channels), and the power threshold for ranging.

When inter-sensor RSS measurements are not available, the weights $w_{i,j}$ for RSS measurements between pairs of sensor nodes are all equal to zero, i.e.,

$$w_{i,j} = \begin{cases} 0, & \forall i \in \{1,\ldots,n\}, \forall j \in \{1,\ldots,n\} \\ e^{-\bar{P}^2_{rx_{i,j}}/P^2_{th}}, & \forall i \in \{1,\ldots,n\}, \forall j \in \{n+1,\ldots,n+k\} \end{cases} \tag{12}$$

and the cost function becomes

$$S(\Theta_k) = \sum_{i=1}^{n} \sum_{j=n+1}^{n+k} 2w_{i,j} \left(\hat{d}_{i,j} - d_{i,j}(\Theta_k)\right)^2 . \tag{13}$$

For the sake of generality, in the following we consider the MDS Internode case, bearing in mind that the same procedures apply to the pure MDS case by simply using weights (12) in place of (11).

The minimization of $S(\Theta_k)$ cannot be performed in closed form, but the problem can be solved iteratively. Given the state vector at the iterative step h, $\Theta_k^{(h)} = \left[\theta_1^{(h)}, \ldots, \theta_{n+k}^{(h)}\right]$, the next state can be computed by applying this simple updating function (see [17] for the details):

$$\theta_i^{(h+1)} = a_i \Theta_k^{(h)} \mathbf{b}_i^{(h)} \tag{14}$$

where

$$a_i = \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} + \sum_{j=n+1}^{n+k} 2w_{i,j} \right)^{-1} \tag{15}$$

and $\mathbf{b}_i^{(h)} = \left[b_{i,1}^{(h)}, \ldots, b_{i,n+k}^{(h)}\right]^T$ is a vector whose entries are given by:

$$b_{i,j}^{(h)} = \begin{cases} \alpha w_{i,j} \left(1 - \frac{\hat{d}_{i,j}}{d_{i,j}(\Theta_k^{(h)})}\right) & j \neq i \\ \sum\limits_{\substack{\ell=1 \\ \ell \neq i}}^{n} w_{i,\ell} \dfrac{\hat{d}_{i,\ell}}{d_{i,\ell}(\Theta_k^{(h)})} + \sum\limits_{\ell=n+1}^{n+k} 2w_{i,\ell} \dfrac{\hat{d}_{i,\ell}}{d_{i,\ell}(\Theta_k^{(h)})} & j = i, \end{cases} \tag{16}$$

with $\alpha = 1$ if $j \leq n$ and $\alpha = 2$ otherwise. The iterative procedure stops when $S(\Theta_k^{(h-1)}) - S(\Theta_k^{(h)}) < \varepsilon$ for a certain ε. We observe that, although the updating equations are simple to compute, the number of operations grows linearly with the number of virtual beacons, so that the execution of the MDS algorithm progressively slows down as the number of sampling positions increases. The same scalability problem, however, affects the other localization algorithms considered in our previous work. In particular the complexity of EKF is roughly order of $O(mn^3)$, with m number of steps of the robot and n number of objects in the area, while the complexity of the MDS algorithms is instead $O(nmL)$, where L is the number of recursions performed by the algorithm to converge to the solution. The value of L grows with the number of sampling positions m, though the dependence of L on m is not available in an explicit form. Nonetheless, we experimentally found that MDS is lighter than EKF for reasonable values of n and m.

5 Comparison Experiments and Conclusions

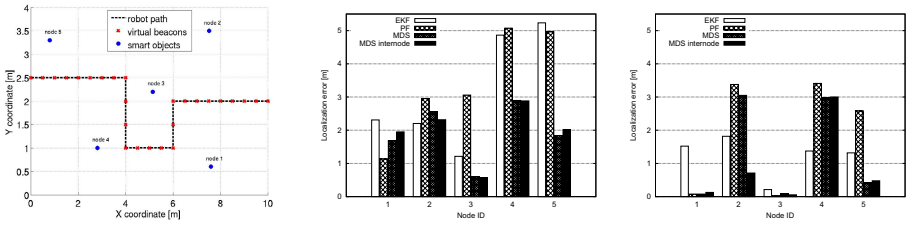
In this section, we compare the performances of the three SLAM algorithms introduced in the previous sections, namely EKF, PF, MDS, and MDS Internode in terms of mean localization error. Note that, MDS makes use of assumptions on the nature of the sensed signal that EKF neglects, being more general. Therefore, the comparison is not completely fair and results are significant limitedly to the considered problem and scenarios.

In Fig. 10 and Fig. 11 we report the results obtained in indoor (IN) and outdoor (OUT) scenarios, respectively. Images (a) show the location of the nodes in the experiments and the trajectory followed by the mobile robot across the area. Each RSS harvesting station along the path is marked by a cross. Figures (b) and (c) show the final localization error of each node for the different algorithms, when using single-channel (b) and multichannel ranging (c), respectively. Furthermore, in Table 1 we collect the mean localization errors over all the nodes, in the different cases.

First of all, comparing the results achieved with the same setting in the two scenarios, we observe that all the algorithms provide better location estimate in outdoor, because of the less severe multi-path fading. Second and more interesting, observing the results reported in figures (b) and (c) in the respective scenarios, we see that in almost all the cases the localization error of all the considered algorithms is reduced when using multichannel ranging rather than single-channel ranging. This experimental evidence confirms the positive effect of multichannel RSS averaging in reducing the uncertainty of the ranging estimate, as explained in [8]. The counterpart is that the collection of RSS samples over multiple channels requires a more sophisticated communication algorithm and, in general, may take a longer time. However, we observe that with single-channel ranging, it is still necessary to collect multiple RSS samples for each pairs of nodes, in order to average the fast fading term. Conversely, with multichannel ranging we collect one or a few RSS samples in each RF channel, but we repeat the operation in successive time instants in different channels, so that the fast fading is still averaged out when taking the mean RSS value. Therefore, the multichannel RSS ranging takes approximately the same time as single-channel ranging.

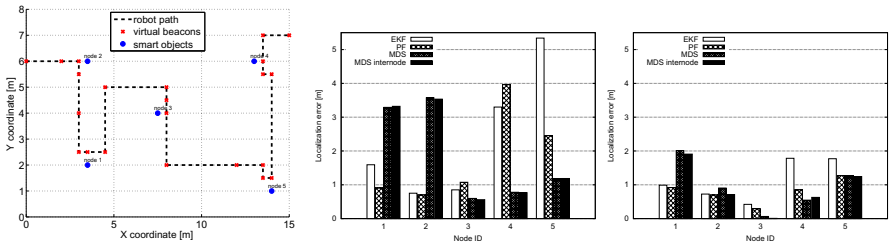
Indeed, the time taken to collect RSS samples over k different channels can be roughly estimated as $M = k(nT + S)$, where n is the number of in-range nodes, T is the slot duration and S is the switching delay that accounts for the time taken by the nodes to switch to the next channel and receive the next `RSS_GET` packet from the robot. With the TmoteSky sensor nodes used in the experiments (see Appendix), the slot time turns out to be approximately equal to $T \simeq 10$ [ms], while the switching time is $S \simeq 50$ [ms]. Hence, collecting RSS samples over $k = 4$ maximally spaced-apart RF channels from $n = 10$ nodes takes approximately $M = 600$ [ms].

Finally, we note that the MDS Internode algorithm yields, in a few cases, slightly worse localization accuracy than the standard MDS. In the other cases, however, the MDS Internode scheme may provide significant improvements as, for instance, for nodes 2 in Fig. 10. The reason is that rough internode ranging estimates may generally impact negatively the localization accuracy provided by the MDS algorithm when the first guess of the nodes position is good. However, in case the nodes are severely misplaced at the beginning of the MDS algorithm, the availability of internode ranging information makes it possible to correct this deficiencies. This is the case of node 2 in



(a) (b) (c)

Fig. 10. Indoor scenario: (a) experimental setup, (b) mean estimate error using single-channel RSS ranging, (c) mean estimate error using multichannel RSS-ranging



(a) (b) (c)

Fig. 11. Outdoor scenario: (a) experimental setup, (b) mean estimate error using single-channel RSS ranging, (c) mean estimate error using multichannel RSS-ranging

Fig. 10. In fact, observing the time evolution of the state vector Θ_k during the execution of the mapping algorithms (not reported here for space constraints) we could see that the initial guess for this node position, obtained by applying the Particle Filter initialization approach, was close to the position of node 1, which is actually symmetric with respect to the robot trajectory. With the path followed by the robot in this experiment, the EKF, PF and MDS algorithms were not able to recover node 2 from that erroneous initialization, so that the final localization error was large. Conversely, using the internode ranging information between nodes 1 and 2, the MDS Internode algorithm was able to correct the initial error and enhance the accuracy of the final position estimation of node 2.

The attentive reader might have noted that mean localization errors reported in the experiments of Sec. 2 are smaller then the ones reported in the experiments in Sec. 5. Actually, experiments presented in the two sections are different: the path of the robot is different, the position of the motes is different, and the environments are different. However, in our experience, the main reason for the difference between the (smaller) error reported in Sec. 2 and the (bigger) errors reported in in Sec. 5 is the robot path. In our experiments, we saw that if the robot goes all around a node, it is able to more precisely locate the node. In Sec. 2 the robot moved along two loops around (or inside the rectangle of) the nodes. In Sec. 5 the robot passed by the nodes just once with no loop, resulting in a bigger residual error in the estimation of the position of the motes.

Table 1. Mean localization errors for indoor (first and second rows) and outdoor (third and fourth rows) environments, using single-channel and multichannel ranging

	EKF	PF	MDS	MDS Internode
IN singlech	3.16 m	3.44 m	1.92 m	1.95 m
IN multich	1.25 m	1.9 m	1.32 m	0.87 m
OUT singlech	2.37 m	1.82 m	1.88 m	1.87 m
OUT multich	1.14 m	0.8 m	0.95 m	0.9 m

Acknowledgement. The authors wish to acknowledge all the researchers that have contributed to set up and run the experiments that are at the basis of the results reported in this chapter. In particular, we wish to thank for their work, assistance, and suggestions the following persons (in alphabetic order): Andrea Bardella, Matteo Danieletto, Marco Mina, Alberto Pretto, and Francesco Zorzi.

Appendix 1: Experimental Testbed

The testbed used for the experiments consists of an autonomous mobile robot and of a WSN composed of a dozen of wireless sensor nodes.

The Wireless Sensor Network

The experiments have been carried out using two different platforms for wireless sensor nodes, namely the EyesIFX and TmoteSky.

EyesIFX sensor nodes are produced by Infineon Technologies. The EyesIFX can be programmed and powered via USB for easy interconnection with other digital devices [18,19]. Each wireless node is equipped with a radio interface that provides 19.2 kbps transmission rate by using an FSK modulation in the 868.3 MHz band. The platform is fitted with light and temperature sensors. Furthermore, an integrated Received Signal Strength circuit can be used to estimate distance.

The TmoteSky radio transceiver is the Chipcon CC2420, whose PHY and MAC is compliant to the IEEE 802.15.4 standard, operating in ISM band at 2.4 GHz and providing a bit rate of 250 kbit/s. The module also provides an 8-bit register named Received Signal Strength Indicator (RSS), whose value is proportional to the power of the received radio signal. The core of the mote is the MSP430, a Texas Instrument low-power microcontroller, which is used to control the communication and sensing peripherals. The microcontroller is provided with 10 kB of RAM and 48 kB of integrated flash memory, used to host operating system and programs, whereas additional 1 MB of flash memory is available for data storing. Besides, the board is equipped with integrated light and humidity sensors. Motes have been programmed in NesC, using the TinyOS open-source operating system.

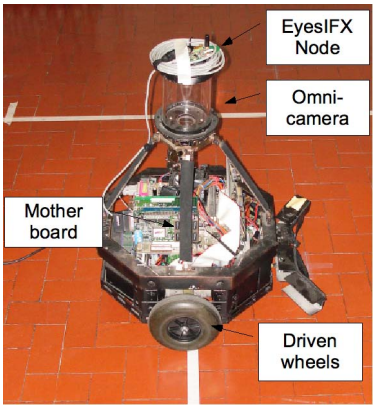


Fig. 12. The robot and its equipment

The robot

The robot is a wheeled custom designed robot based on the Pioneer 2 by MobileRobots Inc, see Fig. 12. The robot is equipped with a standard ATX motherboard with an Intel 1,6 GHz Intel Pentium 4, a 256 MB RAM and a 160 GB hard disk, running Linux OS. The only on–board sensors are: an omnidirectional camera (composed of a standard CCD camera and a convex omnidirectional mirror) and the odometers connected to the two driven wheels. Communication are provided by a PCMCIA wireless ethernet card toward the laboratory Intranet and by an EyesIFX node connected to one of the robot's USB ports toward the WSN. The EyesIFX sensor is mounted on top of the omnidirectional camera.

Mounting an EyesIFX node on the robot enables to integrate the robot in WSN. Thus, among the rest, the robot itself can act as a gateway for the WSN services. To seamless use the special command set of the EyesIFX mounted on the robot from the robot control software, we exploited the robotics architecture MIRO [20,21].

Miro is a distributed object oriented framework for mobile robot control, based on CORBA[2] technology. The Miro core components have been developed in C++ for Linux, but due to the programming language independency of CORBA further components can be written in any language and on any platform that provides CORBA implementations. The Miro core components have been developed under the umbrella of ACE[3], an object oriented multi-platform framework for OS-independent interprocess, network and real time communication. They use TAO[4] as their ORB[5]. TAO is a CORBA implementation designed for high performance and real time applications.

Conversely, the communication process is managed by two different classes: the `EyesConnection` class and the `EyesEvent` class. The first class takes care of the

[2] Common Object Request Broker Architecture.
[3] Adaptive Communications Environment.
[4] The ACE ORB.
[5] Object Request Broker.

communication *from the robot to sensor* (and so, toward the WSN). The second class takes care of the communication *from the sensor to the robot* (and so, from the WSN). This class is an object wrapper for the event handler registered with the ACE Reactor-Task.

To use the services of this structure, we exploit TAO services, in particular the *Name Service*. When the Name Server is running, one can share services easily adding them to a server or one can ask for a service with a request to the Name Server. We use this service with the multicast support which hides also the Name Server address. TAO implements several ways to share data, we decided to use the producer/consumer paradigm. TAO translate this in a supplier/consumer structure which needed a Notify Channel to share data. Therefore, every time new data are available from the WSN, these are pushed on the robot and are ready to be read by the robot software.

References

1. Sichitiu, M., Ramadurai, V.: Localization of wireless sensor networks with a mobile beacon. In: 2004 IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, pp. 174–183 (2004)
2. Menegatti, E., Zanella, A., Zilli, S., Zorzi, F., Pagello, E.: Range-only SLAM with a mobile robot and a Wireless Sensor Networks. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 8–14 (May 2009)
3. Montemerlo, M., Thrun, S.: FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics. Springer (January 2007)
4. Menegatti, E., Danieletto, M., Mina, M., Pretto, A., Bardella, A., Zanella, A., Zanuttigh, P.: Discovery, localization, and recognition of smart objects by a mobile robot. In: Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., von Stryk, O. (eds.) SIMPAR 2010. LNCS, vol. 6472, pp. 436–448. Springer, Heidelberg (2010)
5. Menegatti, E., Danieletto, M., Mina, M., Pretto, S., Zanconato, A., Zanuttigh, P., Zanella, A.: Autonomous discovery, localization and recognition of smart objects through WSN and image features. In: Proc. of: IEEE International Workshop Towards Smart Communications and Network technologies applied on Autonomous Systems (SaCoNAS), IEEE GLOBECOM (2010)
6. Bardella, A., Danieletto, M., Menegatti, E., Zanella, A., Pretto, A., Zanuttigh, P.: Autonomous robot exploration in smart environments exploiting wireless sensors and visual features. Annals of Telecommunications 67(7-8), 1–15 (2012), doi:10.1007/s12243-012-0305-z
7. Thelen, J., Goense, D., Langendoen, K.: Radio wave propagation in potato fields. In: First workshop on Wireless Network Measurements (co-located with WiOpt 2005), Riva del Garda, Italy (2005)
8. Zanella, A., Bardella, A.: RSS-based ranging by multichannel RSS averaging. IEEE Wireless Communications Letters (2013), doi:10.1109/WCL.2013.100913.130631
9. Bahl, P., Padmanabhan, V.: Radar: an in-building RF-based user location and tracking system. In: Proceedings of the IEEE INFOCOM 2000, vol. 2, pp. 775–784 (2000)
10. Kurth, D., Kantor, G., Singh, S.: Experimental results in range-only localization with radio. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 1, pp. 974–979 (October 2003)
11. Pathirana, P., Bulusu, N., Savkin, A., Jha, S.: Node localization using mobile robots in delay-tolerant sensor networks. IEEE Transactions on Mobile Computing 4(3), 285–296 (2005)
12. Goldsmith, A.: Wireless Communications. Cambridge University Press (2005)

13. Djugash, J., Singh, S., Kantor, G., Zhang, W.: Range-only slam for robots operating cooperatively with sensor networks. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, pp. 2078–2084 (2006)
14. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. In: Intelligent Robotics and Autonomous Agents. The MIT Press (August 2005)
15. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004), April 26-May 1, vol. 4, pp. 3602–3608 (2004)
16. Bardella, A., Bui, N., Zanella, A., Zorzi, M.: An experimental study on IEEE 802.15. 4 multichannel transmission to improve RSSbased service performance. In: Proceedings of Real-World Wireless Sensor Networks, pp. 154–161. Springer, Heidelberg (2010)
17. Costa, J.A., Patwari, N., Hero III, A.O.: Distributed weighted-multidimensional scaling for node localization in sensor networks. ACM Transactions Sensor Networks 2, 39–64 (2006)
18. Crepaldi, R., Harris, A., Scarpa, A., Zanella, A., Zorzi, M.: Signetlab: deployable sensor network testbed and management tool. In: Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys 2006), New York, NY, USA, pp. 375–376 (2006)
19. Crepaldi, R., Harris, A., Scarpa, A., Zanella, A., Zorzi, M.: Testbed implementation and refinement of a range-based localization algorithm for wireless sensor networks. In: 3rd IEE Mobility Conference 2006, October 25-27 (2006)
20. Kraetzschmar, G.K., Utz, H., Sablatnög, S., Enderle, S., Palm, G.: Miro - middleware for cooperative robotics. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 411–416. Springer, Heidelberg (2002)
21. Utz, H., Sablatnog, S., Enderle, S., Kraetzschmar, G.: Miro - middleware for mobile robot applications. IEEE Transactions on Robotics and Automation 18(4), 493–497 (2002)

Robot-WSN Cooperation for Scalable Simultaneous Localization and Mapping*

Arturo Torres-González, José Ramiro Martinez-de Dios, and Aníbal Ollero

Robotics, Vision and Control Research Group, University of Seville
Camino de los Descubrimientos s/n, 41092 Sevilla, Spain
{arturotorres,jdedios,aollero}@us.es

Abstract. This chapter proposes a scalable SLAM method that uses range measurements sensed by Wireless Sensor Networks (WSN) nodes. It integrates direct robot-node range measurements as well as measurements static nodes take from other nodes –internode measurements– exploiting WSN nodes capability of organizing into networks. To cope with the high number of measurements, the method adopts an PF-EIF SLAM filter, significantly more scalable and efficient than traditional schemes based on EKF. The integration and use of internode measurements can significantly improve map and robot estimations accuracy. It can also anticipate the deployment and convergence of the Particle Filters (PFs), resulting in lower computational burden. The proposed method has been compared with traditional schemes based on EKF both in simulation and in experiments carried out in the *CONET Integrated Robot-WSN Testbed.*

1 Introduction

SLAM (Simultaneous Localization and Mapping) is a fundamental problem in robotics that has been researched for many years. It consists in a robot placed at an unknown environment which generates a map of this environment and at the same time it localizes itself on this map. Finding solutions to this problem has been an important goal for robotics researchers. It is said that SLAM will make possible to build truly autonomous robots. *Range Only* SLAM methods rely on sensors that provide range measurements between the robot and a set of landmarks, assumed static. A variety of *Range Only* (*RO*-)SLAM methods have been developed, using EKF SLAM like [6] and [2] and FastSLAM [5]. In SLAM partial observability has been solved using tools for the initialization of new nodes in the state vector such as probability grids [2,3], Particle Filters [4,6] or Sum of Gaussians [5].

In this chapter we propose a scalable SLAM method that uses range measurements sensed by Wireless Sensor Networks (WSN) nodes. In our work, WSN nodes are taken as landmarks. Scalability is a fundamental property of WSNs. Large numbers of inexpensive WSN nodes can be deployed in a scenario. However, existing *RO*-SLAM methods have been validated assuming a very low number of nodes. Many existing

* This work was supported by the EC-SAFEMOBIL (European Commission ICT-2011-288082), CLEAR (DPI2011-28937-C02-01) and the Ministerio de Educación y Deportes FPU program.

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014,*
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_2,

RO-SLAM methods disregard scalability and do not scale well, which is a significant contradiction since the higher number of measurements integrated in the SLAM filter, the higher accuracies can be obtained. On the other hand, traditional SLAM methods use only direct robot-node range measurements disregarding the fact that nodes can belong to a network and can communicate with other nodes to provide internode range measurements that can be of interest for improving SLAM estimations.

The proposed method addresses SLAM for WSN combining two mechanisms. First, it integrates not only direct robot-node range measurements but also range measurements between static nodes, referred to as internode measurements. Integrating internode measurements can significantly reduce the uncertainty in the map estimation, which indirectly also improves the estimation of the robot pose. Integrating a high amount of measurements require tools to specifically deal with scalability. Second, the proposed method is based on Extended Information Filter (EIF) in contrast to the most widely used approach based on Extended Kalman Filter (EKF). Dual of Kalman Filters, the update stage of Information Filters is more efficient than that of Kalman Filters. That makes them more computationally efficient for cases with a simple prediction model and high number of measurements, as in our problem, in which large numbers of inexpensive WSN nodes can be deployed in the scenario.

This chapter describes the use of EIF for *RO*-SLAM, presents how internode measurements can be integrated in EIF SLAM and gives implementation and integration details. The proposed method has been validated in simulations and also in real experiments performed in the *CONET Robot-WSN Integrated Testbed* (`https://conet.us.es`) [9].

This chapter is structured as follows. A review of the related work and the objectives of this chapter are presented in Section 2. The proposed EIF *RO*-SLAM method is described in Section 3. Section 4 presents how internode measurements are integrated in the proposed SLAM filter. Section 5 validates each of the mechanisms of the proposed method in simulation and real experiments: use of internode measurements and EIF instead of EKF for saving computational burden. Conclusions and future research close the chapter.

2 Motivation

2.1 Related Work

SLAM using only measurements from range sensors (*RO*-SLAM) has been an objective of many researchers in last ten years. Range measurements have the problem of partial observability, i.e. several range measurements are necessary to disambiguate a location. In *RO*-SLAM partial observability is addressed by adopting mainly two different approaches: delayed and undelayed mapping initialization. In delayed initialization, new beacon nodes are introduced in the SLAM state vector when an external element has an initial estimation of its pose. Examples of tools for delayed initialization are Particle Filters [6] and Probability Grids [2,3], among others. On the other hand, undelayed initialization involves introducing a new beacon node directly in the state vector without having any previous information. This can be done with tools based on Sum of Gaussians [5], which adds to the state vector the mean of M Gaussians, each representing one

different hypothesis of the node position. When new measurements arrive, the weight of each Gaussian is updated. Gaussians with low weights are removed –the hypotheses are considered bad and are discarded– until only one survives the pruning (hopefully the good one).

Various SLAM filters have been developed in *RO*-SLAM. Two of them, EKF and RBPF (Rao-Blackwellised Particle Filter), are the most widely applied. EKF SLAM is a well known method that obtains good results in most implementations. Among its main advantages, it is optimal in presence of Gaussian noise. FastSLAM, based on RBPF, has a different approach. It factorizes the state vector dividing it in the vehicle pose estimation and the map estimation. It is more flexible than EKF due to the use of a Particle Filter as the core of the algorithm, which allows having different noise distributions apart from only Gaussian as in the case of Kalman Filters. In FastSLAM each particle of the filter represents an hypothesis of the robot pose and the map.

These methods have been validated with different sensors but the experiments performed always have small maps with a number of nodes between three and six. Even in simulation results, the number of nodes used is usually lower than ten. This low number of nodes allows the validation and accuracy analysis of the algorithms but disregards two critical aspects of its use in realistic applications: efficiency and scalability.

Also, all the above methods used only direct measurements between the robot and the nodes, ignoring the fact that most commercial off-the-shelf (COTS) nodes have networking capability and can measure their distance to other nodes. Very few methods except for [2] make use of internode measurements. Djugash et al. proposed different ways for incorporating internode measurements in *RO*-SLAM mainly by improving map estimation using virtual nodes and adopting off-line map improvement using multidimensional scaling (MDS). MDS with internode measurements was also used in [8]. However, these methods are executed off-line and not suitable for most applications, which require on-line robot and map estimations.

This chapter presents a method that intends to enable the use of *RO*-SLAM methods in WSN applications. The combination of two main mechanisms difference this method from existing ones. The first one is the adoption of an EIF in order to improve scalability. The second is the integration of internode measurements exploiting the fact that nodes belong to a larger WSN network. Using internode measurements significantly reduces map uncertainty and also improves robot localization accuracy.

2.2 Objectives

Consider a sensor network which nodes have been deployed at random locations in a GPS-denied environment. For instance, this is the case of a set of sensor nodes that have been deployed –thrown– by an Unmanned Aerial Vehicle (UAV) for real-time monitoring of the status of a disaster or accident in an urban or industry area, where the buildings prevent correct GPS reception. The nodes are equipped with range sensors and also with suitable sensors for monitoring the event, e.g. toxic gas concentration sensors in a pollution episode. Assume the nodes can measure the distance to the robot or to other nodes. They also have communication capabilities and computing capacity sufficient for executing simple communication protocols and simple calculations such as filtering.

The proposed method takes advantage of the capability of WSN nodes for interchanging messages that allow measuring range between every pair of nodes. Integrating internode measurements in SLAM allows considering information that was disregarded in traditional methods based only on direct robot-node range measurements. Integrating internode measurements can significantly reduce the uncertainty in the map estimation, which indirectly improves estimation also of the robot pose. However, using internode measurements in SLAM involves solving scalability problems. N static nodes can provide $Nx(N-1)$ single-hop internode measurements. This could overload traditional schemes based on Extended Kalman Filters. Below is an analysis that justifies the adoption of EIF instead of EKF to improve scalability.

Consider a 2D scenario where N WSN nodes have been deployed. Assume that a SLAM state vector of size n has been defined. The SLAM state vector should contain the coordinates of the N nodes and the robot state estimation. The complexity of the prediction step of EKF is $O(n^2)$ and that of the correction step is $O(k^{2.8}+n^2)$ [1], being k the number of measurements gathered at one time step. The EIF complexity is $O(n^{2.8})$ for the prediction step and $O(n^2)$ for the correction step [1].

First, we assume that the robot takes one measurement from each node at each time. The number of measurements gathered at one time step is $k \approx \frac{n}{2}$. Thus, the complexity of the EKF will be proportional to $O(n^{2.8})$ and that of the EIF, $O(n^{2.8})$. We assume internode measurements and that each node can perform range measurements with any other node. Thus, the total number of measurements is $k \approx \frac{n}{2} + \left(\frac{n}{2}\right)^2$. In this case the complexity of the EKF becomes $O(n^{5.6})$ while the complexity of the EIF keeps being $O(n^{2.8})$. Thus, EIF is significantly more efficient and scalable than traditional EKFs for solving SLAM with internode measurements. On the other hand, the memory use is similar in EIFs and EKFs. Both require similar memory for representing the system state and intermediate data and matrices.

Section 3 describes the EIF SLAM and Section 4 presents how internode measurements are integrated in the EIF SLAM filter.

3 EIF SLAM

3.1 Extended Information Filter

The Extended Information Filter SLAM, or EIF SLAM, is an alternative to the typical EKF SLAM representation. Both algorithms represent the posterior probability density function of a state vector by a Gaussian distribution, but EIF SLAM represents this posterior in its information form, i.e. using the *information matrix* Ω and the *information state vector* (or *information vector*) ξ. This is known as the canonical representation, and is related to the moments representation (mean μ and covariance Σ) by the following equations:

$$\Omega = \Sigma^{-1} \tag{1}$$

$$\xi = \Sigma^{-1}\mu \tag{2}$$

Our method is an on-line version of the traditional EIF SLAM, i.e. it provides on-line map and robot location estimations. The SLAM state vector x_t consists of the vehicle pose and orientation (x, y, θ) and the locations (x_i, y_i) of each of the N nodes in the map:

$$x_t = [x\ y\ \theta\ x_1\ y_1 \ldots x_N\ y_N]^T \tag{3}$$

The proposed on-line EIF SLAM algorithm is described in Algorithms 1 and 2. EIF, like other Bayesian filters, is applied in two main stages: *prediction* and *correction*. The prediction stage is carried out by the *motion update* function (see Algorithm 1) and the correction stage, by the *measurement update* function (see Algorithm 2). Moreover, the SLAM algorithm must implement the PFs, which are presented in Section 3.2.

The operation of EIF is a prediction-correction loop, like EKF. But here, because of the use of the information form, there is a third step in the loop that recovers μ_t, the estimate of the state vector, using the expression:

$$\mu_t = \Omega_t^{-1} \xi_t \tag{4}$$

Thus, it is necessary to perform the following steps:

1. Linearize the motion model, i.e. calculate de Jacobian of the vehicle kinematic model (6) in the prediction step.
2. Calculate and linearize the observation model (7)(8) in the correction step.
3. Update the Particle Filters.

The prediction stage includes the kinematic model of the robot used in SLAM. In our case we are assuming a mobile robot with differential motion configuration. Its kinematic model is the following:

$$\mu_t = f(\mu_{t-1}, u_t) = \begin{bmatrix} \mu_{t-1,x} + \Delta t v_t \cos(\mu_{t-1,\theta}) \\ \mu_{t-1,y} + \Delta t v_t \sin(\mu_{t-1,\theta}) \\ \mu_{t-1,\theta} + \Delta t \alpha_t \end{bmatrix}, \tag{5}$$

where $\mu_{t-1} = [\mu_{t-1,x}, \mu_{t-1,y}, \mu_{t-1,\theta}]^T$ is the estimate of the robot's position and orientation at time $t-1$, Δt is the sample time, and $u_t = [v_t, \alpha_t]^T$. v_t and α_t are the linear and steering velocities of the robot at time t.

The linearization of the kinematic model f is the matrix A, Jacobian of f:

$$A = \frac{\partial f}{\partial \mu_{t-1}} = \begin{bmatrix} 1 & 0 & -\Delta t v_t \sin(\mu_{t-1,\theta}) \\ 0 & 1 & \Delta t v_t \cos(\mu_{t-1,\theta}) \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The prediction step of the EIF is detailed in Algorithm 1. Lines 2 and 3 present the prediction model, robot's kinematic model (5) and its Jacobian (6), respectively. Line 10 is the prediction of the state like in an EKF. The projection matrix F_x defined in line 1 reflects the fact that nodes are static and only the robot's states change in this step. Thus, the predicted state of the map is just its previous estimation. Lines 4 to 9 updates the values of the information matrix Ω and the information vector ξ.

Algorithm 1. Motion update in EIF SLAM

Require: $\xi_{t-1}, \Omega_{t-1}, \mu_{t-1}, u_t$

1. $F_x = \begin{pmatrix} 1\,0\,0\,0\cdots 0 \\ 0\,1\,0\,0\cdots 0 \\ 0\,0\,1\,0\cdots 0 \end{pmatrix}$
2. $\delta_t = \begin{pmatrix} \Delta t v_t \cos(\mu_{t-1,\theta}) \\ \Delta t v_t \sin(\mu_{t-1,\theta}) \\ \Delta t \alpha_t \end{pmatrix}$
3. $A_t = \begin{pmatrix} 1\,0 & -\Delta t v_t \sin(\mu_{t-1,\theta}) \\ 0\,1 & \Delta t v_t \cos(\mu_{t-1,\theta}) \\ 0\,0 & 1 \end{pmatrix}$
4. $\Psi_t = F_x^T [A_t^{-1} - I] F_x$
5. $\lambda_t = \Psi_t^T \Omega_{t-1} + \Omega_{t-1} \Psi_t + \Psi_t^T \Omega_{t-1} \Psi_t$
6. $\Phi_t = \Omega_{t-1} + \lambda_t$
7. $\kappa_t = \Phi_t F_x^T (Q_t^{-1} + F_x \Phi_t F_x^T)^{-1} F_x \Phi_t$
8. $\bar{\Omega}_t = \Phi_t - \kappa_t$
9. $\bar{\xi}_t = \xi_{t-1} + (\lambda_t - \kappa_t)\mu_{t-1} + \bar{\Omega}_t F_x^T \delta_t$
10. $\bar{\mu}_t = \mu_{t-1} + F_x^T \delta_t$
11. **return** $\bar{\xi}_t, \bar{\Omega}_t, \bar{\mu}_t$

The measurements gathered are range measurements. Thus, the observation model adopted is the distance between the robot and the observed node:

$$h_i(\mu_t) = r_t^i = \sqrt{\delta_x^2 + \delta_y^2}, \tag{7}$$

being $\delta_x = \mu_{t,x}^i - \mu_{t,x}$ and $\delta_y = \mu_{t,y}^i - \mu_{t,y}$, where $(\mu_{t,x}, \mu_{t,y})$ is the estimate of robot position at time t and $(\mu_{t,x}^i, \mu_{t,y}^i)$, the estimate of the location of node i at time t.

Function h is nonlinear and should be linearized using its Jacobian H:

$$H_i = \frac{\partial h_i}{\partial \mu_t} = \left[\frac{\delta_x}{r_t^i} \; \frac{\delta_y}{r_t^i} \; 0 \cdots \frac{-\delta_x}{r_t^i} \; \frac{-\delta_y}{r_t^i} \cdots \right] \tag{8}$$

The correction step of the EIF is implemented by Algorithm 2. It is responsible for the processing of measurements. R_t, defined in line 1, is the covariance matrix of the measurement noise. Lines from 2 to 6 calculate the predicted observation $h_i(\mu_t)$ for node i and its Jacobian H_i. Lines 7 and 8 represent the addition of the information provided by the new measurements to the information matrix Ω_t and the information vector ξ_t.

3.2 Particle Filter

The Particle Filter (PF) is an alternative non parametric implementation of the Bayes filter. The key of the PF is to represent the posterior as a set of random state samples of the original distribution. This representation is an approximation, more accurate with higher number of samples, but it is capable of expressing any probability distribution, not only a Gaussian. This flexibility, together with its multi-hypothesis capability, enables its use in *RO*-SLAM to solve the partial observation problem.

Algorithm 2. Measurement update in EIF SLAM

Require: $\bar{\xi}_t, \bar{\Omega}_t, \mu_t, z_t$
1. $R_t = \sigma_r^2$
2. **for all** measurements z_t^i **do**
3. $\quad \delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \mu_{t,x}^i - \mu_{t,x} \\ \mu_{t,y}^i - \mu_{t,y} \end{pmatrix}$
4. $\quad h_i(\mu_t) = r_t^i = \sqrt{\delta_x^2 + \delta_y^2}$
5. $\quad H_i = \left[\frac{\delta_x}{r_t^i} \; \frac{\delta_y}{r_t^i} \; 0 \cdots \; \frac{-\delta_x}{r_t^i} \; \frac{-\delta_y}{r_t^i} \; \cdots \right]$
6. **end for**
7. $\xi_t = \bar{\xi}_t + \sum_i H_i^T R_t^{-1} [z_t^i - h_i(\mu_t) + H_i \mu_t]$
8. $\Omega_t = \bar{\Omega}_t + \sum_i H_i^T R_t^{-1} H_i$
9. **return** ξ_t, Ω_t

In the PF, the samples are called particles and are denoted as:

$$\mathscr{X}_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{9}$$

Each particle $x_t^{[m]}$ (with $1 \leq m \leq M$) is a sample of the state at the time instant t, that is, an hypothesis of the state vector at time t. Here M denotes the number of particles in the set $\mathscr{X}_t$. In practice, the number of particles M is usually high (e.g. $M = 500$), although in some implementations M is adapted dynamically.

Like other Bayes filters, PF performs a recursive estimation based on the last estimation made at the previous time instant. Thus, PF builds a new set of particles $\mathscr{X}_t$ recursively based on the set $\mathscr{X}_{t-1}$. It has three main steps: motion estimation, importance sampling and importance resampling.

The motion estimation step is like an EKF prediction step: it generates a hypothetic state $x_t^{[m]}$ based on the particle's previous state $x_{t-1}^{[m]}$ and control action u_t. Importance sampling, or sampling, updates the weight $w_t^{[m]}$ of each particle. This weight (also called *importance factor*) is used to incorporate a new measurement z_t in the particle set. $w_t^{[m]}$ represents the probability that the measurement z_t corresponds to the state of the particle $x_t^{[m]}$, i.e. $w_t^{[m]} = p(z_t | x_t^{[m]})$. The set of particles with their corresponding weights represent the estimation of the state posterior.

The importance resampling draws with replacement M particles of the set $\mathscr{X}_t$. The probability for a particle of being in the new set is its own weight. The new set will have the same size, thus there will be particles that will not appear in this new set (those with lower weight) and there will be duplicated particles (those with higher weight). Importance resampling has the function of replacing the less probable particles with the more probable ones. Without this step the convergence to a unique solution would be slower and less accurate.

When the first measurement of one node is gathered, a new Particle Filter is deployed. The particles are displayed uniformly into a ring centered in the actual robot pose, with radius z_t and width σ_m. The value of σ_m will depend on the measurement noise. Figure 1 shows an example in which particles are deployed around the robot

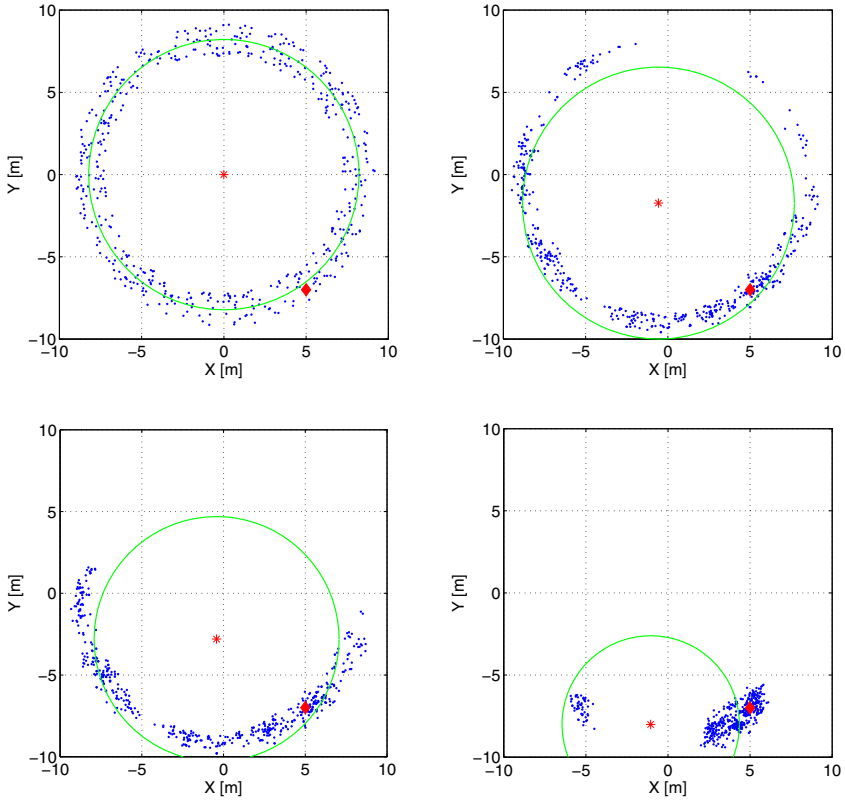


Fig. 1. Distribution of particles along PF evolution

location (the asterisk in the center) and evolve until they converge. The diamond represents the ground truth location of the node.

While the robot moves around and gathers new measurements of different nodes, the PFs will evolve until each one converges to a unique Gaussian hypothesis. Then, the converged filter is removed and a new state initialized with the node location estimated by the Particle Filter is added to the EIF state vector.

4 EIF SLAM with Internode Measurements

The proposed method takes advantage of the capability of WSN nodes of interchanging messages that allow measuring range between every pair of nodes. Thus, the on-line EIF scheme integrates not only direct robot-node range measurements but also measurements between static nodes. Integrating internode measurements is expected to provide the following improvements:

- Faster rate of convergence of the Particle Filters, involving lower computer burden.

- Higher accuracy in the estimation obtained by Particle Filters, which directly improves map estimation.
- Indirectly, higher accuracy in the estimation of the robot's pose.

It should be noticed that it is straightforward to design protocols in WSN for gathering internode range measurements. The robot periodically broadcasts a measurement request packet. The nodes within the robot's sensing range, we call them *nodes A*, take a measurement and save it in a table. Then, each *node A* broadcasts a measurement request packet. The nodes receiving this request take a measurement and send it back to the requesting *node A* in a reply packet. Then, *nodes A* receive all the reply packets, make new range measurements and save them in their table. Next, they compute the mean value of measurements in the table and transmit them to the robot in a reply packet. Now, the robot can integrate these measurements in the SLAM algorithm.

Algorithm 2 shows the steps to integrate direct robot-node measurements. If the robot receives an internode measurement between two static nodes i and j the algorithm performs differently depending on the current state of the nodes involved in the measurement:

1. If both nodes are in the state vector –their PFs have previously converged, then the algorithm executes function *measurement update* (Algorithm 2), modifying the observation model and its Jacobian by the following expressions, being r_t^{ij} the distance between the estimation of the location of nodes i and j:

$$h_{ij} = r_t^{ij} \tag{10}$$

$$H_{ij} = \begin{bmatrix} 0\,0\,0 \cdots \frac{\delta_x}{r_t^{ij}} \; \frac{\delta_y}{r_t^{ij}} \cdots \frac{-\delta_x}{r_t^{ij}} \; \frac{-\delta_y}{r_t^{ij}} \cdots \end{bmatrix} \tag{11}$$

2. If only one of the nodes (e.g. i) is in the state vector but the PF of the other node has not converged yet, then the measurement is used to update this PF.
3. If only one of the nodes (e.g. i) is in the state vector but this is the first measurement of the other (j), a new PF is deployed with the ring centered at node i.
4. If none of the nodes are in the state vector the measurement is stored for future use.

5 Simulation and Experimental Results

5.1 Simulation Results

The simulation scenario is based on Djugash Plaza dataset [10], but replacing the original nodes –which were only four– with a new set of 43 nodes scattered in this scenario. Each node is assumed to be equipped with a range sensor, such as the *Nanotron NanoPAN* sensors used in the experiments. These sensors use TOF (Time Of Flight) ranging technology. In preliminary indoor tests their measurement error was measured experimentally to be Gaussian with an error variance is $\sigma_m^2 = 1m$, as can be seen in Fig. 2. The PFs were configured with 500 particles.

Most SLAM methods provide the map and robot location in a local coordinate frame, i.e. a local map. For comparison with the ground truth, the map and robot's path estimated by the SLAM method was submitted to an affine transform that re-aligns the

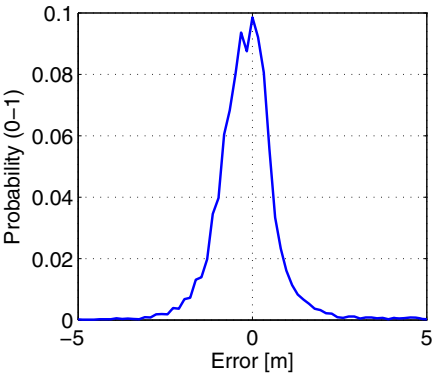


Fig. 2. Result of the error in range measurements of the *Nanotron NanoPAN* sensor nodes taken from different positions in preliminary indoors tests carried out in the *CONET Integrated Testbed*

local solution into the global coordinate frame. The figures presented in this section use cyan color to represent the ground-truth of the map and the robot's path and red color to represent the estimations of the SLAM filter.

This section evaluates and compares the performance of the two improvements adopted with respect to the baseline EKF approach: a) the use of internode measurements and b) the use of EIF instead of EKF. Figure 3 shows the result of a simulation of the EKF SLAM using only direct robot-node measurements. Figure 4 shows the result in case of using EKF with internode measurements. The rest of the algorithm and parameters were exactly the same. It can be seen that using internode measurements improves the map accuracy. Also, the robot's path estimation has lower error.

Figure 5 shows the cumulate errors in robot's location with respect to the ground-truth in both cases. Internode measurements have direct impact on improving map estimation accuracy and indirectly increases the accuracy of robot location estimation.

The next step after integrating internode measurements is comparing the SLAM filter with EKF and that with the proposed EIF method. Figure 6 shows the result of the EIF method using internode measurements. This method and the EKF performed successfully and have low errors with respect to the ground-truth.

Figure 7 compares the cumulate error in robot location resulting from the EIF and the EKF both with internode measurements. Error distribution was very similar in both cases. However, efficiency and scalability of the proposed EIF is significantly better as analyzed in Section 2.

It was pointed out in Section 4 that using internode measurements can help to increase the rate of convergence and accuracy of Particle Filters. Figure 8 shows the times in which the PF of each node was deployed (red) and converged (blue) in an experiment with EIF with internode measurements (bottom) and with an EKF without them (top). In this figure it is easy to notice that integrating internode measurements greatly reduces the PF convergence times and it also has impact on PF deployment times. Using internode measurements the robot can receive and integrate measurements from nodes that are beyond the robot sensing range.

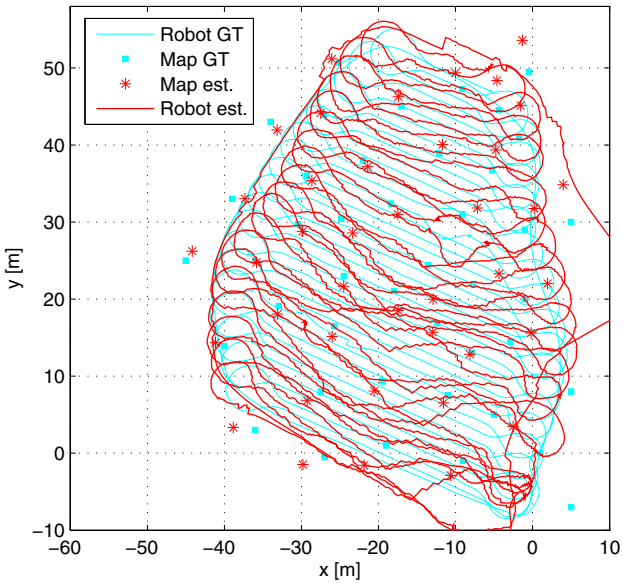


Fig. 3. Result of the EKF SLAM algorithm using only direct robot-node measurements

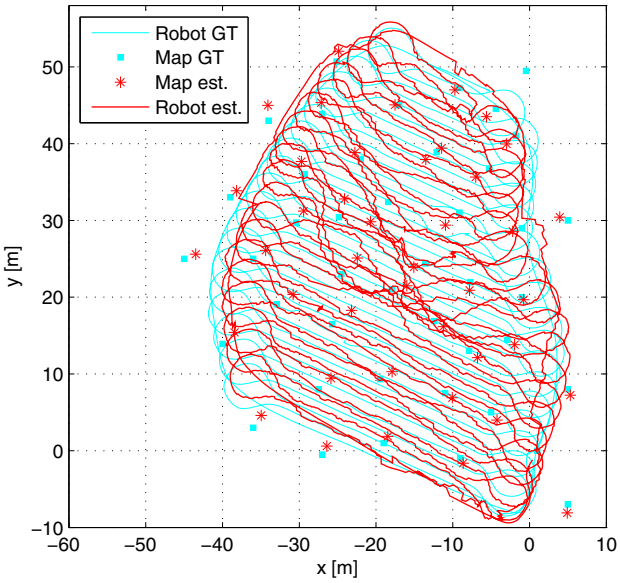


Fig. 4. Result of the EKF SLAM algorithm using robot-node and internode measurements

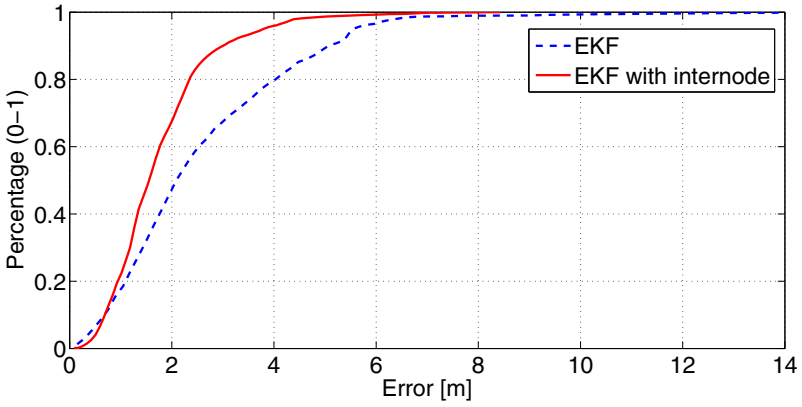


Fig. 5. Cumulate robot location error provided by the EKF SLAM with and without internode measurements

The computing times of both methods were also analysed. In this example, with low map size and low number of nodes, the EKF SLAM with internode measurements required an average of 64.3 ms/iteration while the EKF with internode required 44.8 ms/iteration (a reduction of 31%). These numbers were measured with the MATLAB Profiler tool. This advantage was measured to be more noticeable as the map size increases, confirming the scalability analysis performed in Section 2.2.

5.2 Experimental Results

Real experiments have been carried out in the *CONET Robot-WSN Integrated Testbed*, see Fig. 9. The testbed (`http://conet.us.es`) is a remote open tool to assess and compare multi-robot and WSN methods and algorithms [9]. It is comprised of 5 Pioneer AT robots and 140 WSN nodes –static and mobile– of different models (TelosB, MicaZ, Iris and Mica2) and a network of 6 *Nanotron NanoPAN* nodes. Each robot is equipped with a PC for processing all the data and controlling the robot, a Hokuyo 2D laser range sensor, a Microsoft Kinect camera, GPS and Inertial Measurement Unit, among others. The testbed uses an open and modular architecture and is installed since 2010 at the basement of the building of the School of Engineering of Seville (Spain).

Several experiments have been performed in the *CONET Integrated Testbed* in order to obtain real datasets. In these experiments the mobile robot was one of the Pioneer 3-AT robots, and the nodes deployed were equipped with *Nanotron NanoPAN* modules. Recall from Section 5.1, the measurement error was measured to be Gaussian with $\sigma_m^2 = 1m$. All the available *Nanotron NanoPAN* nodes were used in the experiment. The ground-truth data is collected by a Player[1] module called AMCL (Adaptive Monte Carlo Localization), which integrates the Hokuyo 2D laser measurements and a map of the testbed into a particle filter to give an accurate estimation of the robot's real location [11].

[1] `http://playerstage.sourceforge.net/`

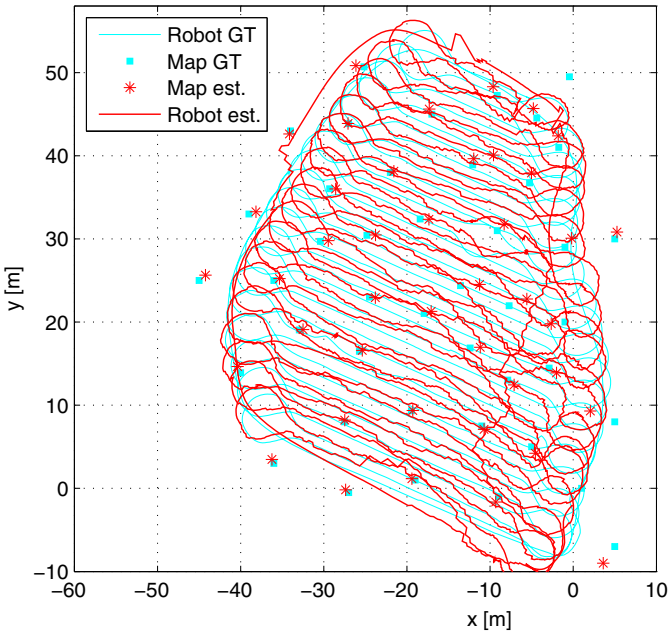


Fig. 6. Result of the proposed EIF SLAM algorithm with internode measurements

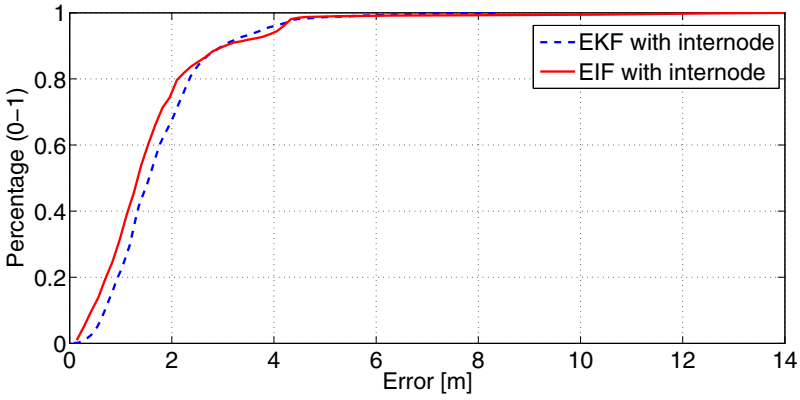


Fig. 7. Cumulate robot location error provided by the EKF SLAM and EIF SLAM both with internode measurements

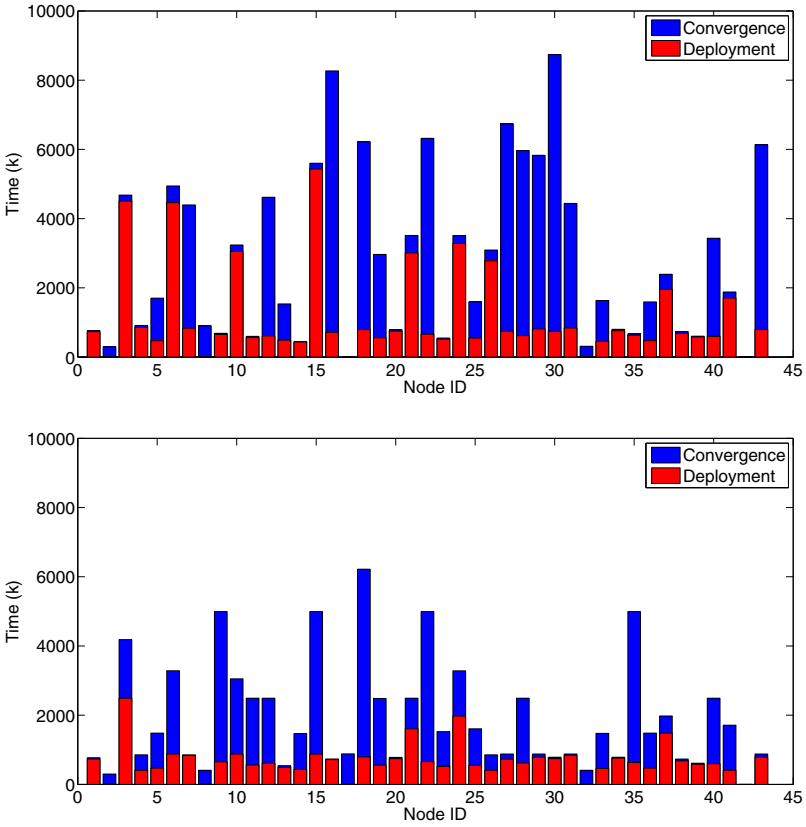


Fig. 8. Deployment and convergence times of Particle Filters of every node in the scenario: (Top) EKF without internode measurements, (Bottom) EIF with internode measurements

Fig. 9. Picture taken in the validation experiments carried out in the *CONET Integrated Testbed*

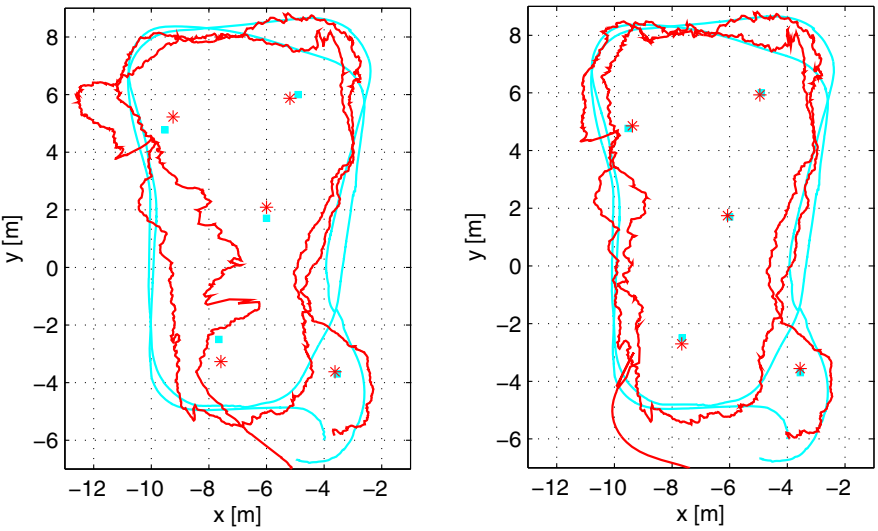


Fig. 10. Results with real experiments. Red color corresponds to estimations while the ground truth is in cyan. (Left) EIF without internode measurements. (Right) EIF with internode measurements.

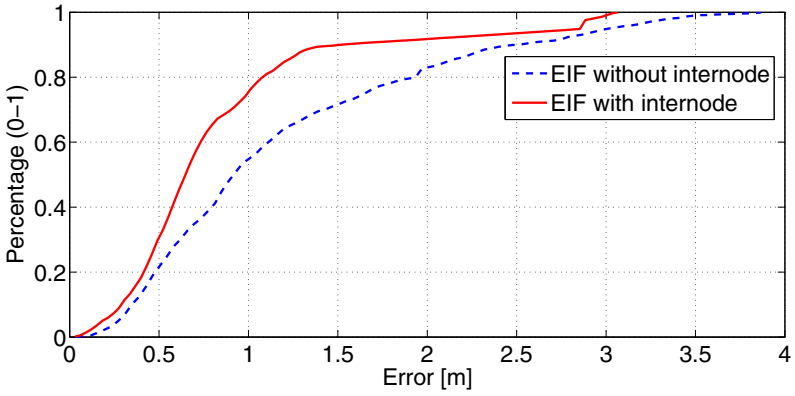


Fig. 11. Cumulate robot location error provided by the proposed EIF SLAM with and without internode measurements

Figure 10 shows the results of executing the proposed EIF SLAM algorithm with and without internode measurements. It can be checked that errors with respect to the ground-truth are lower in the proposed method that integrates internode measurements.

Figure 11 compares the cumulate error in estimation of robot's position resulting from the proposed EIF without internode measurements and the EIF with them. It can

be seen how internode measurements improve robot accuracy in an average of 25% in these experiments.

These experiments have validated the proposed method and the advantages of using internode measurements. The PFs converged faster, the map error is reduced in a 50% and the robot error, in a 25%.

6 Conclusions

This chapter described an EIF SLAM algorithm suitable for operation with WSN. First, it exploits WSN nodes capability of organizing into networks and communicating with other nodes. Second, to cope with the high number of measurements, the method adopts an EIF SLAM filter, significantly more scalable and efficient than traditional schemes based on EKF.

The integration and use of internode measurements can significantly improve map and robot estimations accuracy. It can also anticipate the deployment and convergence of the Particle Filters, resulting in lower computational burden. The main disadvantage is the increase in the number of measurements. The update stage of Extended Information Filters is more efficient than that of Extended Kalman Filters. That makes EIF more computationally efficient for cases with a simple prediction model and high number of measurements, as in our problem. The proposed EIF method has been extensively compared with traditional schemes based on EKF both in simulation and in experiments carried out in the *CONET Integrated Robot-WSN Testbed*.

Range-Only SLAM and range-only localization methods in general are very interesting for WSN. The proposed method intends to contribute to the application of SLAM techniques, developed mainly under a robotics perspective, for WSN problems and applications.

One limitation of the proposed method is that it cannot integrate internode measurements until at least one PF has converged. The adoption of these robot-WSN collaboration in undelayed SLAM approaches such as based on Gaussian mixtures is under current research. This work proposed a mechanism for exploiting robot-WSN collaboration synergies in on-line SLAM integrating internode measurements. The extension to other efficient such as RBPF is being analysed. The validation of the scalability advantages of the proposed scheme in large experimental settings is object of current work.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics, Intelligent Robotics and Autonomous Agents, 3rd edn. The MIT Press, Cambridge (2005)
2. Djugash, J., Singh, S., Kantor, G., Zhang, W.: Range-only SLAM for robots operating cooperatively with sensor networks. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, pp. 2078–2084 (2006)
3. Olson, E., Leonard, J., Teller, S.: Robust range-only beacon localization. In: Autonomous Underwater Vehicles, 2004 IEEE/OES, June 17-18, pp. 66–75 (2004)
4. Blanco, J.-L., Gonzalez, J., Fernandez-Madrigal, J.-A.: A pure probabilistic approach to range-only SLAM. In: IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, pp. 1436–1441 (2008)

5. Blanco, J.-L., Fernandez-Madrigal, J.-A., Gonzalez, J.: Efficient probabilistic Range-Only SLAM. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, September 22-26, pp. 1017–1022 (2008)
6. Menegatti, E., Danieletto, M., Mina, M., Pretto, A., Bardella, A., Zanconato, S., Zanuttigh, P., Zanella, A.: Autonomous discovery, localization and recognition of smart objects through WSN and image features. In: 2010 IEEE GLOBECOM Workshops (GC Wkshps), December 6-10, pp. 1653–1657 (2010)
7. Caballero, F., Merino, L., Ollero, A.: A general Gaussian-mixture approach for range-only mapping using multiple hypotheses. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), May 3-7, pp. 4404–4409 (2010)
8. Bardella, A., Danieletto, M., Menegatti, E., Zanella, A., Pretto, A., Zanuttigh, P.: Autonomous robot exploration in smart environments exploiting wireless sensors and visual features. Annals of Telecommunications-Annales des Tlcommunications 67(7-8), 297–311 (2012)
9. Jiménez-González, A., Martinez-de Dios, J.R., Ollero, A.: An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. Sensors 11(12), 11516–11543 (2011)
10. Djugash, J., Hamner, B., Roth, S.: Navigating with ranging radios: Five data sets with ground truth. Journal of Field Robotics 26(9), 689–695 (2009)
11. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo localization for mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1322–1328 (1999)

Mobility-Assisted Localization Techniques in Wireless Sensor Networks: Issues, Challenges and Approaches

Subir Halder and Amrita Ghosal

Department of Computer Science and Engineering,
Dr. B.C. Roy Engineering College, Durgapur, India

Abstract. Many network operations and applications of wireless sensor networks (WSNs) need sensor nodes for obtaining their locations. Sensor nodes equipped with geographical positioning system (GPS) devices are aware of their locations at a precision level of few meters. However, installing GPS devices on a large number of sensor nodes is not only costly but affects the form factor of these sensor nodes. Moreover, GPS-based localization is not applicable in indoor environments such as buildings. There exists an extensive amount of research literature that aims at obtaining absolute locations as well as relative spatial locations of sensor nodes in a wireless sensor network without requiring specialized hardware at large scale. The typical approach that significantly reduces the cost is replacing the large set of statically deployed GPS-enhanced sensor nodes with limited number of mobile anchors. These mobile anchors are aware of their own locations and move in order to cover the entire network, and then try to infer locations of sensor nodes using various techniques e.g. geometric, statistical etc. Thus, keeping this in mind the chapter presents key issues and inherent challenges faced by the mobility-assisted localization techniques in WSNs. Also, we take a close look at the algorithmic approaches of various important fine-grained mobility-assisted localization techniques applicable for low power, resource constrained and highly distributed sensor nodes.

1 Introduction

A wireless sensor network (WSN) consists of a large number of energy-constrained, low-cost and low-power sensor nodes. Each sensor node is a device, equipped with multiple on-board sensing elements, wireless transmitter–receiver modules, computational and power supply elements and is characterized by limited computational and communication capabilities. In WSNs, identifying locations of sensor nodes is important for both network operations and most application level tasks because sensory data without spatial and temporal coordination is of very limited use [1, 2]. Determining the physical location of the sensor nodes after they have been deployed is known as localization [3]. Knowledge of location information i.e., localization enables in implementing efficient routing (such as geographic forwarding) [4-6], target tracking [7, 8] etc. Successful localization techniques are useful in many applications few of which are elaborated below-

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*,
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_3,

- In events such as disaster relief, forest fire, failure of structure etc., early location prediction helps in planning adequate response system that may either prevent those events from occurring or mitigate the consequential damages. The efficient response system is dependent on the accurate location information. Therefore, accurate localization scheme based response system may prevent such calamities.
- Navigation and vehicle tracking is another area where accurate location estimation scheme using WSNs may be extremely useful. Vehicle tracking with autonomous interception mechanism can be deployed in an outdoor area. It senses entry as well as movement of an offending evader in the area. A cooperative mobile agent may be dispatched for intercepting the evader as soon as it gets detected before any damage is done. The successful realization of such a tracking and interception system is dependent on accurate location information.
- The topology in WSNs is highly dynamic in nature. This is because some sensor nodes die if they drain out their energy faster compared to other sensor nodes leading to coverage and connectivity disruptions in the network. In such scenarios in order to reestablish coverage and connectivity, new sensor nodes may be injected into the network. Here geographic routings are found to be more efficient than topology based routing schemes. The basic issue that should be addressed in a geographic routing scheme is its ability to gather location information and to have a location tracking mechanism for establishing connectivity before routing data. So, localization or finding locations of sensor nodes is a fundamental step in routing or transmission of data in WSNs.

Localization schemes can be characterized by a set of attribute pairs such as sensor nodes could be static or mobile, deployed indoor or outdoor, in a 2-D or a 3-D space. Location measurements may or may not require additional hardware. The use of additional hardware in a sensor node should be avoided, as it not only raises the cost, but increases both form factor and operational resource requirements. Another way to classify a localization process could be on-demand or periodic. A sensor node may take an active role in determining its locations, or it may wait for computation of location information by other devices. Furthermore, a localization process may be centralized or distributed according to the nature of the underlying algorithm. Finally, the objective of a localization procedure varies from absolute to relative locations. Relative locations, which involve virtual coordinates, are sufficient for certain applications. Depending on the application's necessities, either coarse- or fine-grain location information may also be adequate.

1.1 Motivation and Objective

A compendium of knowledge representing rich collection on localization issues in WSNs can be found in the recent past literatures [9-12]. They share the same main idea that sensor nodes with unknown coordinates are guided by one or more sensor nodes with known coordinates in order to estimate their positions. The sensor nodes with known coordinates are called anchor or beacon nodes. The existing localization schemes proposed for sensor nodes in WSNs are broadly categorized into five groups:

distributed algorithms, centralized algorithms, iterative algorithms, mobility-assisted approach, and statistical techniques. Each of these categories has its leverages and limitations. Out of these localization methods, in recent years, mobility-assisted localization has been actively pursued where mobile beacon or mobile anchor(s) move across the network while sending beacon packets or beacons around [13]. The mobility-assisted localization schemes relieves WSNs from the significant cost of deploying GPS receivers and the pressure of provisioning energy for interacting with each other during the localization process. Further, the use of mobile anchors has the advantage of reduced deployment cost (only a few mobile anchors are required), installation cost and more flexible deployment with one mobile anchor equivalent to many "virtual" anchors at specific positions. Furthermore, mobility-assisted localization algorithms show significant savings on energy consumption, communication overhead (only local communication is involved) and improves accuracy [14]. In this chapter, we provide key issues and inherent challenges faced by the mobility-assisted localization techniques in WSNs. We also attempt to review the existing state-of-the-art fine-grained mobility-assisted localization with emphasis on the algorithmic approaches.

1.2 Chapter Organization

This chapter has been organized as follows. In section 2, we briefly explain the generic approach, key issues and main challenges faced by the mobility-assisted localization techniques. Mobility-assisted localization approaches either use range-based techniques which depend on geometry of nearness exploiting relationships among spatial coordinates of sensor nodes and mobile anchors or use range-free techniques based on the geometric solutions. In Section 3, we study existing algorithmic aspects of each of such techniques. Planned trajectory of a mobile anchor is more beneficial than the random trajectory and it considerably affects the performance of localization techniques. Section 4, provides summary of such existing planned trajectory used in mobility-assisted localization techniques. Section 5 deals with performance issues while Section 6 talks about open issues in mobility-assisted sensor node localization. Finally, Section 7 concludes this chapter.

2 Issues and Challenges for Mobility-Assisted Localization

In this section, we first introduce the technical preliminaries of mobility-assisted localization techniques for WSNs. We mention key issues of location discovering, which are very essential aspects in any localization technique including mobility-assisted ones. Finally, the inherent challenges incurred by the mobility-assisted localization techniques are also discussed.

2.1 Background

In WSNs, a sensor node can determine whether it is in the radio range of an anchor according to a beacon signal received from the one-hop anchor. Considering this

capability, in mobility-assisted localization scheme, mobile anchor periodically broadcasts beacons containing its coordinates while traversing the network area where sensor nodes are deployed as shown in Fig. 1. If a sensor node receives beacon broadcast multiple times from different positions, it is more or less similar to receiving beacons from multiple anchors. Upon receiving the beacons, a sensor node determines its location relative to the mobile anchor according to the received signal strength (RSS) of the beacon through Bayesian inference. The accuracy of location estimation in mobility-assisted localization scheme depends on the distribution of anchor locations used by the mobile anchor. Main drawback associated with mobility-assisted localization scheme is significant localization delay. It is because sensor nodes can be localized only when they are in direct contact with the mobile anchors and receive sufficient signals from them.

In presence of several categories of mobility-assisted localization schemes, similar to [15], we broadly classify the existing mobility-assisted localization schemes into two classes- distance-based or range-based and connectivity-based or range-free. In range-based schemes, sensor nodes estimate their distances from mobile anchors using some specialized hardware. These measurements are used in methods like multilateration, which is based on the idea that a sensor node's location is uniquely specified when at least the distances or angles of three or more reference points are available for a sensor node. A special case of multilateration is trilateration [16]. Although the use of range measurements results in a fine-grained localization scheme, range-based algorithms require sensor nodes containing hardware for making range measurements. Range-free schemes do not use radio signal strengths, angle of arrival of signals or distance measurements and do not need any special hardware. Range-free algorithms require that each sensor node knows- (i) nodes that are within radio range, (ii) their location estimates and (iii) the (ideal) radio range of sensor nodes. No other information is used for localization. Thus, range-free schemes are more cost-effective as they do not require sensor nodes to be equipped with any special hardware. Also, it requires less information than range-based schemes. Range-free schemes can only provide a coarse grained estimate of each sensor node's location and are therefore suitable for applications that need approximate locations.

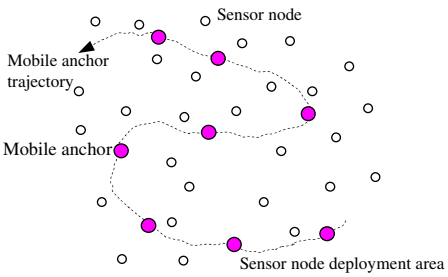


Fig. 1. Localization using a mobile anchor

2.2 Key Issues

The particular requirements for localization schemes for sensor networks generally depend on the nature of the applications, constraints imposed by hardware and network infrastructure. Based on these, some of the specific issues concerning the design of the mobility-assisted localization scheme are as follows-

Accuracy and Precision of Localization: The most important metrics for localization techniques are accuracy and precision. For a localization system, accuracy and precision depend on a number of factors. Every system has its own granularity of measurements, which describes the smallest measurable distance. Depending on the equipment and technique used, localization schemes may have a granularity of measurements within few inches or even bigger. Similar to the previous, the required granularity of localization in sensor networks also depends on the application.

Absolute versus Relative Locations: Localization systems consisting of GPS devices determine the absolute location in terms of the latitude, longitude and altitude with respect to the earth's coordinates. On the other hand, locations may be obtained with respect to a given frame of reference, such as the location of a mobile anchor. Based on the application requirement, locations can be either absolute or relative. It is noted that a relative location can always be transformed to an absolute location if the absolute location of the reference point is known.

Communication Requirements: Communication between a sensor node and a mobile anchor or other sensor nodes can provide significant benefits such as time synchronization and improvements in accuracy and precision. However, a fundamental issue in WSNs is the minimization of communication requirements in the sensor nodes to conserve energy. This introduces unique considerations for designing the localization scheme as well.

Cost: It is an important issue, as the requirements for designing large scale sensor networks are (a) to keep the cost of each sensor node low and (b) taking advantage from the collective sensing and computation capabilities of a large number of sensor nodes in the network. Thus, it is extremely desirable that the localization system does not require expensive hardware at the sensor nodes. The cost of building external infrastructure for enabling localization also plays a role. However, that is usually considered less critical because it does not increase with the size of the network.

2.3 Inherent Challenges

Localization plays a significant role in many applications, few of which are briefed in section 1. However, mobility-assisted localization itself is a complex problem to be solved, because of the demanding requirements for low cost, high energy efficiency, and scalability for any network size, as well as practical issues associated with sensor node deployment. Herein, we sum up some major challenges specially faced by the mobility-assisted localization approaches to obtain accurate location information.

Anchor Trajectory: In mobility-assisted localization, sensor nodes can be localized only when they are in direct contact with the mobile anchor and receive sufficient signals from it. Anchor trajectory thus has to be properly planned so as to be shortest in length as well as it should be quick and full so as to provide accurate localization. A poor trajectory may result in not only large localization delay but also low localization ratio and high localization error. Due to random sensor node dropping, sensor nodes placement pattern is not known a priori. In a dynamic environment, even if the initial pattern was known, the final sensor node distribution may be different (e.g., moved by wind or animals). Anchor trajectory thus, is the key challenge for the mobility-assisted localization and it should be planned on the fly rather than beforehand.

Sensor Node Density: Mobility-assisted localization approaches are hardly required to deal with different sensor node densities. However, in a dense network with sufficient number of mobile anchors, a good localization result could be accomplished without much movement of mobile anchor. In contrast, for sparse networks, mobile anchor may require traversing more distance within the network area to localize sensor nodes. Thus, the key challenge for the localization problem in a sparse network is to achieve the maximum localization accuracy by traversing optimal path, given a limited number of anchors.

Noisy Measurements: Since measurements on proximity, range and angle are subject to noise due to inherent uncertainty of a wireless signal, mobility-assisted localization approaches are expected to be able to deal with noisy measurements. Therefore, additional efforts on modeling the noises and alleviating the impacts on localization performance are critical for the success of mobility-assisted localization methods.

Infrastructure-Less Environment: Sensor nodes are generally deployed in some inaccessible terrain or areas where infrastructures are very less. In order to estimate the sensor node's relative location to the moving anchor using RSS, it is necessary to calibrate the system, thus obtaining the propagation characteristic of the beacon in the air. Hence, the design of mobility-assisted localization schemes should be automatic without human calibration and extensive environment profiling.

Obstacles and Terrain Irregularities: Obstacles and terrain irregularities jointly can also cause devastation on mobility-assisted localization process. Large rocks can occlude line of sight, prevent measuring range, or interfere with radios, introduce errors in range measurement and produce incorrect location information. In indoor environment, natural features like walls can hinder measurements as well. All of these challenges are likely to come up in real life implementations, so mobility-assisted localization schemes should be able to cope up with these.

Resource Constraints: To enable cooperation among sensor nodes in the mobility-assisted localization process, information exchange between neighbouring sensor nodes adds to energy consumption and bandwidth occupancy. For example, in centralized localization algorithms, where cooperation is orchestrated through a central node (usually the base station), extra communication cost is incurred for collecting and forwarding the measurements to the base stations and sending the localization results to the nodes.

Table 1. Challenges and their probable solution in mobility-assisted location approach

Challenge	Highlight of probable solution
Anchor trajectory planning	Must be planned on the fly rather than beforehand to optimize the traversing path.
Low sensor node density	Difficult to achieve the maximum localization accuracy in sparse network.
Range measurement in presence of noise	Additional efforts are needed to mitigate the impact of noise.
Localization in infrastructure-less environment	Automated localization scheme is most preferrable than manual.
Presence of obstacles and terrain irregularity	Additional efforts are required to alleviate the impact of occlude line of sight, interference.
Resource constraint	Distributed localization scheme more useful for resource constraint sensor node.

Table 1 presents a summary of the main chanllanges faced by the mobility-assisted localization schemes and their possible resolution. It is worth noting that there is no single solution that combats all the challenges. The choice of the solution depends on the application, scenario, and available resources.

3 Mobility-Assisted Localization Approaches

It has been seen that mobility of sensor nodes has double impact on the localization process. On the flip side, the uncertainty of sensor node movements leads to increased difficulty in localization. As mentioned in section 2.1 localization approaches are categorized into range-based and range-free that can be further divided into subcategories using common approaches. Range-based approaches are typically based on angle-of-arrival (AoA), received signal strength indicator (RSSI), time-of-arrival (ToA), time-difference-of-arrival (TDoA) measurements. Range-free localization uses topological information (e.g., hop count) rather than range information.

3.1 Range-Based Localization Approaches

Geometric techniques manage estimating the locations of the sensor nodes from the range measurement and geometric computations. The underlying idea is that Euclidean distance between two sensor nodes is measured by their radio signals through ToA, TDoA, RSSI etc. After obtaining at least three different Euclidean distances of mobile anchor, the unlocalized sensor node applies either trilateration or multilateration algorithm for finding out its own location. In this section, we have reviewed few popular and innovative range-based techniques for measuring distances between mobile anchors and unlocalized sensor node. Further, we have discussed how measured distances are used by the unlocalized sensor node for estimating its own location. Pros and cons of each technique are also discussed in this section.

3.1.1 Localization Based on ToA Measurement

Sensor node localization using time-of-arrival (ToA) [17-19] technique measures the time a signal takes to arrive at several number of sensor nodes. ToA measurement requires knowing the time when the signal was transmitted, thus, most of the time, synchronization between sender and receiver is needed. However, there are existing works where ToA measurement is done without time synchronization [17, 19]. Next, we have discussed about the algorithmic approach of sensor node localization based on ToA measurement.

Let *X*, be an arbitrary positioned unlocalized sensor node. Mobile anchor moves from one direction to another direction while transmitting beacon, as shown in Fig. 2. ToA measurement is performed in two steps. In the first step, mobile anchor broadcasts a beacon or a ranging request which can be received by all the unlocalized sensor nodes in the radio range. Then in the second step, each unlocalized sensor node sends an acknowledgement (ACK) to the mobile anchor for responding to the request. In order to prevent collision, the unlocalized sensor node performs a random or schedule back-off before sending the ACK.

Now, ToA measurement obtains the distance in the following way: Distance between mobile anchor and unlocalized sensor node *X* is $d_A^X = c\left(T_{X,1} - T_{A,1}\right)$, where c is the speed of light, $T_{A,1}$ is the time when mobile anchor broadcasts beacon and $T_{X,1}$ is the time when *X* receives the beacon. Similarly, distance between unlocalized sensor node *X* and mobile anchor is $d_X^A = c\left(T_{A,2} - T_{X,2}\right)$ where $T_{X,2}$ and $T_{A,2}$ are the times when *X* sends ACK and mobile anchor receives ACK respectively. Since, here $\left(T_{A,2} - T_{A,1}\right)$ and $\left(T_{X,2} - T_{X,1}\right)$ are the elapsed times at mobile anchor and unlocalized sensor node *X* respectively, they can be calculated using local clock of mobile anchor and unlocalized sensor node.

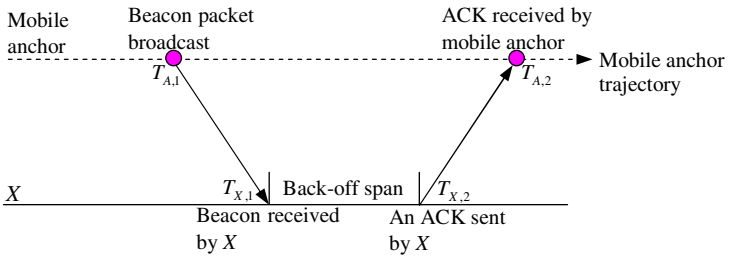


Fig. 2. An example of basic ToA ranging scheme

Therefore, calculated distance between mobile anchor and unlocalized sensor node *X* is

$$\frac{d_A^X + d_X^A}{2} = \frac{c}{2}\left[\left(T_{A,2} - T_{X,2}\right) - \left(T_{X,1} - T_{A,1}\right)\right].$$

The major challenge facing ToA based ranging techniques is the difficulty in accurately measuring when the signal was transmitted, since the propagation speed could be extremely high compared to the distance to be measured.

In [17-19], ToA based ranging technique as discussed above has been used to measure distance between mobile anchor and unlocalized sensor node. Further, to improve the location accuracy, in [17, 19] authors have considered a planned anchor trajectory where mobile anchor follows an S-shape route through the deployment area to localize the sensor nodes.

3.1.2 Localization Based on TDoA Measurement

Time-difference-of-arrival (TDoA) based range measurement techniques improve upon the ToA based range measurement technique by eliminating the need to know exact time when the signal was transmitted. In TDoA based approach, range is measured in the following way: a mobile anchor transmits RF and ultrasonic signals one after another, as shown in Fig. 3. An unlocalized sensor node *X* receives those signals and computes the time-difference-of-arrived signals i.e. δ. Finally distance between the mobile anchor and unlocalized node is obtained by multiplying δ and the speed of the ultrasonic signal (about 344.424 m/s at room temperature).

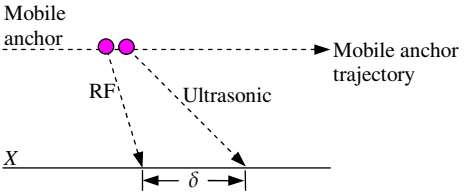


Fig. 3. An example of time-difference-of-arrival

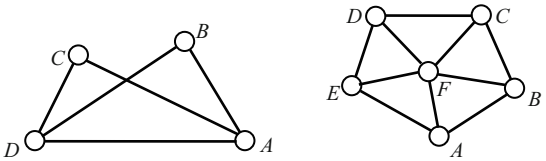


Fig. 4. Examples of graphs (a) locally rigid but not globally rigid, (b) globally rigid

Computing the distance between each pair of node's locations e.g., mobile anchor and unlocalized sensor node is a trivial problem whereas it is not trivial to find locations of nodes when Euclidean distances between each pair of nodes are given. This problem can be formulated as a graph realization problem, aiming at mapping the nodes in the graph to points in the plane so that the Euclidean distances between nodes equal the respective edge weights. Priyantha *et al.* [20] have proposed a localization scheme using mobile nodes based on TDoA measurement. As finding the location from Euclidean distances between each pair of nodes is not trivial, therefore, in [21] authors have provided a novel solution by designing a movement strategy that produces a global rigid graph (shown in Fig. 4) of known distances among the static sensor nodes. Using those known distances, finally unlocalized sensor nodes calculate their own locations.

3.1.3 Localization Based on RSSI Measurements

Another category of range measurement techniques estimates the distance between mobile anchor and sensor node from the RSS measurements. Since the sensor nodes are equipped with radios to perform communication, the distance estimation by measuring the radio signal strength requires no additional hardware, and is unlikely to significantly impact local power consumption, sensor size and hence cost. A simplified model for RSSI based range measurement is given by the following equation-

$$RSSI \propto d^{-\alpha}$$

where d is the distance between mobile anchor and unlocalized sensor node and α is a constant relevant to the atmosphere. Given a RSSI value measured by the radio of unlocalized sensor node, the unlocalized sensor node is able to calculate its distance from the mobile anchor.

In [22-24], authors have proposed an energy efficient localization scheme based on mobile anchor where distance between mobile anchor and unlocalized sensor node is calculated by RSSI measurement. RSSI based range measurement is extremely susceptible to multipath fading, variations in temperature and humidity. Therefore, localization scheme based on RSSI range measurement is prone to errors. To mitigate the erroneous range measurement, profiling [25] has been used in which a map of RSS values is constructed during an initial training phase. Sensor nodes then estimate their positions by matching observed RSS values with the training data. Further, a novel mobile-assisted localization scheme called perpendicular intersection (PI) has been proposed in [24]. Instead of directly mapping RSSI values into physical distances, by contrasting RSSI values from the mobile beacon to a sensor node, PI utilizes the geometric relationship of a perpendicular intersection to compute sensor node positions. Through real life implementation using TelosB motes it has been shown that the PI achieves high location accuracy and low overhead. Further to improve location accuracy, in [23], Kim and Lee have proposed a trajectory planning for the mobile anchor with an objective of minimizing the movement energy consumption per unit distance and transmission energy consumption per beacon.

3.1.4 Localization Based on Network Density Clustering

A novel mobile anchor-assisted localization algorithm based on network-density clustering (NDC) for WSNs has been proposed in [26]. Initially, authors have proposed a network-density based clustering scheme. The clustering scheme chooses a sensor node as the cluster head with the highest local core density. After choosing the cluster head, based on density-reachable principle, member nodes of the cluster are chosen. After forming clusters, the scheme uses multidimensional scaling map (MDS-MAP) [27] to obtain the initial coordinate of all the cluster heads. The MDS approach includes three steps. The first step is to form the distance matrix with distances between all pairs of nodes in the network by measuring either RSS or ToA. In the second step, the singular vector decomposition (SVD) is performed to determine an initial relative map of the sensor nodes on the plane. The last step performs the necessary flip, rotation and scaling according to the distances between mobile anchors. After knowing

the coordinates, the cluster head becomes the anchor. Now both cluster head and mobile anchor help the unlocalized member node in becoming a localized node. Authors have considered the trajectory planning of the mobile anchor as a traveling salesman problem, in which the mobile anchor traverses all the cluster heads. Since the traveling salesman problem, is an NP-complete problem, in order to reduce computational complexity, authors have adopted a heuristic method e.g. genetic algorithm to obtain sub-optimal solution for path planning of mobile anchor.

The proposed technique improves both the utilization rate of the mobile anchor and localization accuracy with reduced anchor movement length. But the time complexity of the proposed scheme is considerably high and is about $O(n^3)$, where n is the number of nodes in the network.

3.2 Range-Free Based Localization Approaches

Range measurements, often, may not be obtainable due to various constraints e.g. cost. Under these circumstances, proximity information provided by the radios attached to the sensor nodes could lead to adequate solutions for the localization problem. In this section, we have discussed about the underline idea of few popular range-free methods used in mobility-assisted localization techniques.

3.2.1 Localization Based on Monte Carlo Method

One of the well known range-free techniques specially used in mobility-assisted localization technique is sequential Monte Carlo (SMC) method. In SMC based localization technique [28-30], possible locations of a sensor node are represented with a set of sample locations, which are updated when mobile anchors move. They provide the coordinate of a new location using the SMC approach. Location estimation using SMC method is performed in the following way:

Let t be the discrete time, l_t denotes the position distribution of the sensor node at time t, l_t^i denotes the ith sample of the location of a sensor node at time t, and o_t denotes the observations from mobile anchors received between time (t-1) and t. A transition equation $p(l_t \mid l_{t-1})$ describes the prediction of sensor node's current position based on previous position, and an observation equation $p(l_t \mid o_t)$ describes the likelihood of the sensor node being at location l_t given the observations.

Eventually, from the above discussion, the Monte Carlo method based localization techniques require quite a number of steps in order to compute location of sensor node. The main steps of the SMC method based localization techniques are as follows-

Initialization: The sensor node has no knowledge about its position at time 0, so the initial samples are selected randomly from all possible locations:

$$\left\{ l_0^i \mid 1 \le i \le N \right\} \xleftarrow{Initialize} Random\ positions.$$

Prediction: At time t, the sensor node uses the transition distribution $p(l_t \mid l_{t-1})$ to predict its possible locations based on previous samples and its variation at time (t-1):

$$\left\{l_t^i \mid 1 \le i \le N\right\} \xleftarrow{p(l_t|l_{t-1})} \left\{l_{t-1}^i \mid 1 \le i \le N\right\}.$$

Filtering: At time *t*, the sensor node uses new information to eliminate predicted locations that are inconsistent with observations:

$$\left\{l_t'^i \mid 1 \le i \le N\right\} \xleftarrow{p(l_t|o_t)} \left\{l_t^i \mid 1 \le i \le N\right\},$$

where $l_t'^i$ denotes the *i*th sample of the location of a sensor node after filtering step.
Re-sampling: The purpose of re-sampling step is to gradually remove samples with lower weights and keep those with higher weights:

$$\left\{l_t''^i \mid 1 \le i \le N\right\} \xleftarrow{\text{Re}-sampling} \left\{l_t'^i \mid 1 \le i \le N\right\},$$

where $l_t''^i$ denotes the *i*th sample of the location of a sensor node after re-sampling step.

The main drawbacks of SMC are that it needs a high density of mobile anchor and the sampling technique it uses to generate probable locations is very slow and computation-intensive. Rudafshani and Datta [28] have presented a mobile-assisted localization for sensor networks based on SMC method. In order to mitigate the drawbacks, in the proposed scheme, each unlocalized sensor node uses the weights of its neighbours (rather than weights of samples of neighbours) to weigh its samples. Evaluation results of this scheme confirm improved localization accuracy and low dependency on the number of mobile anchors.

3.2.2 Localization Based on Convex Method

The presence of obstacles such as mountains or building in the node deployment area imposes huge challenge in localizing the sensor nodes for the mobility-assisted localization techniques. One such scheme is proposed in [31], where authors have considered the presence of obstacles during the localization of the sensor nodes. The working principle of the proposed technique is elaborated below.

Let us consider the current position of the mobile anchor is *a* and lower and upper bounds of transmission radii are *r* and *R* respectively, as shown in Fig. 5. If the sensor node (square box in Fig. 5), located at position *x*, receives the beacon signal, it can be concluded that the distance between the mobile anchor and sensor node satisfies either $\|x-a\| \le R$ or $\|x-a\| > r$.

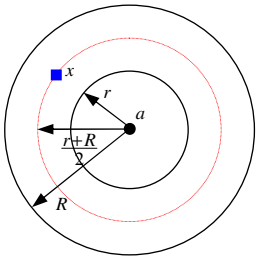


Fig. 5. The single constraint case

Hence, for the single constraint case, the estimated position can be found by minimizing the following expression-

$$\left(\|x-a\|-r\right)^2+\left(\|x-a\|-R\right)^2.$$

Similarly, for the inequalities under multiple constraints, optimal position of sensor node is obtained by solving the following problem:

$$\max_x \sum_t \left[\left(\|x-a_t\|-r_t\right)^2+\left(\|x-a_t\|-R\right)^2\right],\ t=1,2,\ldots,K$$

where a_t is the position of the mobile anchor at time t, r_t and R_t denote the lower and upper bounds of the transmission radius of mobile anchor at position a_t in time t. It is evident from the given equation that the problem is nonconvex.

As the above problem is nonconvex and cannot be directly approximated by using convex relaxation techniques, therefore, it is transformed into the equivalent following convex problem-

$$\max_{x,y} \sqrt{\sum_t \left[\left(y-2a_t^T x+\|a_t\|^2-r_t^2\right)^2+\left(y-2a_t^T x+\|a_t\|^2-R_t^2\right)^2\right]}$$

s.t. $\|x\|^2 \le y$

By solving the above convex problem using interior-point algorithms [31], sensor node can obtain its own location. The benefit of the proposed localization technique is that it can provide accurate location for both feasible and infeasible cases [31]. However, as the location calculation involves solving several problems, the computation cost of each sensor node increases significantly.

3.2.3 Localization Based on Geometric Constraints

The geometric measurement based localization technique involves three steps in localizing the sensor nodes. The steps are as follows: 1) select three anchor points among the received beacons; 2) obtain the intersection area with two anchor points using geometric constraints; and 3) calculate the location with the third anchor point.

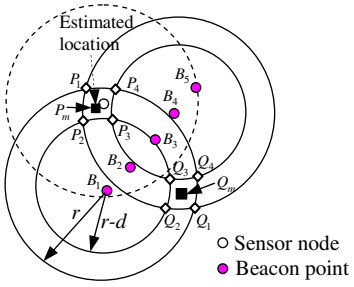


Fig. 6. Location estimation using geometric constraint method

1) Three Beacon Points Selection: It is assumed that a mobile anchor moves around the deployed area of the sensor nodes at a constant speed and broadcasts a beacon that includes its own absolute location information every *d* distance intervals, called beacon distance. Now, if a sensor node receives a beacon, it concludes that mobile anchor is located within its own communication circle. This is referred as the beacon point for location of mobile anchor.

When a sensor node receives the first beacon from a mobile anchor, that location is selected as a beacon point, e.g. B_1, B_2 etc., as shown in Fig. 6. If the sensor node receives no further beacons during a predefined time after receiving its last beacon, this beacon is selected as a beacon point; i.e., B_4 in Fig. 6. The above process is repeated each time the mobile anchor passes through the communication circle of the sensor node; i.e. B_1, B_4, and B_5 are finally selected as three beacon points.

2) The Intersection Area: Since each sensor node receives a beacon at every beacon point *d* and communication/radio range of a mobile anchor is *r*, so, the beacon points are located between the distances (*r-d*) and *r* from the sensor node. Therefore, from the geometric constraints, the sensor node must be in the ring area defined by two circles with radii *r* and (*r-d*) from a beacon point. Now, if the first beacon point is obtained, the sensor node is positioned within the ring area made by the beacon point. Further, if the second beacon point is obtained, the sensor node is also positioned within the second ring area.

Hence, it can be concluded from the discussion that the sensor node is located within the intersection area among the two ring areas, as shown in Fig. 6; if we let one piece of area constituting of vertices P_1, P_2, P_3 and P_4 be A_P and the other piece A_Q, location of the sensor node is within area $A_P \cup A_Q$.

3) Location Calculation: After obtaining the area $A_P \cup A_Q$, one can calculate the midpoint P_m of the intersection points P_1 and P_3, and Q_m of Q_1 and Q_3. The midpoints P_m and Q_m are potential location of the unlocalized sensor node. However, when the distance between the two beacon points is more than 2(*r-d*), two circles with a radius of (*r-d*) have no intersection points (P_3 and Q_3). In this case, P_3 and Q_3 are defined as the midpoints of the two beacon points.

The performance of the geometric constraint (GeoCon) based location estimation method exhibits minimization of average location error [32] even under the constraint of smaller communication range or beacon distance.

So far, we have discussed several potential methods that have been used to compute the location of a sensor node. Such methods include ToA, TDoA, RSSI, NDC, SMC, Convex, Geometric constraint. The choice of the method also influences the final performance of the localization system. Such a choice depends on the available information and on the processor limitations. Table 2 compares each one of the methods described in this section.

Table 2. Comparison of the methods used to estimate location

Method	Extra Hardware	Challenges
ToA [15]	None	Nodes synchronization
TDoA [17]	Ultrasound transmitter	Limited distance of work
RSSI [21]	None	Interference, Variation of the RSSI
NDC [23]	None	High time complexity
SMC [27]	None	Very slow and computation-intensive
Convex [28]	None	High computation cost
GeoCon [29]	None	Location error depends on radio range

4 Trajectory Planning in Mobility-Assisted Localization Techniques

The localization accuracy can be enhanced by increasing the resolution of the movement trajectory for an arbitrarily faster mobile anchor. Therefore, a fundamental issue in mobility-assisted localization technique is the planning of the movement trajectory of the mobile anchor to maximize the localization accuracy. Fundamentally, trajectory planning for a particular application has two goals: (a) to offer network coverage and (b) to provide good quality beacons. Compared to the first goal, the second goal of path planning, which is unique in the sensor network localization problem, is much more challenging. In this section, we confer four well-known anchor trajectories, which offer general desirable characteristics, and identify, through detailed simulations, which of them offers higher localization accuracy. These four mobile anchor trajectories are namely SCAN, HILBERT [33], CIRCLES [34], and DREAMS [13].

4.1 SCAN

A simple and easily implementable mobile anchor trajectory is SCAN [33]. SCAN divides the square deployment area into $q \times q$ sub-squares and connects their centers using straight lines as shown in Fig. 7(a). In SCAN, the mobile anchor traverses the network area along 1-D either along the x axis or y axis. If the mobile anchor travels along the y axis, the distance between two successive segments of the trajectory, which are parallel to the y axis, defines the resolution of the trajectory. If the communication range of the sensor node is R', the resolution should be at most $2R'$ in SCAN, to make sure that the sensor node receives the beacons. The benefit of using SCAN is that it offers uniform coverage to the whole network area, and ensures that all sensor nodes are able to receive beacons from the mobile anchor under a properly selected resolution. However, the imperative weakness associated with SCAN is collinearity of beacons. To be more specific, for large resolution, many sensor nodes will receive beacons only from one line segment and one direction, which create uncertainty and prevent them from obtaining a good estimate along the x axis. To evade this problem, the trajectory must be adequately dense for the sensor nodes so that sensor node will be able to hear the mobile landmark when it moves on two successive segments along the y axis.

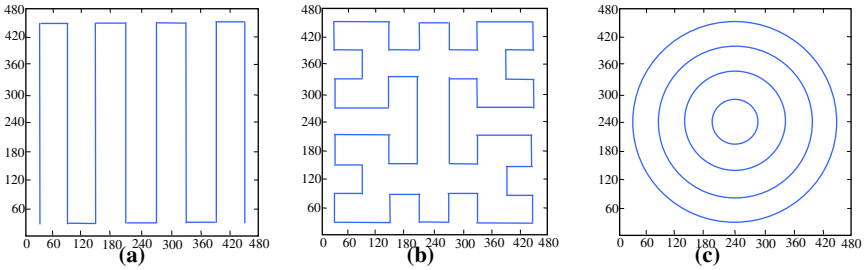


Fig. 7. The mobile anchor trajectories in deployment area 480×480 sq. m and Resolution 60 m. (a) SCAN, (b) HILBERT, (c) CIRCLE.

4.2 HILBERT

In order to reduce the collinearity without significantly increasing the path length, HILBERT curve based technique, namely HILBERT, is proposed in [33], which makes the mobile anchor to take more turns. Similar to SCAN, HILBERT divides the 2-D square deployment area into square cells and connects their centers using straight lines as shown in Fig. 7(b). For example, a level-q HILBERT curve divides the 2-D space into 4^q square cells and centers of those cells are connected using 4^q line segments, each of length equal to the length of the side of a square cell. Therefore, the resolution of HILBERT is the length of the side of a square cell. As the HILBERT curves based trajectory planning ensures more path turns compared to SCAN, therefore, sensor nodes receive non-collinear beacons and obtain a good estimate for their positions. Since HILBERT curve always connects the centers of two successive square cells, the mobile anchor will never move on the border of the deployed area. Thus, in HILBERT, sensor nodes near the border will possibly receive beacons only from one direction and their location estimates will not be accurate.

4.3 CIRCLES

Since the straight line based trajectories of mobile anchor perpetually introduce collinearity, thereby a circular trajectory called CIRCLES is proposed in [34]. CIRCLES consist of a sequence of concentric circles centered within the deployment area as shown in Fig. 7(c). In CIRCLES, the resolution (R) is half of the radius of the innermost circle, and it sequentially increases the radius by R at each outer circle. The main benefit of using CIRCLES is that as it does not introduce collinearity, therefore, all senor nodes with in the circles become localized. However, when deployment area is square, CIRCLES fails to cover the four corners effectively without adding larger circles. Now, if large circle is added in order to cover four corners then basically the path length of mobile anchor are increased. Furthermore, CIRCLES has an inherent scalability issue. To be more specific, when the deployment area increases, CIRCLES require the anchor path to contain larger circles. As the circles become larger, the amount of non-collinearity reduces, which in turn reduces the localization accuracy.

4.4 DREAMS

A deterministic dynamic anchor mobility scheduling (DREAMS) algorithm is proposed in [13] where no prior knowledge about the deployed area is needed. The anchor trajectory is defined by the track of depth-first traversal (DFT) of the network graph and so it is deterministic. The mobile anchor performs DFT dynamically, under the instruction of nearby sensor nodes on the fly. The mobile anchor at first visits a sensor node by moving randomly and then performs a DFT on the network graph based on the instruction of the present visited sensor node. It stops moving once it returns to the first sensor node and the sensor node has no unvisited neighbours. During DFT, the anchor performs intelligent distance-based heuristic movement from node to node following RSS, and sensor nodes run the built-in localization procedure to self-localize using received beacon signals. To shorten the anchor trajectory, DFT may be performed using a local minimum spanning tree subgraph, whose edges are weighted by RSS. Also unvisited, but localized, sensor nodes may be excluded from DFT if the exclusion does not affect discovery of unlocalized sensor nodes. Real life implementation of DREAMS shows that it produces accurate location estimation.

5 Summary on Mobility-Assisted Localization Techniques

In this section we have discussed on the evaluation of existing mobility-assisted localization schemes. The ability to fix the location of a sensor node in terms of fine-grain location would determine the usefulness of a particular mobility-assisted localization scheme. The three basic evaluation metrics exist to evaluate the usefulness of a particular mobility-assisted localization scheme and they are localization accuracy, computation and communication costs, and number of mobile anchor.

5.1 Localization Accuracy

A localization scheme should report locations accurately and consistently from measurement to measurement. The key metric for evaluating a localization technique is accuracy and it is defined as how much the estimated position deviates from the true position. Accuracy is denoted by an accuracy value and precision value. Precision indicates how often we expect to get at least the given accuracy. For example, some low-cost GPS receivers can locate positions within 15m for approximately 93% of the measurements. More expensive GPS receivers usually achieve much better, reaching 1m to 3m accuracies 99% of the time. Here, these distances denote the accuracy and the percentages denote precision of the location information GPS can provide. Localization accuracy relies mainly on the physical sources of localization errors. The physical sources are represented by a wide range of noises and quantization losses of range measurements. In mobility-assisted location, mobility of the anchor has severe impact on the signal compared to the static anchor. For example, the frequency of the signal may undergo a Doppler shift or introduce errors in the range measurement. Doppler shifts occur when the mobile anchor is moving relative to the unlocalized sensor

node. The resulting shift in frequency is related to the positions and relative speed of the mobile anchor and unlocalized sensor node.

In mobility-assisted localization, one can achieve fine grained localization but in exchange of increased localization delay. It is due to the fact that sensor nodes can only be localized only when they are in direct contact with the mobile anchor and receive sufficient signals from it. Anchor trajectory, thus, have to be properly planned so as to be shortest in length and meanwhile well cover every sensor for quick, full and accurate localization.

5.2 Computation and Communication Costs

As energy is one of the scarcest resources in WSNs, it is necessary to consider the computation and communication costs of the localization process in the evaluation of mobility-assisted localization schemes. In mobility-assisted localization schemes, localization accuracy is improved by mobility of the anchor at the expense of significant amount of energy consumption. Moreover, algorithm like MDS-MAP demand range measurements from all the unlocalized sensor nodes (see section 3.1.4). This is expensive in terms of forwarding the measurements to the processing point and solving the high dimension matrix. On the contrary for distributed algorithm, multihop localization faces the tradeoff between the communication cost on propagating the mobile anchor locations and the degree of accuracy. The number of iterations in a localization process is apparently in the center of tradeoff between energy consumption for improvement of localization results and the degree of accuracy achievable through refining.

5.3 Number of Mobile Anchors

It is important to note that mobility-assisted localization techniques constantly need an assured level of connectivity. The discussions on the different mobility-assisted localization algorithms suggest that relatively more number of mobile anchor in the network area lead to better localization performance. However, a more number of mobile anchors in the network area do not necessarily guarantee high accuracy in location estimations. It is due to the fact that increase in mobile anchor leads to increase in collision of beacons. So there is also need of careful planning for anchor trajectory, otherwise collisions of beacons may occur.

5.4 Summary of Performance

Existing mobility-assisted localization schemes under various scenarios were so far discussed in the previous sections. In the following section, we have summarized the performance of these schemes considering various parameters e.g., radio channel model, accuracy etc.

The localization accuracy of a solution is usually quantified using the average Euclidean distance between the estimated locations and the true locations normalized to the radio range or other system parameters [12]. For mobility-assisted localization, the

effect of sensor node density is not as important as in static localization scenarios. In addition, communication/computation cost may not be of same importance to the off-line simulations as to the real implementations. In order to compare different evaluation approaches, we have used broad set of evaluation criteria such as radio channel model, trajectory planning, scenario, computation/communication cost and accuracy. Table 3 presents the evaluation criteria that are considered for evaluation by each scheme reported in section 3. Note that this section is not meant to be an exhaustive evaluation comparison of the state-of-the-art schemes. For the sake of conciseness, radio range, velocity of the mobile anchor and beacon broadcast frequency are denoted by R, V and F respectively in the table, while h, l and N represent height, width of the network area and number of samples respectively.

Table 3. Evaluation of the existing schemes' performance

Schemes	Radio Channel Model	Trajectory Planning	Scenario	Computation/ Communication Cost	Accuracy
ToA [19]	Rayleigh channel	S-shape	Outdoor	$O(2hl/R^2)$	5m
TDoA [21]	-	Straight line	Indoor	$O(N)$	1.5%R
PI [24]	Log-normal shadowing	Equilateral triangle	Indoor /Outdoor	$O(F\,R/V)$	2.04m (Lab), 1.27m (Parking lots)
NDC [26]	-	Hexagonal	-	$O(1/R)$	20.6%R
SMC [30]	-	Random	Outdoor	$O(N^2)$	50%R
Convex [31]	Non-isotropic	Straight line	-	$O(2hl/3R^2)$	11.68%R
GeoCon [32]	Non-isotropic	Random	-	-	5m

6 Open Issues

There has been extensive research on mobility-assisted sensor node localization, nevertheless, there are several important open issues especially relevant to mobility assisted localization in a WSN which either remain unsettled or unexplored comprehensively. Some of these issues are listed below.

1. **Energy Consumption:** The problem of minimizing energy consumption of the mobility-assisted localization process deserves more attention. Even though, energy consumption issues have been addressed in the existing mobility-assisted localization technique, the energy efficiency goal still remains a challenge.
2. **Design Complexity:** The moving trace of mobile anchor must be optimized since mobile anchors are only capable of low-speed and short-distance mobility in real environment due to high power consumption of locomotion. Since the distribution of mobile anchors can affect location performance in static WSNs, therefore,

efficient trajectory planning for mobile anchors can further increase location accuracy for target estimation.

3. **Non-convex Topologies:** Localizing the sensor nodes located in the boundary is a problem because less information is available about them and that too of lower quality. This problem is exacerbated when a node deployment area has a non-convex shape. Sensor nodes outside the main convex body of the deployment area can often prove to be unlocalizable. Even when locations can be found, the results tend to feature disproportionate error. Further, an efficient trajectory planning for mobile anchors can increase location accuracy in such situation.
4. **Cost:** Several existing works has shown that, the use of mobile anchors in sensor node localization is beneficial, because it provides additional measurements on spatial relationships along their corresponding trajectories. However, a mobile anchor, with comparatively more resources than an ordinary sensor node, is expensive. Therefore, only a small number of mobile anchors can realistically be used for localization. Further, those small numbers of mobile anchors must effectively cooperate with sensor nodes to obtain maximum utility.
5. **3-D Localization:** In the existing scenario, sensor node localization is typically to find out the location of nodes in a 2-D network area. However, in real life application, sensor nodes are usually deployed in a 3-D space, which leads to differences on both ranging results and localization schemes. Investigation on mobility-assisted localization schemes focusing on the 3-D space is of particular interests to real life applications of WSNs. In [15, 35], an attempt has been made to localize the sensor nodes in a 3-D network. However, the existing localization schemes in 3-D space have not been completely examined.

7 Conclusion

Discovering accurate locations of sensor nodes in WSNs is decisive to both network functions and most application level tasks. In this chapter, we have presented key issues and inherent challenges faced by the mobility-assisted localization techniques in WSNs. Further, we discussed the algorithmic approaches of various important fine-grained mobility-assisted localization techniques. In the chapter, mobility-assisted localization techniques are usually referred to as either range-based or range-free. However, such a wide categorization is grossly insufficient, because it restricts categorization for hardware requirements of the localization schemes. In order to validate the proposed line of investigation, we reviewed existing solutions, discussed the difficulties of using range measurements and proximity information in details for mobility-assisted localization techniques. In addition, well-known mobile anchor trajectories presented in existing works are also reviewed in details. Further, a summary of simulation results of important mobility-assisted localization techniques is presented on the basis of secenario, location accuracy, computation/communication costs etc. Several open issues for further research have also been included.

References

1. Aspnes, J., Eren, T., Goldenberg, D.K., Morse, A.S., Whiteley, W., Richard Yang, Y., Anderson, B.D.O., Belhumeur, P.N.: A Theory of Network Localization. IEEE Trans. on Mobile Computing 5(12), 1663–1678 (2006)
2. Liu, Y., Yang, Z., Wang, X., Jian, L.: Location, Localization, and Localizability. J. of Computer Science and Technology 25(2), 274–297 (2010)
3. Sichitiu, M.L., Ramadurai, V.: Localization of Wireless Sensor Networks with a Mobile Beacon. In: Proc. of Int'l Conf. on Mobile Ad-Hoc and Sensor Systems, pp. 174–183 (2004)
4. Kuhn, F., Wattenhofer, R., Zhang, Y., Zollinger, A.: Geometric Ad-Hoc Routing: of Theory and Practice. In: Proc. of 22nd Annual Symposium on Principles of Distributed Computing, pp. 63–72 (2003)
5. Tan, G., Bertier, M., Kermarrec, A.M.: Visibility-graph-based Shortest-path Geographic Routing in Sensor Networks. In: Proc. of 28th Annual IEEE Int'l Conf. INFOCOM, pp. 1719–1727 (2009)
6. Shu, L., Zhang, Y., Yang, L.T., Wang, Y., Hauswirth, M., Xiong, N.: TPGF: Geographic Routing in Wireless Multimedia Sensor Networks. Telecommunication Systems 44(1-2), 79–95 (2010)
7. Zhang, W., Cao, G.: DCTC: Dynamic Convoy Tree-based Collaboration for Target Tracking in Sensor Networks. IEEE Trans. on Wireless Communications 3(5), 1689–1701 (2004)
8. Zhang, L., Cheng, Q., Wang, Y., Zeadally, S.: A Novel Distributed Sensor Positioning System using the Dual of Target Tracking. IEEE Trans. on Computers 57(2), 246–260 (2008)
9. Hu, L., Evans, D.: Localization for Mobile Sensor Networks. In: Proc. of 10th Int'l Conf. on Mobile Computing and Networking, pp. 45–57 (2004)
10. Mao, G., Fidan, B., Anderson, B.D.O.: Wireless Sensor Network Localization Techniques. Computer Networks 51(10), 2529–2553 (2007)
11. Amundson, I., Koutsoukos, X.D.: A Survey on Localization for Mobile Wireless Sensor Networks. In: Fuller, R., Koutsoukos, X.D. (eds.) MELT 2009. LNCS, vol. 5801, pp. 235–254. Springer, Heidelberg (2009)
12. Wang, J., Ghosh, R.K., Das, S.K.: A Survey on Sensor Localization. J. Control Theory Application 8(1), 2–11 (2010)
13. Li, X., Mitton, N., Simplot-Ryl, I., Simplot-Ryl, D.: Dynamic Beacon Mobility Scheduling for Sensor Localization. IEEE Trans. on Parallel and Distributed Systems 23(8), 1439–1452 (2012)
14. Natalizio, E., Loscrí, V.: Controlled Mobility in Mobile Sensor Networks: Advantages, Issues and Challenges. Telecommunication Systems 52(4), 2411–2418 (2013)
15. Cui, H., Wang, Y.: Four-Mobile-Beacon Assisted Localization in Three-Dimensional Wireless Sensor Networks. Computers and Electrical Engineering 38(3), 652–661 (2012)
16. Kuo, S.-P., Kuo, H.-J., Tseng, Y.-C.: The Beacon Movement Detection Problem in Wireless Sensor Networks for Localization Applications. IEEE Trans. on Mobile Computing 8(10), 1326–1338 (2009)
17. Luo, J., Shukla, H.V., Hubaux, J.-P.: Non-Interactive Location Surveying for Sensor Networks with Mobility-Differentiated ToA. In: Proc. of 25th IEEE Int'l Conf. INFOCOM, pp. 1241–1252 (2006)

18. Chan, F.-K., Wen, C.-Y.: AOA-aided TOA Distributed Positioning for Mobile Wireless Sensor Networks. In: Proc. of Int'l Conf. on Industrial Electronics and Applications, pp. 1774–1779 (2010)
19. Chen, H., Liu, B., Huang, P., Liang, J., Gu, Y.: Mobility-Assisted Node Localization Based on ToA Measurements Without Time Synchronization in Wireless Sensor Networks. Mobile Networks and Application 17(1), 90–99 (2012)
20. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: Proc. of 6th Int'l Conf. on Mobile Computing and Networking, pp. 32–43 (2000)
21. Priyantha, N.B., Balakrishnan, H., Demaine, E.D., Teller, S.: Mobile-assisted Localization in Wireless Sensor Networks. In: Proc. of 24th Int'l Conf. on INFOCOM, vol. 1, pp. 172–183 (2005)
22. Graefenstein, J., Albert, A., Biber, P., Schilling, A.: Wireless Node Localization based on RSSI using a Rotating Antenna on a Mobile Robot. In: Proc. of 6th Workshop on Positioning, Navigation and Communication, pp. 253–259 (2009)
23. Kim, K., Lee, W.: MBAL: A Mobile Beacon-Assisted Localization Scheme for Wireless Sensor Networks. In: Proc. of 16th Int'l Conf. on Computer Communications and Networks, pp. 57–62 (2007)
24. Guo, Z., Guo, Y., Hong, F., Jin, Z.: Perpendicular Intersection: Locating Wireless Sensors with Mobile Beacon. IEEE Trans. on Vehicular Technology 59(7), 3501–3509 (2010)
25. Ladd, A., Bekris, K., Rudys, A., Wallach, D., Kavraki, L.: On the Feasibility of using Wireless Ethernet for Indoor Localization. IEEE Trans. on Robotics and Automation 20(3), 555–559 (2004)
26. Zhao, F., Luo, H., Quan, L.: A Mobile Beacon-Assisted Localization Algorithm based on Network-Density Clustering for Wireless Sensor Networks. In: Proc. of 5th Int'l Conf. on Mobile Ad-Hoc and Sensor Networks, pp. 304–310 (2009)
27. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.J.: Localization from Mere Connectivity. In: Proc. of 4th ACM Int'l Symposium on Mobile Ad Hoc Networking & Computing, pp. 201–212 (2003)
28. Rudafshani, M., Datta, S.: Localization in Wireless Sensor Networks. In: Proc. of 6th Int'l Symposium on Information Processing in Sensor Networks, pp. 51–60 (2007)
29. Klingbeil, L., Wark, T.: A Wireless Sensor Network for Real-Time Indoor Localization and Motion Monitoring. In: Proc. of Int'l Conf. on Information Processing in Sensor Networks, pp. 39–50 (2008)
30. Huang, R., Zaruba, G.V.: Monte Carlo Localization of Wireless Sensor Networks with a Single Mobile Beacon. Wireless Networks 15(8), 978–990 (2009)
31. Chen, H., Shi, Q., Tan, R., Poor, H.V., Sezaki, K.: Mobile Element Assisted Cooperative Localization for Wireless Sensor Networks with Obstacles. IEEE Trans. on Wireless Communications 9(3), 956–963 (2010)
32. Lee, S., Kim, E., Kim, C., Kim, K.: Localization with a Mobile Beacon Based on Geometric Constraints in Wireless Sensor Networks. IEEE Trans. on Wireless Communications 8(12), 5801–5805 (2009)
33. Koutsonikolas, D., Das, S., Hu, Y.: Path Planning of Mobile Landmarks for Localization in Wireless Sensor Networks. Computer Communication 30(13), 2577–2592 (2007)
34. Huang, R., Zaruba, G.V.: Static Path Planning for Mobile Beacons to Localize Sensor Networks. In: Proc. of IEEE Int'l Conf. on Pervasive Computing and Communication, pp. 323–330 (2007)
35. Ou, C.-H., Ssu, K.-F.: Sensor Position Determination with Flying Anchors in Three-Dimensional Wireless Sensor Networks. IEEE Trans. on Mobile Computing 7(9), 1084–1097 (2008)

Part II

Cooperative Robots and Sensor Networks Applications

On the Cooperation between Mobile Robots and Wireless Sensor Networks

Chia-Yen Shih[1], Jesús Capitán[1], Pedro José Marrón[1], Antidio Viguria[2], Francisco Alarcón[2], Marc Schwarzbach[3], Maximilian Laiacker[3], Konstantin Kondak[3], José Ramiro Martínez-de Dios[4], and Aníbal Ollero[4]

[1] Networked Embedded Systems Group, Faculty of Business Administration & Economics, Dept. of Computer Science & Business Information Systems (ICB) University of Duisburg-Essen, Schtzenbahn 70, 45117, Essen, Germany
{chia-yen.shih,jesus.capitan,pjmarron}@uni-due.de

[2] Center for Advanced Aerospace Technologies (CATEC), C/Wilbur y Orville Wright 17-19-21 41309 La Rinconada, Seville, Spain
{aviguria,falarcon}@catec.aero

[3] German Aerospace Center (DLR), Institute of Robotics and Mechatronics Robotics and Mechatronics Center (RMC), Oberpfaffenhofen 82234, Wessling, Germany
{marc.schwarzbach,maximilian.laiacker,konstantin.kondak}@dlr.de

[4] Robotics, Vision and Control Research Group, University of Seville, Depto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingeniería, Avda. Camino de los Descubrimientos, sn 41092, Seville, Spain
{jdedios,aollero}@cartuja.us.es

Abstract. Employing cooperative heterogeneous systems can enrich application scenarios and achieve higher application performance. The combination of mobile robots and Wireless Sensor Networks (WSNs) is a good example of such cooperation, and many recent research results have highlighted the benefits of the marriage of these two technologies. The main objectives of this chapter include: (1) providing a survey on a variety of applications with cooperating mobile robots and WSNs based on the roles they play for interaction, and (2) elaborating different cooperative interactions of robots and WSNs in our ongoing project, *PLAtform for the deployment and operation of heterogeneous NETworked cooperating objects* (PLANET), which is an integrated framework of heterogeneous cooperative objects for network deployment and operations.

1 Introduction

Employing cooperative heterogeneous systems can be more advantageous to enrich application scenarios than merely using homogeneous systems. By leveraging the cooperation between heterogeneous technologies, the applications can achieve higher performance. The combination of mobile robots and Wireless Sensor Networks (WSNs) is a good example of such cooperation, and many recent research results highlight the benefits of the marriage of these two technologies. For instance, by using WSNs, mobile robots can remotely monitor and

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*, Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_4,

be controlled, can navigate in unknown spaces, and can localize themselves. Reversely, by incorporating intelligent and mobile robots, a WSN can perform various tasks more efficiently and reliably, such as network deployment, sensing coverage, network connectivity, data collection, network maintenance, etc. The coupling between robots and wireless sensors has been implemented in different ways. In many robotic applications including search and rescue, environment exploration, surveillance and others, mobile robots have been used to perform tasks remotely. Such robots are equipped with on-board sensors, which make the robots *intelligent* and allow them to perceive the surroundings. Performing robotic tasks often involves complicated computation on collected data and information processing on the mobile robots. To offload the work of these robots, WSNs can be easily deployed in many scenarios to provide environment information, to monitor the mobile robots and to serve as a backbone for computation as well as communication among the mobile robots. In contrast, WSNs are often deployed to acquire physical parameter measurements and therefore are susceptible to capricious environments. Moreover, the low-priced, battery-powered sensing devices can be fragile and have limited capability in communication, computation and lifetime. Incorporating mobile robots can complement WSNs to deal with the dynamic environments and to enhance application performance in aspects such as sensing coverage, network connectivity and system longevity.

The first objective of this chapter is to provide a survey on a variety of applications, in which mobile robots and WSNs cooperatively perform the application tasks. In particular, we focus on different cooperative interactions of both technologies and discuss these interactions based on the roles played by the mobile robots and the WSNs. The second objective is to elaborate different cooperative interactions of robots and WSNs in our project, *PLAtform for the deployment and operation of heterogeneous NETworked cooperating objects* (PLANET) [50], which is an ongoing European project (FP7) that aims to develop an integrated framework of heterogeneous cooperative objects for network deployment and operations. The remainder of the chapter is structured as follows. In Section 2, we first discuss robotic applications with assisting static WSNs for robot control, robot navigation and localization; in Section 3, we provide a survey of WSN applications that are supported by mobile robots, including *Unmanned Aerial and Ground Vehicles* (UAVs and UGVs), for autonomous WSN deployment, sensing coverage, network connectivity and data collection; in Section 4, we address robot-WSN cooperation for several real-life environment monitoring applications in the Doñana National Reserve (DBR, Spain) developed in PLANET regarding WSN deployment, remote environment monitoring, data collection and surveillance. Finally, we conclude our work in Section 5.

2 WSN-Assisted Robotic Applications

2.1 Mobile Robot Control

Mobile robot control involves in monitoring and controlling the mobile robot remotely to ensure correct and safe autonomous operations. Numerous tech-

niques have been developed for robot control, using proprietary communication channels or via Internet, etc. With advancing technologies in WSNs, recent research starts to explore the sensing, computation and communication capabilities of WSNs for remote mobile robot control. In [37], a video sensor network called the Distributed Interactive Video Array (DIVA) system, was developed for autonomous coordination of UGVs equipped with on-board visual sensor with limited perspective of the surrounding environment. The DIVA project aimed to develop a wireless video camera networks (1) to track target in a global coordinate system, and (2) to command/control robots to perform autonomous deployment for intruder detection. To achieve these, DIVE employed a communication protocol for the inter-communication between the DIVA camera nods and UGVs. The DIVA prototype has been demonstrated with 255 UGVs and unmanned sensors in an physical security application for event detection. In [8], an Intelligent Space (iSpace) includes distributed sensors providing measurements for controlling mobile robots to perform physical services such as object carrying or human guiding. iSpace uses sensors, e.g., Hokuyo URG-04LX laser rangefinders, to observe the space and acquire information about the environment and the mobile robots. In [33], the authors developed a remote robot monitoring and control system for the Boe-Bot robots through Internet with a set of SunSPOT nodes (including a base station and a free-range SPOT) and on-board image sensors. The Boe-Bot is integrated with a SPOT, which interacts with the Basic Stamp Controller to drive the motor that moves the robot. The user control commands are sent via the base station to the free-range SPOT, which interacts with the on-board SPOT to drive the robot movement.

In a large-scale or complex application, a single robot may be insufficient and thus many approaches were developed using multiple cooperative mobile robots. A survey was given in [12] discussing the related work and open issues regarding multi-robot systems. These mobile robots are typically integrated with on-board sensors to perceive their environment and exchange information with each other in order to cooperatively complete the common task. WSNs can extend the perception of mobile robots by providing environment information, and thus can reduce the complexity of coordinating multiple robots. Batalin et al. [3] studied a multi-robot task allocation (MRTA) problem[28], and proposed a *Multi Field Distributed In-network Task Allocation* (DINTA-MF) algorithm using a pre-deployed static sensor network. In DINTA-MF, the idea is that the WSN computes several assignment fields in the WSN, which then distributes them to different robots using a greedy policy. In [9], a technique for distributed agent control was introduced to monitor, track and control a set of mobile agents using wireless image sensors. Robot control is based on tracking the agent movement using vision processing and on sending real-time control messages to the agents via wireless link, guiding them to the required locations. In [16], a multi-robot task assignment problem was studied with the objective of minimizing the communication overhead among sensor nodes to extend the WSN lifetime. The WSN was used to detect events, which are associated with the locations, at which a mobile robot will perform its task.

2.2 Mobile Robot Navigation

Navigation is a fundamental capability of autonomous mobile robotics. Robot navigation refers to the robot's ability (along with on-board sensors) to determine its own position, to navigate around the environment and to plan a path towards its designated goal. Thus, robot navigation focuses on three main aspects: robot localization, path planning and map building. The issue is that the limited range of the on-board sensor can hinder the navigation ability of robots. By incorporating distributed sensors, a navigation approach can extend the perception of robots to gather more information about its environment and to achieve better navigation decisions. Note that we address robot localization in the next subsection and focus on the remaining aspects in this section.

Many robotic applications (e.g., search and rescue) involve dynamic environments. In order to navigate through the area, robots need to be highly aware of the changing environment. The use of WSNs can help to aid the navigation by providing up-to-date information. Corke et al. [19] demonstrated a system in a large-scale search and rescue experiment using 54 Mica Motes to navigate a helicopter along a planned path. The implementation of such a cooperative robot-WSN interaction covers: (1) sensor node localization based on the differential GPS of the helicopter; (2) a computed path based on a built map, which is encoded as a distributed representation of obstacles; and (3) robot navigation along the created path by interacting with the ground sensors. Batalin et al. [4] proposed an algorithm in which, given a specific goal, the sensor nodes compute a navigation field (similar to [61]) to provide the robot with the *best possible* direction to reach the goal. The navigation problem is modeled as a Markov Decision Process and the robot state transitions depend on the current state and action. A value iteration algorithm is used to compute the best path including actions with the maximum utility.

Li et al. [42] formulated a guiding application as a robotics motion planning problem in the presence of obstacles. A protocol is described using artificial potential fields to plan the optimal paths that avoid the hazardous areas. The best moving direction depends on an artificial force balanced between an *attractive* potential (generated by the goal) that pulls the object to the goal, and a *repulsive* potential (generated by the obstacles) that pushes the object away from the goal. Chen et al. [14] proposed a distributed guiding navigation protocol for constructing *area-to-area* optimal guiding paths. The protocol partitions the WSN into areas using the Delaunay Triangulation method, and includes a reactive strategy to guide the object away from the dangerous zone along an area-to-area path based on the control messages sent by the WSN. In [64], Yao et al. studied a distributed path planning problem and proposed a solution framework called Distributed PRM (D-PRM), which includes four phases: (1) first, each sensor node perceives its environment and system connectivity in order to build its local roadmap; (2) second, the robot disseminates a task message to specify the goal and triggers the next distributed planning phase; (3) third, to find the shortest path based on distributed roadmaps, a distributed navigation field is created based on the distance to the goal and the best path is built by gradient

decent; (4) finally, the robot queries for the path and interacts with the WSN to reach the goal.

While the use of WSNs can lead to good navigation performance, it also poses new challenges in dealing with frequently updated sensor data to maintain the degree of awareness. To solve this, Bhattacharya et al.[7] proposed an approach that integrated a roadmap-based navigation algorithm for the mobile robots and a distributed *Roadmap Query* (RQ) protocol for the WSN, which saves communication cost by querying neighboring nodes of the mobile robot.

2.3 Localization

As described previously, one primary prerequisite for mobile robots to achieve autonomous navigation is its capability of estimating its position relative to the environment. Moreover, the Simultaneous Localization And Mapping (SLAM) problem, a thoroughly studied topic in robotics, studies whether a mobile robot can be placed in an unknown environment to build up a map and, meanwhile, use it to localize itself (refer to [23] for a thorough discussion on SLAM approaches). The localization problem for robots remains challenging since on-board sensor ranges are limited and visual-based techniques can become quite complex in dynamic environments. There are methods for both accurate indoor and outdoor localization, such as differential GPS or beacon systems. However, those are usually expensive, require the installation of previous infrastructure, or need a high computational load. Embedding WSNs in the unknown environment can help in solving the robot localization problem for several reasons: they are cheap, ubiquitous, easy to deploy and have few computational requirements.

The localization problem has also been inherently crucial for WSNs, since the location information annotates the place, at which the sensory data are collected or the events are detected. Thus, many solutions have been proposed in WSNs [43,29,63,2,54]. With these techniques, sensor nodes can localize themselves and provide their positions as well as the environment status. Due to the maturity of WSN localization techniques, recent robotics research has started to consider integrating WSNs for robot localization and SLAM. In [27], the authors proposed a map-free navigation solution based on *Radio Signal Strength Indicator* (RSSI) for an indoor mobile robot. The robot navigates autonomously by acquiring the information of RSSIs from a set of pre-installed sensor nodes. Then it estimates its distances to the sensor nodes based on the RSSI values, and uses triangulation to locate itself. In [39], the similar WSN localization technology was adopted to estimate the sensor-robot distances, which are then used to build a map for SLAM based on a two-step particle filter. In [22], the authors described a range-only SLAM approach with two integrated modalities: a mobile robot equipped with a sensor to measure its distance to the landmarks, and a set of sensor nodes, which act as landmarks and are able to measure the distance to other landmarks. It was demonstrated that both capabilities are complementary to solve SLAM with reduced drift in positioning and faster in terms of map-building. In [26], the sparse sensor data issue was considered, and a SLAM solution was developed based on RF WSNs. In [58], a multi-robot range-only

SLAM solution was proposed to solve the issue of map building with a team of autonomous robots and a WSN for urban search and rescue. The memory of the sensor nodes are used for map data exchange among mobile robots.

Even though RSSI measurements are noisy and sometimes inaccurate, there are different methods to alleviate this problem. For instance, sensor noises can be modeled [10], and techniques based on fingerprinting [31] reduce these kinds of effects, since RSSI measurements are compared with a previously computed RSSI map that already contains noisy information. Moreover, even when they are not accurate, WSN measurements can be helpful combined with other on-board sensors in order to discard multiple hypothesis or reset accumulative position errors such as those from odometry systems [11].

3 Robot-Assisted WSN Applications

3.1 Autonomous Deployment

Advances in self-configurable sensor techniques have made autonomous deployment possible. Using autonomous aerial or ground robots as WSN deployment tools is very advantageous, especially in scenarios where no fixed deployment is required, or the monitored areas are large and inaccessible. Mobile robots can help in achieving autonomous sensor deployment in several ways. First, a flying/ground robot can carry sensor nodes and drop them at specified locations. Second, a sensor-integrated robot can act as a mobile node and perform self-deployment by moving itself to the assigned location. In general, the autonomous deployment problem can be formalized as follows: given a deployment area A, the deployment algorithm outputs a set of coordinates, $L = \{(x_i, y_i, z_i)|i \leq N\}$, as the sensor deployment locations; where N is the number of the nodes to be deployed. Then, these coordinates are given to the mobile vehicles as waypoints that they need to visit in order to perform the deployment task.

The most flexible tools for the deployment of sensor nodes are flying robots since their movement is not limited by obstacles on the ground, and they can reach unattended places. In particular, rotary-wing UAVs are ideal when precise positioning of a sensor node is desired since they can hover over a point during the deployment (differential GPS systems can provide an accuracy of 10cm or better). In general, the payload capacity of helicopters is sufficient for carrying a small node, but more could be carried depending on the node's weight. For instance, a node designed for an underwater WSN will incorporate a watertight housing and an anchor, making it much heavier than a typical land-based node. Corke [17] et al. presented a UAV-assisted sensor deployment approach, which involves the development of the deployment platform for the helicopter and of the helicopter-sensor inter-communication. They use a gas-powered UAV [59] (with a PC-104 stack augmented with sensors) that flies autonomously based on a behavior-based control architecture. A set of waypoints is given to the helicopter controller, which makes the helicopter drive to each point, hover above it, and deploy a sensor. While aerial deployment of WSNs is cost-efficient, it also presents some unique challenges. One major issue is to ensure that air-dropped

nodes only fall within the area of interest. Kulkarni [40] et al. made use of vision-based approaches to solve the terrain recognition problem, which is formulated as a multidimensional optimization problem. The solution uses image segmentation techniques for autonomous aerial WSN deployment.

Additionally, control techniques for physical interaction between flying robots and the environment present new possibilities for the deployment and retrieval of sensor nodes. The first example is aerial manipulation [38], where a robotic manipulation device is attached to a UAV. This device can be used to deploy the nodes, retrieve them or even repair them. By establishing a contact between the manipulator and the node, a physical chain is created between the UAV, the manipulator, the node and the environment. In this situation, not only should the movement of the UAV be controlled but also the force interactions between all parts of the chain. The second example is load transportation, where a load is connected to the fuselage of the UAV by means of a rope. As it is shown in [6], even transportation and gentle deployment using multiple autonomous helicopters (in the case where the payload of one helicopter is not sufficient to lift a sensor node) is possible and can be demonstrated in field experiments. Similarly, the force interactions between a particular helicopter and the rope connecting this helicopter to the rest of the system should be controlled. Coordinating and controlling multiple aerial robots for the deployment of heavy sensors that are uneasy to be carried by one single UAV was also addressed in the AWARE project [48]. In particular, a platform was developed for autonomous self-deploying and operation of static and mobile WSNs cooperating with UAVs. The platform includes heterogeneous devices (e.g., ground camera nodes, mobile robots, low-powered sensor nodes, etc.) communicating through a publish/subscribe middleware and operating in the disaster management scenarios. AWARE also developed an automatic control system [44] that enabled load transportation using one or multiple UAVs and that was used in real flight experiments for WSN deployment.

In addition to aerial robots, ground robots have also been considered for WSN deployment, but most research has solved the deployment problem by introducing mobility to the traditional static sensor networks. In [20], the authors addressed autonomous node mobility by presenting Robomote [21], a robot platform functioning as a single mobile node in a mobile WSN. The primary design was to ease the WSN deployment and to reduce its associated cost. Robomote is compatible with the mote platform and consists of an Atmel 8535 microcontroller, two motors, a compass for heading and IR sensors. Moreover, an additional mote is added as the master node. Other similar examples for mobile sensor nodes integrated in robotic platforms include iRobotSense [52], Sensr [47], *Pioneer 3-AT* platforms in CONET testbed [34], and Garcia robots in TrueMobile testbed [35].

3.2 Sensing Coverage and Network Connectivity

Due to their limited battery energy and unreliable communication, traditional WSNs with static nodes often face the challenge of maintaining required sensing

coverage and network connectivity. Typical approaches often deploy redundant nodes to avoid coverage fractions and network partitions. However, recent research has investigated techniques to improve QoS in coverage and connectivity by exploiting the motion capabilities of robots. In [32], sensor deployment for coverage was viewed as a multi-robot exploration and mapping problem whose goal is to create a global map of the environment using one or more mobile robots to visit each location sequentially.

Normally, QoS in sensing coverage can be expressed as $QoS = S_c/S$, where S and S_c denote the *target* area and the *covered* area [41], respectively. S_c can be defined as the union of the coverage areas of static nodes and robots. Note that the sensing coverage of each node (or robot) is relative to its sensing range and its position. Since the location of a static node is fixed, the QoS can be expressed as a function of the positions of the mobile robots. Thus, the QoS in sensing coverage of the WSN can be improved by adjusting the position of mobile robots. In [5], the authors studied the dynamic sensor coverage problem in the absence of global localization information for a target detection scenario. In this scenario, to achieve coverage maximization, a Pioneer 2DX robot with a mote integrated relies on local instructions disseminated by static beacons (or local markers) and follows rules to *repel* each other and to avoid obstacles.

Several approaches are inspired by physics to enhance the initial deployment. In [66], a *virtual force* algorithm was proposed to maximize sensor field coverage using a combination of attractive and repulsive forces to decide on the path and the moving rate of the mobile robots; in [49], the authors proposed an approach based on fluid dynamics and modeled mobile sensor nodes as particles of compressible fluids to achieve effective coverage and scalable self-deployment; in [36], the presence of obstacles was considered and the movement of mobile robots was modeled using the theory of gas; in [32], a deployment approach based on potential field was proposed to maximize coverage area. Each sensor node is viewed as a virtual particle, subject to virtual forces, which repel other nodes as well as obstacles until the required network is deployed. In addition to the physics-based approaches, some approaches are based on computational geometry. In [62], the authors proposed two deployment protocols based on a local calculation of Voronoi diagrams to control the node movement and achieve target coverage.

Finally, some other approaches focus on utilizing mobile robots for repairing deployed WSNs. In [18], a token-based connectivity repair algorithm was developed. Each node initially broadcasts its network ID as a token and updates the token value with the greater received ID. Note that the tokens are only propagated within the connected area, and the disconnected areas will have a different token ID. Then, a robot helicopter [59] *sweeps* across the deployment area to collect the tokens in order to estimate the location of the connectivity hole and perform a recovery task. In [45], the authors studied sensor failure detection, and proposed a recovery algorithm that uses mobile robots to detect and replace the failed sensor. Moreover, they try to minimize the motion energy of the robots and the communication overhead. In [24], two coverage repair algorithms,

Randomized Robot-assisted Relocation of Static Sensors (R3S2) and a grid-based variant (G-R3S2), were proposed to improve coverage by using robots moving randomly to transfer redundant sensors to the sensing hole location within a minimal time delay. In the second algorithm, robot movements are limited to a virtual grid and are instructed by the neighboring sensors, which recommend the least visited grid points to increase the chance of discovering the redundant sensors and the sensing holes.

3.3 Data Collection

Data collection from static nodes using mobile robots has been deeply researched in the literature. According to the mobility patterns of collectors, methods can be classified in: random mobility and controlled mobility. In [51] the collectors, referred to as data MULEs, move randomly and collect data opportunistically from sensors within communication range. Sensor nodes, which are assumed static, wait for a MULE to be within range before starting communication. Then, the MULE collects data and moves to a different location. Moreover, the collector transfers the data to a base station when it enters its radio coverage. In case of controlled mobility, data are collected by vehicles whose trajectories pass near sensors, which know that trajectory in some cases. Based on the predicted data transfer times, static sensors can sleep during inactive periods in order to save energy [13]. In other schemes, the mobility of the collector is controlled so that it visits the deployed nodes avoiding sensor buffer overflows [57].

Data collection methods can also be classified according to the discovery method that allows static nodes to detect the presence of the collector. In scheduled *rendezvous* schemes sensor nodes and collector agree on specific times, at which they will be in contact [65]. In *on-demand* schemes, the static nodes can wake up as a result of a process initiated by the collector. Wake-up radios are widely used in these cases. Static nodes continuously monitor the wake-up radio channel and, as soon as they detect activity on the channel, they power up the data radio and start communicating with the collector.

Data collection using UAVs has been an attractive research topic. Some works have proposed theoretical and/or simulated analysis, architectures and protocols. For instance, a centralized WSN medium-access control for aerial platforms has been proposed in [46]; a middleware for the integration of WSNs and aerial vehicles was presented in [25]; and the work in [30] proposed a MAC protocol for data collection using a UAV with a directional antenna. In the simplest data collection approach, the deployed nodes gather and buffer the readings. When the UAV flies near the nodes, it sends a beacon packet and the nodes send the readings in reply. Experiments with the mentioned baseline approach were described in [55,60]. However, these schemes lack sufficient scalability to be applied in problems where hundred or thousands of nodes have been deployed.

Finally, in [15], the scalability of the baseline scheme is increased by grouping nodes. All nodes from one group send packets with readings in response to the UAV specific beacon for that group. The groups and their collection zones are computed taking into account nodes locations and radio coverage, among others.

4 Cooperating Mobile Robots And WSNs In PLANET

PLANET is an FP7 European IP project, which aims to provide an integrated platform that enables the deployment, operation and maintenance of heterogeneous networked cooperating objects, including mobile robots and wireless sensors. Figure 1 illustrates the PLANET integrated framework with cooperating objects (CO). We have also developed a middleware communication platform to achieve CO intercommunication and data collection. In this section, our discussion focuses on several aforementioned robot-WSN cooperating interactions in the real-life DBR wildlife monitoring application developed in PLANET regarding WSN deployment/recovery, remote environment monitoring and data collection using UAVs. More ongoing information (e.g., reports and experiment videos) about PLANET is available on the official website [50].

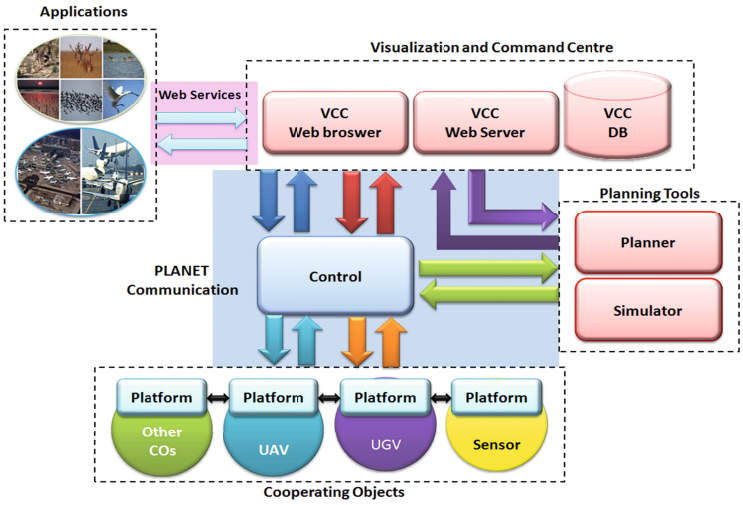


Fig. 1. The PLANET integrated framework

4.1 Network Deployment and Maintenance with UAVs

In PLANET, one of the DBR application scenarios, pollution monitoring, requires a WSN deployment in the Doñana National Park marshes (see Figure 2), which lie on the Guadalquivir River estuary on the Spanish Atlantic coast.

The intensive agriculture occupation has threatened the preservation of Doñana marshes water cycles. Calibrating the pollutant and sediment transport capabilities in these marshes requires a substantial effort and is not always feasible due to several reasons. First, the marshes are flooded during the rainy season and it is impossible to have a permanent installation of data collection stations. Second, accessing that area is currently only possible by horseback or using certain types of boats, which involves some risk for biologists. These problems are

Fig. 2. Doñana marshes

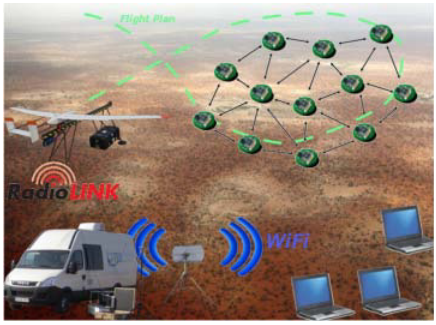


Fig. 3. WSN deployment using UAVs

tackled in PLANET by deploying a WSN using UAVs, as shown in Figure 3. The WSN deployment involves (1)a planning phase, in which locations for the sensor nodes as well as trajectories for the UAVs are provided by a planner [56], and (2) an operation phase, in which the UAVs will be sent to deploy sensor nodes at the positions specified by the planner, following the computed trajectories. In the following, we describe the specifications of the UAV platforms, fixed-wing (airplanes) and rotary-wing (helicopters), and the different methodologies used for deployment in PLANET.

PLANET Fixed-Wing UAV for WSN Deployment. For WSN deployment in DBR, the PLANET project uses the *Viewer* UAV, which has a communication range of 40km and a payload of 2kg. A device based on endless screw has been designed for sensor deployment. Figure 4 illustrates the Viewer and its integrated deployment devices. To determine the location where the sensor could land, a physical model of the node motion including air resistance is considered. Dealing with air resistance is very complex and in the general case leads to coupled nonlinear differential equations without closed analytic solutions. Instead, we solve them by using numerical integration methods on a case by case basis. Furthermore, in order to deploy a sensor node from a fixed-wing UAV, the first step is to command the UAV to go to the position where the node has to be deployed following a straight path, as depicted in Figure 5. When the UAV is pointing to the desired location for the deployment, it is commanded a constant velocity. Once the UAV is in the path, it is possible to calculate the actual

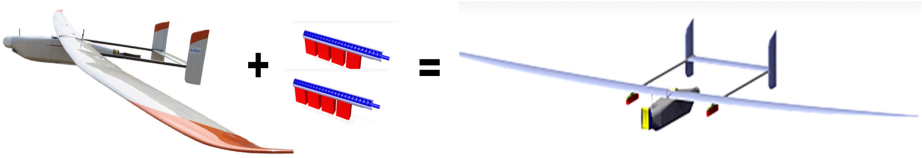


Fig. 4. The PLANET Viewer for WSN deployment in DBR

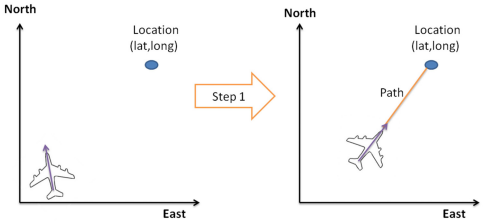


Fig. 5. First step of deployment

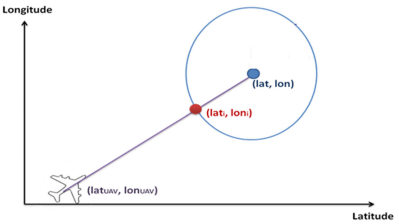


Fig. 6. Deployment position

position at which the node has to be dropped. The final deployment position (see Figure 6) will depend on several specifications that have to be defined: maximum impact velocity of the sensor when it reaches the ground (in order not to damage the sensor), minimum and maximum altitude of the UAV, and minimum and maximum velocity of the UAV.

PLANET Rotary-Wing UAV for WSN Deployment. In our pollution monitoring application, sensor nodes for water quality monitoring (see Figure 7) are deployed by a helicopter UAV whose hovering capability allows precise deployment. Each air-dropped sensor floats on the water and uses an anchor for positioning. An important issue is to plan the movements of the helicopter in advance taking into account the restrictions of the flight. Specially in sensitive areas like the Doñana marshes, electric propulsion of the flying system is preferred for acoustic reasons and for protecting the environment from the fuel. However, while the electric drive is rather efficient nowadays, it can not offer the endurance of a piston engine system. The integration of the control systems of the UAV with the PLANET framework allows commanding routes and monitoring status easily. Note that it is also possible to scale up the deployment system by adding several helicopters. The PLANET framework allows for the connection of several UAV platforms , which is important for a large-scale deployment.

Additionally, network maintenance can be performed by the UAVs in several ways. If temporary coverage of an area is needed, with an on-board sensor node, a UAV can be placed at a desired point in the air to recover functionality. If the UAV endurance is the main focus, fixed-wing airplanes are more suitable since they can achieve several hours of flight time. Otherwise, rotatory-wing UAVs can achieve better position accuracy. Moreover, for permanent recovery of a WSN it is possible to deploy additional nodes. A novel approach is to recover faulty equipment with the use of aerial manipulation technology [53] and to replace it. This is highly valuable for the marshes since man-made equipment is posing a potential threat to the wildlife.

Fig. 7. Design and components of the PLANET water sensor

4.2 Remote Environmental Monitoring

We address cooperation between mobile robots and static sensors in two PLANET scenarios where remote environment monitoring is performed by UAVs integrated with sensors or sampling devices: (1) pollution monitoring using a helicopter; and (2) aerial stratification of bats and insects.

Water Sampling Using a Sensor-Integrated Helicopter. In PLANET, pollution events at the DBR marshes are detected by a WSN deployed by UAVs as described previously. Pollution is detected by analyzing water parameters such as temperature, dissolved oxygen, conductivity PH, nitrate, heavy metals, phosphorus, etc. Moreover, UAVs with vertical take-off and landing capabilities can perform these measurements. The PLANET project uses the helicopter ADAM *(Advanced Demonstrator for Aerial Manipulation)*(See Figure 8), a research platform of DLR, to monitor pollution events by taking water samples and measuring their parameters. The technical features of ADAM include: a rotor diameter of 1.8m, a net weight of 9kg, a payload of 4kg and a flight duration of 15 minutes. In-situ measurements are possible carrying with the helicopter a sensor called Sonda and hovering over the desired position. Measurement data can directly be transmitted and displayed via the PLANET communication middleware and the visualization center.

While many values for environmental studies can be measured remotely by the sensors, others can only be measured by taking samples and analyzing them in a laboratory. A method for sampling water by an unmanned helicopter is proposed in PLANET. A pump as illustrated in Figure 9 is lowered into the water and activated until the desired amount of fluid is on-board within a sampling time of less than 20 seconds. After landing, the analysis is performed in a laboratory. The ADAM helicopter has already been used for flights in 2011 in DBR for sampling experiments. Depending on the needs of the application, the same systems used for ADAM could be integrated in different platforms in order to carry higher payloads or get longer endurance.

Capturing Insects Using a Sensor-Integrated Airplane. Although bats are the most diverse and numerous nocturnal insectivorous vertebrates, their role in ecosystem function by controlling arthropod densities and suppressing

Fig. 8. Helicopter ADAM in flight

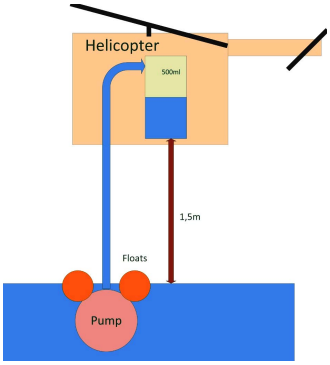


Fig. 9. Water sampling method

agricultural pests has been largely overlooked. Some studies conducted in North and Central America have attempted to quantify the economical impact of insectivorous bat populations on agricultural production, and estimated it to be enormous. These studies have combined radar techniques and insect sampling via traps mounted on helium balloons, but so far, samples have only been taken up to 200m above the ground. At present, appropriate tools to study the important ecological interaction between insects and bats at high altitudes are missing. The goal of this scenario is to develop for the first time a method to sample insects at heights up to 1000m above ground level, and to study the stratification in the aerosphere of bats and insects in different habitats. To achieve this, PLANET has tested a new sampling technique for aerial insects using traps integrated into UAVs. This method has some important advantages over the currently used sampling with helium balloons since the mobility of UAVs allows sampling different heights and habitats in a shorter timespan, and sampling height can be controlled much more accurately. In order to understand the causes of insect stratification at different heights, temperature and wind speed are measured simultaneously to insect collection. Most abundant insects will be identified to species or at least family level. Special attention will be given to potential agricultural pests. Figure 10 illustrates the capturing tool and a mosquito capturing experiment in Doñana.

4.3 Data Collection with UAVs

Assume a large-scale scenario in which a high number of WSN nodes have been deployed in inaccessible areas. The nodes cannot transmit to the base station using multi-hop routing channels. The objective is to use UAVs to collect readings from the nodes distributed in large areas.

We assume that the UAVs cannot have a complete coverage over all deployed nodes at every time: it is unrealistic with large scenarios as those considered in PLANET. Instead, the UAV trajectories should pass over the areas where the nodes have been deployed. In baseline methods the UAV communicates with each of the nodes deployed in the scenario. Scalability and lack of reactivity are the

Fig. 10. (a) Testing the capture tool; (b) Mosquito experiment in Doñana; (c) Captured mosquitoes

main disadvantages. Baseline schemes consider WSN and UAVs as independent units that do not influence each other, lacking flexibility to unexpected events such as node failures or changes in environmental conditions. The approach we follow in PLANET intends to exploit WSN-UAV cooperation to improve reactivity to changes and prolong the lifetime of the ground WSN nodes.

In the proposed method, static WSN nodes organize autonomously into clusters. A network formation stage is used to form clusters. Each cluster head candidate broadcasts a beacon with its remaining energy. Each node receives the beacons from the candidate and adheres to that with the highest energy by sending it a response packet. Each cluster has one cluster head (CH), who is responsible of organizing the packets interchange within the cluster, receiving measurements from all the cluster nodes and transmitting them to the UAV. Once the cluster has been formed, each cluster head creates a TDMA plan that contains static slots. The CH generates and distributes its TDMA plan in its cluster. Each node periodically gathers measurements and transmit them to the CH using its assigned TDMA slot. Each cluster member is only awake in slot from the cluster head and in his slot. It can be set in low-energy mode in inactive periods. The CH is responsible for aggregating the data from the non-CH nodes. When the CH receives the beacon packet from the UAV, it transmits to the UAV on-board node packets with its aggregated data. Figure 11 depicts the basic operation of the strategy for WSN data collection using a UAV.

The energy consumed by CH nodes is significantly higher than that consumed by follower nodes. Thus, the CH role is rotated when the remaining batteries of the current CH are below a certain value. When a CH detects that its remaining energy is below a threshold, it sends a packet announcing CH rotation. Candidates send the CH their remaining energy and the current CH selects the best candidate using criteria based on remaining energy and also considering data from the current UAV trajectory. The objective is to select a cluster head with high energy level and with a low change in the current UAV trajectory. In this problem, the coverage zones of the CHs define the zones for UAV data collection and the UAV should fly passing by these zones, we call them collection zones. The number of packets that the CH can transmit to the UAV depends on the size of the packets, the UAV speed and the size of these collection zones. The

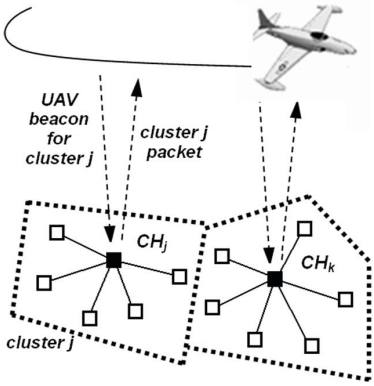


Fig. 11. Basic operation of the strategy for WSN data collection using a UAV

Fig. 12. Preliminary field experiments of WSN data collection using a UAV performed in October 2012

flight plan of the UAV should be updated periodically to cope with dynamic changes in the operation of WSN (CH rotation) and also changes in the UAV flight conditions (e.g. wind conditions). In our scheme, the operation of WSN (CH rotation) modifies the trajectory of the UAV and also, the UAV trajectory modifies the operation of WSN. The current UAV trajectory is considered in CH rotation when selecting candidate nodes to take the CH role.

Currently, the protocols and methods for the operation of the WSN and UAV flight plan computation, have been implemented and tested in the *CONET Robot-WSN Integrated Testbed* (`https://conet.us.es`) [1]. Some preliminary experiments in the field have also been carried out, see Figure 12. In performed experiments, collection zones had a width of 25-30 m and the UAV flew at 20-25 m/s, i.e. they had more than a second to interchange data, what was enough to transmit all the data gathered by the cluster. An average of 3-4 packets per cluster were received.

5 Conclusion

Many studies have shown that coupling mobile robots and WSNs can enrich application scenarios and enhance application performance. Sensor networks offer an efficient way for mobile robots to access physical data beyond their perceptual horizon. Conversely, the mobility of robots breaks the limitations of WSNs—with respect to target area inaccessibility, battery lifetime, hardware vulnerability, communication and computation capability—and can assist them to achieve higher performance. In this chapter, we first surveyed robotic-sensor applications with cooperating mobile robots and WSNs, and particularly put emphasis on the interactions of both in these applications. Then, we demonstrated the usefulness and feasibility of such interactions in several use cases of a real-life environment monitoring application developed in our PLANET project.

In summary, the benefits of integrating both technologies can be easily seen and have attracted a lot of attention in both robotics and WSN communities. We can anticipate more active discussion on issues related to cooperating robots and WSNs, and more advanced applications by adopting both technologies.

References

1. Jiménez-González, A., Martínez-de Dios, J.R., Ollero, A.: An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. Sensors 11(12), 11516–11543 (2011)
2. Bal, M., Liu, M., Shen, W., Ghenniwa, H.: Localization in cooperative wireless sensor networks: A review. In: 13th International Conference on Computer Supported Cooperative Work in Design, pp. 438–443 (2009)
3. Batalin, M., Sukhatme, G.: Using a sensor network for distributed multi-robot task allocation. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, vol. 1, pp. 158–164 (2004)
4. Batalin, M., Sukhatme, G., Hattig, M.: Mobile robot navigation using a sensor network. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 636–641 (2004)
5. Batalin, M.A., Sukhatme, G.S.: Sensor coverage using mobile robots and stationary nodes. In: Proc. SPIE, pp. 269–276 (2002)
6. Bernard, M., Kondak, K., Maza, I., Ollero, A.: Autonomous transportation and deployment with aerial robots for search and rescue missions. Journal of Field Robotics 28(6), 914–931 (2011)
7. Bhattacharya, S., Atay, N., Alankus, G., Lu, C., Bayazit, O., Roman, G.C.: Roadmap query for sensor network assisted navigation in dynamic environments. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 17–36. Springer, Heidelberg (2006)
8. Brscic, D., Sasaki, T., Hashimoto, H.: Acting in intelligent space - mobile robot control based on sensors distributed in space -. In: 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1–6 (2007)
9. McCormick, C., Laligand, P.-Y., Aghajan, H.L., Distributed, H.: agent control with self-localizing wireless image sensor networks. In: Proceedings of the Conference on COGnitive Systems with Interactive Sensors (COGIS 2006) (2006)
10. Caballero, F., Merino, L., Maza, I., Ollero, A.: A particle filter method for wireless sensor network localization with an aerial robot beacon. In: IEEE International Conference on Robotics and Automation, pp. 596–601 (2008)
11. Caballero, F., Merino, L., Ollero, A.: A general gaussian-mixture approach for range-only mapping using multiple hypotheses. In: IEEE International Conference on Robotics and Automation, pp. 4404–4409 (2010)
12. Cao, Y., Fukunaga, A., Kahng, A., Meng, F.: Cooperative mobile robotics: antecedents and directions. In: IEEE/RSJ International Conference on Intelligent Robots and Systems 1995. Human Robot Interaction and Cooperative Robots, vol. 1, pp. 226–234 (1995)
13. Chakrabarti, A., Sabharwal, A., Aazhang, B.: Using predictable observer mobility for power efficient design of sensor networks. In: Proceedings of the 2nd International Conference on Information Processing in Sensor Networks, pp. 129–145 (2003)
14. Chen, P.Y., Chen, W.T., Shen, Y.T.: A distributed area-based guiding navigation protocol for wireless sensor networks. In: IEEE International Conference on Parallel and Distributed Systems, pp. 647–654 (2008)

15. Cobano, J.A., Martínez-De Dios, J.R., Conde, R., Sánchez-Matamoros, J.M., Ollero, A.: Data retrieving from heterogeneous wireless sensor network nodes using UAVs. Journal of Intelligent Robotics Systems 60(1), 133–151 (2010)
16. Coltin, B., Veloso, M.: Mobile robot task allocation in hybrid wireless sensor networks. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2932–2937 (2010)
17. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In: Proceedings of ICRA, vol. 4, pp. 3602–3608 (2004)
18. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Deployment and connectivity repair of a sensor net with a flying robot. In: Ang Jr., M.H., Khatib, O. (eds.) Experimental Robotics IX. STAR, vol. 21, pp. 333–343. Springer, Heidelberg (2006)
19. Corke, P., Peterson, R., Rus, D.: Networked robots: Flying robot navigation using a sensor net. In: Dario, P., Chatila, R. (eds.) Robotics Research. STAR, vol. 15, pp. 234–243. Springer, Heidelberg (2005)
20. Dantu, K., Rahimi, M., Shah, H., Babel, S., Dhariwal, A., Sukhatme, G.S.: Robomote: enabling mobility in sensor networks. In: IPSN 2005 (2005)
21. Department of Computer Science, University of Southern California: Robomote, `http://www-robotics.usc.edu/~robomote`
22. Djugash, J., Singh, S., Kantor, G.A., Zhang, W.: Range-only slam for robots operating cooperatively with sensor networks. In: IEEE International Conference on Robotics and Automation, pp. 2078–2084 (May 2006)
23. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. IEEE Robotics Automation Magazine 13(2), 99–110 (2006)
24. Fletcher, G., Li, X., Nayak, A., Stojmenovic, I.: Randomized robot-assisted relocation of sensors for coverage repair in Wireless Sensor Networks. In: Vehicular Technology Conference Fall, pp. 1–5 (2010)
25. de Freitas, E.P., Ferreia, A.M., Pereira, C.E., Larsson, T.: Middleware support in unmanned aerial vehicles and wireless sensor networks for surveillance applications. In: Papadopoulos, G.A., Badica, C. (eds.) IDC 2009. SCI, vol. 237, pp. 289–296. Springer, Heidelberg (2009)
26. Fu, S.Y., Kuai, X.K., Zheng, R., Yang, G.S., Hou, Z.G.: Compressive sensing approach based mapping and localization for mobile robot in an indoor wireless sensor network. In: Networking, Sensing and Control (ICNSC 2010), pp. 122–127 (2010)
27. Fu, S., Hou, Z.G., Yang, G.: An indoor navigation system for autonomous mobile robot using wireless sensor network. In: International Conference on Networking, Sensing and Control, ICNSC 2009, pp. 227–232 (2009)
28. Gerkey, B., Mataric, M.: Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2003, vol. 3, pp. 3862–3868 (2003)
29. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: Proceedings of the Annual International Conference on Mobile Computing and Networking, pp. 81–95 (2003)
30. Ho, T.D., Park, J., Shimamoto, S.: Power and performance tradeoff of MAC protocol for wireless sensor network employing UAV. In: International Conference on Advanced Technologies for Communications, pp. 23–28 (2010)
31. Honkavirta, V., Perala, T., Ali-Loytty, S., Piche, R.: A comparative survey of WLAN location fingerprinting methods. In: Workshop on Positioning, Navigation and Communication, pp. 243–251 (2009)

32. Howard, A., Mataric, M., Sukhatme, G.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Distributed Autonomous Robotic Systems, vol. 5, pp. 299–308 (2002)
33. Janos, S., Martinovic, G.: Distant monitoring and control for mobile robots using wireless sensor network. In: Proceedings of the Conference CINTI (2009)
34. Jimenez-Gonzalez, A., Martinez-de Dios, J., Ollero, A.: An integrated testbed for heterogeneous mobile robots and other cooperating objects. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3327–3332 (2010)
35. Johnson, D., Stack, T., Fish, R., Flickinger, D., Ricci, R., Lepreau, J.: Truemobile: A mobile robotic wireless and sensor network testbed. In: The 25 th Annual Joint Conference of the IEEE Computer and Communications Societies (2006)
36. Kerr, W., Spears, D., Spears, W., Thayer, D.: Two formal gas models for multi-agent sweeping and obstacle avoidance. In: Proceedings of the Third International Conference on Formal Approaches to Agent-Based Systems, pp. 111–130 (2005)
37. Kogut, G., Blackburn, M., Everett, H.R.: Using video sensor networks to command and control ground vehicles. In: AUVSI Unmanned Systems in International Security (USIS 2003) (2003)
38. Kondak, K., Ollero, A., Maza, I., Krieger, K., Albu-Schaeffer, A., Schwarzbach, M., Laiacker, M.: Unmanned aerial systems physically interacting with the environment. load transportation, deployment and aerial manipulation. In: Handbook of Unmanned Aerial Vehicles. Springer (2012)
39. Kuai, X., Yang, K., Fu, S., Zheng, R., Yang, G.: Simultaneous localization and mapping (slam) for indoor autonomous mobile robot navigation in wireless sensor networks. In: Networking, Sensing and Control (ICNSC 2010), pp. 128–132 (2010)
40. Kulkarni, R., Venayagamoorthy, G.: Bio-inspired algorithms for autonomous deployment and localization of sensor nodes. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40(6), 663–675 (2010)
41. Li, J., Li, K., Wei, Z.: Improving sensing coverage of wireless sensor networks by employing mobile robots. In: IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 899–903 (2007)
42. Li, Q., De Rosa, M., Rus, D.: Distributed algorithms for guiding navigation across a sensor network. In: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, pp. 313–325 (2003)
43. Mao, G., Fidan, B., Anderson, B.D.O.: Wireless sensor network localization techniques. Comput. Netw. 51(10), 2529–2553 (2007)
44. Maza, I., Kondak, K., Bernard, M., Ollero, A.: Multi-uav cooperation and control for load transportation and deployment. Journal Intelligent and Robotics Systems 57(1-4), 417–449 (2010)
45. Mei, Y., Xian, C., Das, S., Hu, Y.C., Lu, Y.H.: Sensor replacement using mobile robots. Computer Communications 30(13), 2615–2626 (2007)
46. Mitchell, P.D., Qiu, J., Li, H., Grace, D.: Use of aerial platforms for energy efficient medium access control in wireless sensor networks. Comput. Commun. 33(4), 500–512 (2010)
47. Monash University. Department of Electrical and Computer Systems Engineering: Sens-r, `http://ctieware.eng.monash.edu.au/twiki/bin/view/WSensornets/Sens-r`
48. Ollero, A., Bernard, M., La Civita, M., Van Hoesel, L., Marron, P., Lepley, J., de Andres, E.: Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles. In: Safety, Security and Rescue Robotics (SSRR), pp. 1–6 (2007)

49. Pac, M., Erkmen, A., Erkmen, I.: Scalable self-deployment of mobile sensor networks: A fluid dynamics approach. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1446–1451 (2006)
50. PLANET project, http://www.plant-ict.edu
51. Shah, R.C., Roy, S., Brunette, S.J., Data, W.: mules: Modelling a three-tier architecture for sparse sensor networks. In: Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications, pp. 30–41 (2003)
52. Janansefat, S., Senturk, I., Akkaya, K., Gloff, M.: A mobile sensor network testbed using irobots. In: 37th Annual IEEE Conference on Local Computer Networks (LCN) (2012)
53. Schwarzbach, M., Kondak, K., Laiacker, M., Shih, C.Y., Marron, P.J.: Helicopter UAV systems for in situ measurements and sensor placement. In: Geoscience and Remote Sensing Symposium (IGARSS) (2012)
54. Shih, C.Y., Marrón, P.: COLA: Complexity-reduced trilateration approach for 3D localization in wireless sensor networks. In: Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 24–32 (2010)
55. Teh, S.K., Mejias, L., Hu, P.C., Experiments, W.: in integrating autonomous uninhabited aerial vehicles (UAVs) and Wireless Sensor Networks. In: Proc. of the Australasian Conference on Robotics and Automation (2008)
56. Soleymani-Fard, R., Shih, C.Y., Baudewig, M.: J.Marron, P.: Coplanner: A wireless sensor network deployment planning architecture using unmanned vehicles as deployment tools. In: SENSORCOMM, pp. 73–76 (2012)
57. Somasundara, A.A., Ramamoorthy, A., Srivastava, M.B.: Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In: IEEE International Real-Time Systems Symposium, pp. 296–305 (2004)
58. Sun, D., Kleiner, A., Wendt, T.M.: Multi-robot Range-Only SLAM by Active Sensor Nodes for Urban Search and Rescue. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 318–330. Springer, Heidelberg (2009)
59. USC Autonomous flying vehicle project: Avatar, http://www.robotics.usc.edu/~avatar
60. Valente, J., Sanz, D., Barrientos, A., del Cerro, J., Ribeiro, A., Rossi, C.: An air-ground wireless sensor network for crop monitoring. Sensors 11(6), 6088–6108 (2011)
61. Verma, A., Sawant, H., Tan, J.: Selection and navigation of mobile sensor nodes using a sensor network. Pervasive and Mobile Computing 2(1), 65–84 (2006)
62. Wang, G., Cao, G., Porta, T.L.: Movement-assisted sensor deployment. IEEE Transactions on Mobile Computing 5(6), 640–652 (2006)
63. Wang, X., Bischoff, O., Laur, R., Paul, S.: WSN-based visual object tracking using extended information filters. In: Procedia Chemistry, pp. 461–464 (2009)
64. Yao, Z., Gupta, K.: Distributed roadmaps for robot navigation in sensor networks. IEEE Transactions on Robotics 27(5), 997–1004 (2011)
65. Zhang, P., Sadler, C.M., Lyon, S.A., Martonosi, M.: Hardware design experiences in zebranet. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 227–238 (2004)
66. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, INFOCOM, vol. 2, pp. 1293–1303 (2003)

Collaborative Autonomous Surveys in Marine Environments Affected by Oil Spills

Shayok Mukhopadhyay[1], Chuanfeng Wang[2], Mark Patterson[3], Michael Malisoff[4], and Fumin Zhang[1]

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30322
[2] George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30322
[3] Marine Science Center, Northeastern University, 430 Nahant Road, Nahant, MA 01908
[4] Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803
{shayok,cwang329,fumin}@gatech.edu, m.patterson@neu.edu, malisoff@lsu.edu

Abstract. This chapter presents results on collaborative autonomous surveys using a fleet of heterogeneous autonomous robotic vehicles in marine environments affected by oil spills. The methods used for the surveys are based on a class of path following controllers with mathematically proven convergence and robustness. Use of such controllers enables easy mission planning for autonomous marine surveys where the paths consist of lines and curves. The control algorithm uses simple dynamic models and simple control laws and thus enables quick deployment of a fleet of autonomous vehicles to collaboratively survey large areas. This enables using a mobile network to survey an area where the different member nodes may have slightly different capabilities. A mapping algorithm used to reconcile data from heterogeneous marine vehicles on multiple different paths is also presented. Vehicles with heterogeneous dynamics are thus used to aid in the reconstruction of a time varying field. The algorithms used were tested, mainly on student-built marine robots that collaboratively surveyed a coastal lagoon in Grand Isle, Louisiana that was polluted by crude oil during the Deepwater Horizon oil spill. The results obtained from these experiments show the effectiveness of the proposed methods for oil spill surveys and also provide guidance for mission designs for future collaborative autonomous environmental surveys.

1 Introduction

Marine surveys are crucial for assessing the effects of maritime disasters, like the Deepwater Horizon oil spill that occurred in 2010. The oil spill spurred research that developed tools and technology to effectively handle such catastrophes in future.

Autonomous surveys are especially attractive in situations where the marine environment is less than ideal for human-based methods. During an autonomous

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*, Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_5,

survey, marine robots must be able to move along a desired path in order to gather sensor data along that path. As a result, path following [1,2,3] for marine vehicles is very important and therefore has been widely studied [4,5,6,7]. Although theoretical work for path following has advanced [8,9], only few theoretical results have been evaluated in field tests [10]. The path following controller used in this work is based on the Frenet-Serret framework [11]. The robustness of the control law has been theoretically justified in [9]. Satisfactory performance of such controllers has been observed with mobile robots [12,13] and ocean gliders [12,14], hence we are motivated to use such controllers with marine robots for collaborative oil spill surveys.

Multi-vehicle swarms with formation controllers [15,16] can also be used for surveys covering large areas. Researchers have produced a wealth of literature concerning solutions to various problems in the field of multi-agent robotics, and their respective applications in real life [17,18]. Literature related to envirnomental mapping (in scenarios like responses to oil spills [19,20]) using multiple underwater vehicles also exists [21,22]. However, it may not always be possible to obtain a fleet consisting of identical robotic vehicles for large on-field surveys. This further motivates this chapter, as the aim of this work is to use a single type of control algorithm for controlling the motion of heterogeneous robots in a real-life marine environment to perform a collaborative survey. In this setting, the sensors on board the different vehicles, which provide sensor data remotely, form a distributed sensor network. Note that unlike results on multi-agent control as in [23], the work presented in this chapter does not present cooperative control, or mapping laws for an entire group of robots. Instead, this chapter focuses on making robots perform individual operations satisfactorily over a shared communication network. The results presented in this chapter are based on combining the information obtained by the robots. This approach is used because it is not necessarily easy, or always possible, to use a single control algorithm for a fleet of heterogenous robotic vehicles. Therefore, the techniques used to reconcile data collected by heterogenous vehicles operating over a shared communication network forms a collaborative survey framework.

This chapter extends the work presented in [24]. It demonstrates simple and effective methods for allowing heterogeneous autonomous vehicles to collaboratively carry out marine surveys on a large area and to reconcile data from various sources to produce required representations. Despite using a simple unicycle model, the controllers enable the vehicles to track lines and curves reliably in the presence of natural disturbances such as wind, water currents, and engineering limitations like sensor inaccuracy, localization errors, and communication delays. This enables the reconstruction of a time varying field surveyed along specific paths of interest, even if the dynamics of the vehicles used differ to a certain extent. Survey data are collected by the vehicles during multiple autonomous missions. Although the dynamics of the different vehicles used in the survey efforts differ greatly, they are used to collaboratively survey a lagoon surface. Once the underwater vehicles find an area of interest, the surface vehicles focus on this area to produce a map of oil concentration. A mapping algorithm is then used

to reconcile data obtained from multiple robotic vehicles. The approach used for mapping in this chapter is inspired by the work of the authors of [25]. In [25], the tidal flow around gliders is approximated by a series of temporal basis functions and spatial basis functions. The creation of a map of oil concentration can benefit from the strategy used in [25] because survey data is collected by the autonomous marine vehicles over a period of multiple days. Therefore, any map-making algorithm must help compensate for the effects of missing data and time varying effects like tides and currents. For the lagoon surveyed in Grand Isle, Louisiana the changes in depth and oil concentration are sufficiently slow due to moderate effects of tides and currents. Hence the approach to bathymetry and oil concentration mapping in this chapter is mostly similar to [25], except the time varying components are neglected and only spatial basis functions are used. This enables the generation of maps that capture the bathymetric features and surface oil concentration in a region of interest, even if data are not available at all locations.

A twenty-one day long survey was performed in July 2011. A coastal lagoon in Grand Isle, Louisiana was selected for the survey, because heavy pollution had been reported there during the Deepwater Horizon oil spill. Large scale cleaning efforts had been performed in Grand Isle after the spill was contained. Using a fleet of heterogeneous marine vehicles, a large amount of data was collected to evaluate the surface concentration of oil one year after the cleanup efforts. This chapter presents analysis of the survey data, experimental results on parameter identification, and path following control. On the basis of the data analysis, the main contribution of this chapter is showing that the control and mapping algorithms/procedures presented in this chapter are effective for carrying out collaborative surveys employing heterogenous marine vehicles. This chapter is organized as follows. Section 2 describes the hardware and software systems for all the vehicles used in the collaborative survey and explains the dynamic model used to describe vehicle motion. The control laws and mapping algorithms are presented in Section 3, followed by experimental results and data analysis in Section 4. Section 5 provides the conclusion.

2 Marine Robots

Three vehicles were employed in the survey efforts, namely: a student-built Autonomous Surface Vehicle (ASV) called Victoria, a student-built Remotely Operated Vehicle (ROV) called β, and Autonomous Underwater Vehicles (AUVs) called Fetch1, and the EcoMapper (Figure 1). Victoria and Fetch1 played a main role in the survey, while the EcoMapper and ROV-β were collaborating in an auxiliary role. This section describes the hardware and software onboard each vehicle and the various mathematical models used.

2.1 ASV-Victoria

Hardware: ASV-Victoria appears in Figure 1, middle row, lefthand image. It was developed and built by a student robotics team named Georgia Tech

Fig. 1. Marine vehicles used in the survey. From top to bottom: AUV-Fetch1, ASV-Victoria (left), ROV-β (right), and the AUV-EcoMapper.

Savannah Robotics. It is now maintained by students from the Georgia Tech Systems Research (GTSR) group. It weighs 50kg, and is approximately 1m long and 0.75m wide. The overall height is 0.75m. Victoria has a twin-hull catamaran design with hulls composed of multiple layers of fiberglass sheets. The twin-hull catamaran design allows for a smooth ride since the turbulence in the center of the boat is reduced. The hull spacing is optimized, providing high stability and load carrying capacity. Victoria uses a specific layout for electronics, propulsion and power systems inside the hulls. This reduces pitching due to sudden changes in acceleration.

The electronic equipment onboard Victoria can be classified into computational units, sensors, actuators and communication systems. Figure 2 shows a high level view of Victoria's electrical systems. Victoria houses two separate computational units. One supports navigation and vision systems. The other computational unit supports lower level thruster control, and data acquisition from all other sensors onboard. Each computational unit is a Compact RIO (cRIO) produced by National Instruments. Here RIO is an abbreviation of reconfigurable input/output. The cRIOs are chosen because they combine an embedded real-time processor, a Field Programmable Gate Array (FPGA), and I/O modules. This increases reliability and speed of operation and makes it easy to

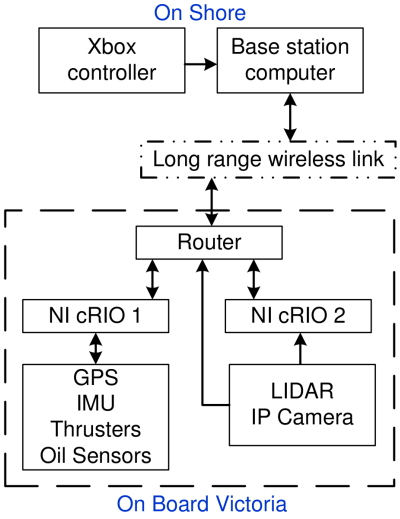


Fig. 2. A high level schematic of Victoria's electrical systems

swap out onboard sensors depending on survey requirements. Two thrusters from CrustCrawler Robotics are mounted on the hulls. They form the main actuation units and are capable of producing up to 60lb of thrust. As a result, Victoria's maximum linear speed is 2m/s. A MicroStrain 3DM-GX1 Inertial Motion Unit (IMU), an Ethernet camera, a Cyclops-7 oil sensor made by Turner Designs, and a Garmin 16x GPS receiver are the main sensors onboard. A long range wireless link forms the backbone of our communication system and enables remote operation from shore. The wireless communication setup includes a Ubiquiti Rocket-M5 base station on shore, a Ubiquiti Bullet-M5 access point, and an Ethernet router onboard Victoria. The vehicle can work in both autonomous mode and remote control mode. In remote control mode, the vehicle can be remotely operated within a range of approximately 500m. An Xbox controller is used to send commands to Victoria during remote operation. An intuitive joystick-based controller coupled with video from Victoria's onboard cameras contribute greatly to the ease of operation. In autonomous mode, a curve tracking controller enables Victoria to track a specified path without human intervention.

Software: National Instruments (NI) LabVIEW is used onboard ASV-Victoria. A high level schematic of Victoria's software architecture is shown in Figure 3. The software is composed of the following main virtual instruments (VIs): *Main PC*, *Cooperative control*, *Main RIO*, and *Main FPGA*. The VI *Main PC* runs on the control laptop. It retrieves data from VIs running on the cRIOs onboard Victoria and displays the data on the control laptop on-shore. The VI *Main PC* also receives commands from the Xbox controller and sends them to Victoria. This allows user-friendly and easy remote operation. The *Cooperative control*

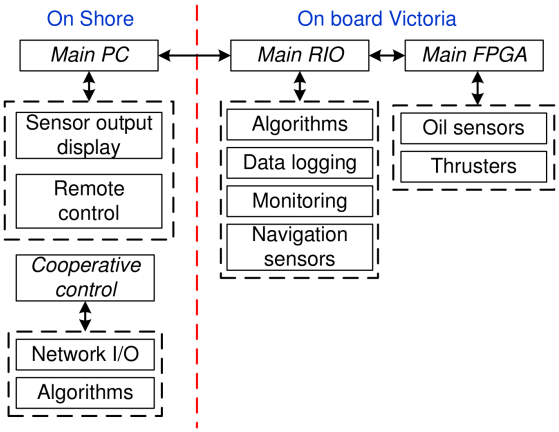


Fig. 3. A high level schematic of Victoria's software architecture

VI runs on the control laptop and communicates over a network with other autonomous vehicles. The algorithms sub-VIs under the *Cooperative control* VI receive position and orientation data from other autonomous vehicles and send appropriate control commands that enable these vehicles to track desired curves. This makes it easy to enlarge the size of a survey fleet by adding other vehicles which can communicate with the control laptop wirelessly. Adding a new *Cooperative control* VI to the main computer program on the control laptop allows a newly added survey vehicle to receive specific commands related to a collaborative survey.

The VIs *Main RIO* and *Main FPGA* run on cRIOs onboard Victoria. The VI *Main RIO* has many sub-VIs. Based on their functions, they can be classified into sub-VIs dealing with algorithms, data logging, monitoring, and sensing. The algorithms sub-VIs perform calculations allowing the vehicle to operate in different states, e.g. GPS waypoint navigation and autonomous curve tracking. The data logging VIs record data from the sensing devices into Victoria's onboard memory. The monitoring VIs help monitor critical parameters like CPU temperature, CPU load and execution speed. This helps to maintain required operating conditions for the computer system. The navigation sensor VIs query navigation equipment like the GPS and LIDAR and provide data to other VIs such as algorithms and logging VIs that request such data. Simple lower level sensing and control tasks are handled by the *Main FPGA* VI running on an FPGA module in a cRIO. This VI handles data acquisition from the oil sensors, receives control commands from the algorithms VIs and converts them into appropriate commands for the thrusters. The oil sensors and thrusters are dedicated resources used for every mission. Hence they are architecturally separated to ensure that system-wide code changes do not result in stray bugs that could affect these vital systems. This improves system reliability.

2.2 Fetch1

Hardware: Fetch1 is seen in Figure 1, top row. Fetch1 was developed by Mark Patterson and James Sias and can be used either as an ASV or AUV [26,27,28]. It has a maximum diving depth of 330m. It served as an ASV for some of the experiments in our collaborative survey efforts. Fetch1 is aluminum hulled. It weighs 220lb and is 6.5ft long. It is driven by a single propeller, and steered by two pairs of single-degree-of-freedom control surfaces. Fetch1 is outfitted with Wi-Fi as well as a FreeWave RF serial modem that maintains constant contact with a shore station as long as the vehicle is on the surface. Fetch1 uses an assortment of sensors including a Garmin WAAS GPS, a Precision Navigation TCM2 Compass/pitch/roll sensor, an Omega pressure sensor for depth, a Raytheon 220kHz sonar altimeter, Teledyne conductivity and temperature sensors, and a Turner Cyclops-7 crude oil sensor.

Software: Fetch1's main flight computer also runs LabVIEW. For all experiments mentioned in this chapter, Fetch1 was operated in teleoperation mode where Victoria's shore-side control computer sent commands to Fetch1. This enabled the two vehicles to collaboratively survey the lagoon in Grand Isle, Louisiana. The communication setup is illustrated in Figure 4. Feedback control algorithms operated at a frequency of 0.5Hz with data exchange every two seconds. This is a limitation imposed by the operating speed of the state machine loop in the flight computer onboard Fetch1. To deal with the imposed delay, we use buffers known as virtual states on the ToughBook and the control laptop. The quantities marked using * in Figure 4 indicate commanded values sent to Fetch1. Initial telemetry requests and values received from Fetch1 are not marked using *. The GPS positions and control-surface angles are periodically received from Fetch1 and stored in a virtual state on the ToughBook. All commands from the control laptop are also stored in this virtual state, so that they can be forwarded to the main flight computer onboard Fetch1 when a transmission is relayed to Fetch1. On the control laptop a virtual state is maintained and updated every time the ToughBook is polled for data and every time a new command is generated by the control algorithm. Therefore, the control algorithm performs its calculations using data that are at most two seconds old. The communication framework described here forms a sensor/control network. This sensor/control network is the backbone of our collaborative survey efforts because this allows us to easily use the same control law to control both Victoria and Fetch1 over the network and make measurements remotely using a variety of sensors mounted on board the different vehicles.

2.3 ROV-β

Hardware: The ROV-β, shown on the right in the second row of Figure 1, is a remotely-operated underwater vehicle built by GTSR. It is 36.45in in length, 22.5in in width and 18.25in in height. It weighs 125lb and can dive to a depth

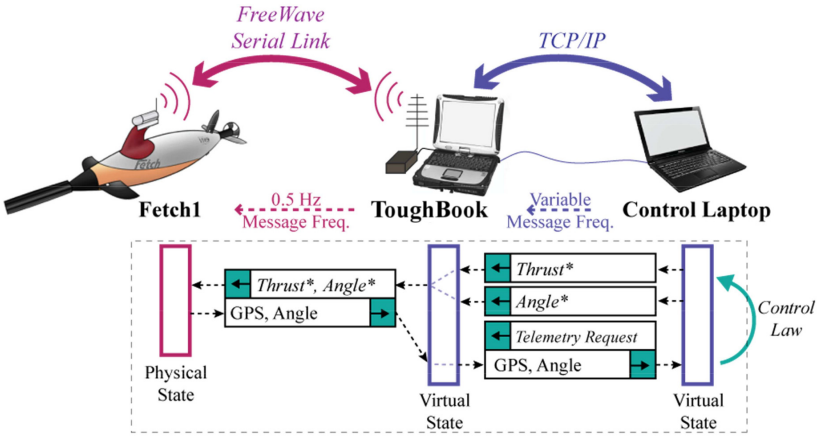


Fig. 4. Communicating with the Fetch

of 330ft. The sensors onboard can sample oil, measure depth, and acquire visual data. A system for collecting water samples and a pneumatically powered manipulator are also housed onboard. The major components of ROV-β are the pressure vessel, frame, propulsion system, buoyancy system, manipulator, and power and control systems. The body is built of aluminum and carbon steel. Black Rust-Oleum paint is used to protect the carbon steel from corrosion. The cylindrical pressure vessel has a volume of 160in^3 and is rated to a depth of 500ft. The pressure vessel has two removable end caps, one on each end of the cylinder. Each cap houses SEACON connectors for through-hull electrical connections. The cap on the front end has a transparent acrylic dome which forms the viewport for the primary camera. The carbon steel frame is welded directly to the pressure vessel. The frame's unique design protects the thrusters from collisions and provides a surface for mounting external subsystems like actuators and sampling tubes. The propulsion system includes four oil-compensated thrusters made by CrustCrawler Robotics. Each thruster is capable of producing a thrust of 25lb. The buoyancy system is made of syntactic foam and mounted atop a rack on the frame to offset the negative buoyancy of the ROV. This places the center of buoyancy above the center of gravity, making the ROV more stable. ROV-β has a pneumatic manipulator to grab articles of interest while underwater. Power for the ROV is supplied from the surface by two deep-cycle lead acid batteries. The control system consists of a cRIO, a router, and an Xbox controller. The cRIO is primarily used to send thruster commands and record sensor data. An onboard network switch provides connectivity between the ROV and the shore via a single Cat5e cable. The onshore wireless router allows the pilot to operate away from the point of launch. ROV-β improves the effectiveness of a survey team because of its capability to go underwater at points of interest and collect samples. This is a key role for the particular collaborative survey conducted in Grand Isle, Louisiana.

Software: Similar to ASV-Victoria, NI-LabVIEW is used onboard ROV-β. The software for ROV-β is a simplified version of the software for ASV-Victoria, because ROV-β is remotely controlled and does not require autonomous control and navigation algorithms. We skip the explicit discussion of software for ROV-β, as it is similar to ASV-Victoria's software.

2.4 EcoMapper

Hardware: The EcoMapper is seen in the bottom row in Figure 1. It is a commercial AUV used for environmental mapping, made by YSI Inc. It weighs 45lb and is 152cm in length from bow to the stern. The diameter of the hull is 14.7cm. It features four independent control planes and a two-bladed propeller. The EcoMapper is rated for a depth of 220ft. It can attain a maximum speed of 4 knots. Onboard it uses a computer with a processor from the x86 genre and magnetic storage for saving survey data. It communicates with a shore station via a 802.11g Wi-Fi link. The EcoMapper's navigation system consists of a GPS for surface operations and a Doppler Velocity Log (DVL) for underwater operations. The nose cone houses the sensors used for surveys. The EcoMapper is equipped with conductivity and temperature sensors, a three-axis digital compass, a depth sensor (measuring vehicle depth from water surface), and the depth-sounding sonar (measuring vehicle height from the bottom). It is powered by rechargeable Li-ion batteries rated at 600Wh, which last for about 8 hours at an operating speed of 2.5 knots. The EcoMapper makes it very easy to generate bathymetry maps. This greatly facilitates mission designs for other autonomous marine vehicles. The EcoMapper thus had a fundamental role in this collaborative survey effort.

Software: The EcoMapper is operated using a software program called Underwater Vehicle Console (UVC). It operates under Windows XP and can be remotely accessed over a Wi-Fi connection. The EcoMapper can be operated in both manual and autonomous modes. When the EcoMapper is on the surface and within Wi-fi range, it can be driven manually. In manual mode, the EcoMapper's status and sensor readings are displayed on the UVC screen but not recorded by the EcoMapper. In autonomous mode, the EcoMapper follows a predefined course either on or below surface, and does not require assistance from the human user. During missions, the EcoMapper acquires information regarding position (latitude and longitude), velocity, pitch, roll, depth from surface, height to bottom, water temperature, speed of sound and water salinity using sensors mounted in the nose cone. This data is saved to log files for use later.

2.5 Mathematical Models for the Vehicles

The unicycle model shown below is widely used to model the kinematics of robots as point particles.

$$\dot{x} = v\cos\theta \tag{1}$$

$$\dot{y} = v\sin\theta \tag{2}$$

$$\dot{\theta} = \omega \tag{3}$$

In Equations (1)-(3), variables x and y represent the position of the robot, θ is the orientation of the robot, v is the linear speed, and ω is the angular velocity. The above model describes both Victoria and Fetch1. For Victoria, we perform more detailed analysis for estimating the model parameters. The linear and angular velocities v and ω in (1)-(3) can be written in terms of the velocities of the left and right thrusters (v_l, v_r) on ASV-Victoria as

$$v = \frac{v_l + v_r}{2} \tag{4}$$

$$\omega = \frac{v_r - v_l}{2l} \tag{5}$$

$$v_l = K_1 n_l,\ v_r\ =\ K_2 n_r \tag{6}$$

where l is the distance between the horizontal axis of the vehicle and the horizontal axis passing through the center of a thruster. The quantities n_l and n_r represent the duty ratios of the signals sent to the left and right thrusters respectively. The constants K_1 and K_2 relate the duty ratios to the velocities v_l and v_r. Now substituting v_l and v_r from (6) into (4) and (5) to find v and ω and then rewriting (1)-(3), we obtain the following modified unicycle model:

$$\dot{x} = \frac{K_1 n_r + K_2 n_l}{2}\cos\theta \tag{7}$$

$$\dot{y} = \frac{K_1 n_r + K_2 n_l}{2}\sin\theta \tag{8}$$

$$\dot{\theta} = \frac{K_2 n_r - K_1 n_l}{2l} \tag{9}$$

Simple algorithms have been developed to estimate the parameters K_1 and K_2 from experimental data. Note that the control algorithms used for marine surveys rely on the models shown above. These models are simplistic and do not explicitly account for lateral drift or any other disturbance, yet the controllers are able to produce satisfactory results. Using such simple models aids code design, and makes the software used to drive our vehicles simpler and more robust. Using controllers for such simple models can aid multi-vehicle collaboration as this model does not assume any particular type of marine vehicle.

3 Identification, Control and Mapping Algorithms

A variety of algorithms have been implemented on ASV-Victoria to enable autonomy. Heading following and GPS-waypoint navigation are the most basic

forms of autonomy algorithms onboard Victoria. The curve tracking algorithm forms the focus of the discussion in this section since it is used extensively for the marine surveys presented in this chapter. The performance of the curve tracking algorithms can be improved if the parameters of the model in (7)-(9) are identified. The identification of model parameters is achieved as follows.

3.1 Parameter Extraction

The parameters K_1 and K_2 used in the model Equations (7)-(9) can be obtained from measurements of the linear velocity v and angular velocity ω as

$$K_1 = \frac{v - l\omega}{n_l} \tag{10}$$

$$K_2 = \frac{v + l\omega}{n_r}. \tag{11}$$

Open loop tests are performed to identify these parameters. In the open loop tests, the thruster commands n_r and n_l stay constant for each particular test. The actual values of v and ω can be estimated from GPS data. Then we can calculate K_1 and K_2 using Equations (10) and (11).

3.2 Curve Tracking

The model in (1)-(3) is suitable for performing simple motions using differential drive vehicles like ASV-Victoria. Converting the model in (1)-(3) to the Frenet-Serret framework simplifies curve tracking control design [11] as follows. Let $\mathbf{r} = [x, y]^T$ be the position vector. Define vectors $\mathbf{x}$ and $\mathbf{y}$ as $\mathbf{x} = [\cos\theta, \sin\theta]^T$ and $\mathbf{y} = [-\sin\theta, \cos\theta]^T$. Notice that $\mathbf{x}$ and $\mathbf{y}$ are unit vectors and are orthogonal, i.e. $\mathbf{x} \cdot \mathbf{y} = 0$. From the above definitions we can write,

$$\dot{\mathbf{r}} = v\mathbf{x} \tag{12}$$

$$\dot{\mathbf{x}} = vu\mathbf{y} \tag{13}$$

$$\dot{\mathbf{y}} = -vu\mathbf{x} \tag{14}$$

where $u = \omega/v$. Note that Equation (12) is equivalent (in vector form) to Equations (1) and (2). It must also be noted that Equations (12), (13) and (14) can be used to describe planar curves. Let us imagine a robot moving along a smooth planar curve with velocity v. In this setting, vector $\mathbf{x}$ becomes the tangent vector as it is always tangent to the curve. Vector $\mathbf{y}$ is known as the normal vector as it is perpendicular to $\mathbf{x}$. Usually when a curve is concerned, the symbol k is used in place of u and the speed $v = 1$.

Using the framework described above, the curve tracking problem is formulated as follows. Let $\mathbf{r}_2$ denote the position vector of the robot and $\mathbf{r}_1$ denote the position of the the closest point on a curve with respect to the robot. Figure 5 shows two particles, the robot (indicated by quantities having subscript 2) and its projection along the curve which the robot is trying to track (indicated by

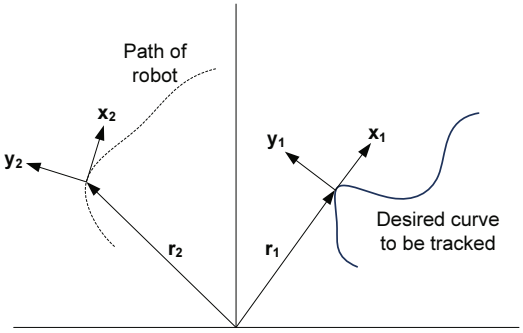


Fig. 5. Curve tracking using two Frenet-Serret frames

quantities having subscript 1). Mathematically, the dynamics are written using Frenet-Serret equations as follows:

$$\dot{\mathbf{r}}_1 = v_1 \mathbf{x}_1 \tag{15}$$
$$\dot{\mathbf{x}}_1 = \mathbf{y}_1 v_1 k_1 \tag{16}$$
$$\dot{\mathbf{y}}_1 = -\mathbf{x}_1 v_1 k_1 \tag{17}$$
$$\dot{\mathbf{r}}_2 = \mathbf{x}_2 \tag{18}$$
$$\dot{\mathbf{x}}_2 = \mathbf{y}_2 u_2 \tag{19}$$
$$\dot{\mathbf{y}}_2 = -\mathbf{x}_2 u_2. \tag{20}$$

In Equations (16) and (17), k_1 represents the algebraic curvature of the curve that the robot is trying to track. The speed v_2 of the robot is assumed to equal unity, so it does not appear in Equations (18), (19) and (20).

Define the vector $\mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1$ as the difference in position between the robot and its projection on the curve, so $\rho = \|\mathbf{r}\|$ (is the relative distance). Define ϕ as the relative bearing between $\mathbf{x}_1$ and $\mathbf{x}_2$. The variable ϕ satisfies the relations $\cos\phi = \mathbf{x}_1 \cdot \mathbf{x}_2$ and $\sin\phi = \mathbf{x}_1 \cdot \mathbf{y}_2$. The variables (ρ, ϕ) are called the shape variables. Taking the time derivative of the shape variables provides the following dynamics.

$$\dot{\rho} = -\sin\phi \tag{21}$$
$$\dot{\phi} = \left(\frac{k_1}{1 + k_1\rho}\right)\cos\phi - u_2 \tag{22}$$

In the above equations, u_2 is the steering command for the robot. The following control law can be used for curve tracking.

$$u_2 = \left(\frac{\pm k_1}{1 + k_1\rho}\right)\cos\phi \pm K_p(\rho - \rho_0)\cos\phi + \mu\sin\phi \tag{23}$$

The "±" signs in the above law represent different versions of the same control law. They are used depending on whether the initial position of the robot is to

the left or the right of the curve. The desired separation between the robot and the curve is specified by ρ_0 in Equation (23). The controller in Equation (23) resembles a PD controller, where K_p and μ can be viewed as the proportional and derivative gains, respectively. It has been theoretically justified earlier that the controller in Equation (23) achieves curve tracking [11]. To enable any vehicle to follow a survey path, only the control input given by u_2 in Equation (23) is required to be calculated. For this, the only quantities required to be measured are the position and orientation of a particular autonomous vehicle. This makes it very easy to write computer code for performing complicated surveys.

Also, the curve tracking dynamics given by Equations (21)-(23) are amenable to rigorous analysis that can mathematically certify the robustness of the curve tracking control law given in Equation (23), with respect to the uncertainties and input delays that prevail in cooperative marine robotic surveys. Input delays refer to the latency between the time a command is issued to the robot, and the time the command is actually executed. We can represent uncertainties by adding an unknown perturbation $\delta(t)$ to u_2 in Equation (23), and by replacing u_2 in Equation (22) by Gu_2 for an unknown constant G. Then we can replace the time argument in u_2 by $t - \tau$ where τ is the latency. In the controls literature, the $\delta(t)$'s and G's are called actuator errors and control gains, respectively.

Since many possible actuator errors, control gains, and delays can occur (depending on wave action, faults in the control, or other factors), it is convenient to study robustness using a class of non-negative valued functions called Lyapunov-Krasovskii functionals, and then one can determine the true control gain using dynamic extensions that ensure adaptive tracking and parameter identification. Much like the classical Lyapunov function approach for ODEs, the robustness properties follow by checking that the Lyapunov-Krasovskii functional decays in a certain way along all trajectories of the perturbed delayed system. See [9,29] where this Lyapunov-Krasovskii based robustness analysis is done for the planar curve tracking dynamics given by Equations (21)-(23), and [30] for extensions to three dimensional curves. For simplicity, we assume in the rest of this chapter that $G = 1$ and that no disturbances δ or delays are present.

3.3 Mapping

Mapping a region of interest is a primary activity performed in most environmental surveys. A bathymetric map is essential to avoid damage to marine vehicles by accidental deployment in shallow water. A map explicitly showing the surface oil concentration obtained as a result of the autonomous surveys is desired. However, covering every point on the surface of a lagoon (even in a small region of interest) is not necessarily a trivial task. Hence, this section focuses on the mapping algorithm used to create meaningful representations from the data collected by a fleet of marine vehicles.

In Figure 6, a square grid is shown overlaid on a surface of interest (which is the lagoon in our case). The cells that lie outside the boundary of the lagoon are not included in the computation. The position vector $\mathbf{x}_i$ of the i^{th} cell's center is used to denote a particular cell i. Figure 6 shows two sample trajectories (red and blue

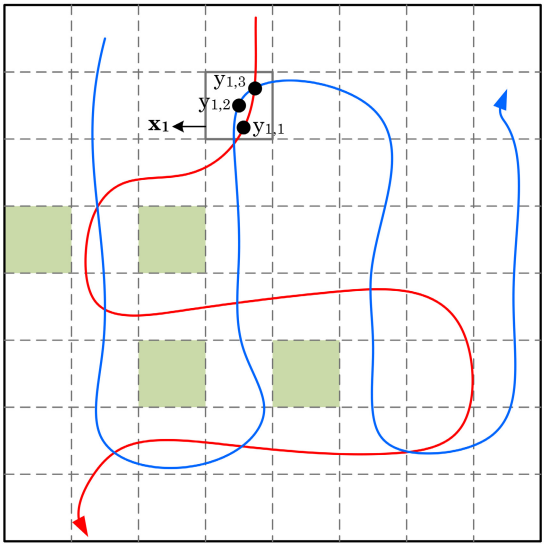


Fig. 6. Paths on a grid in a region of interest. Shaded squares show areas on the grid where no sensor readings are available.

curves) taken by an AUV (e.g., the EcoMapper). When an AUV passes through a cell (assuming that the time the AUV takes to move through a particular cell is greater than one sampling period), it records the sensor measurements (depth or oil concentration in our case). Such cells where the AUV records sensor measurements are represented by unshaded cells in Figure 6. We assume that at least one measurement is recorded in each unshaded cell in Figure 6. Multiple measurements may be available in some unshaded cells. For example, cell $\mathbf{x}_1$ has three measurements available, i.e., $y_{1,1}$, $y_{1,2}$, and $y_{1,3}$. As the grid is made finer in order to get a high-resolution map, it is tedious to assign a survey path such that a vehicle passes through all the cells on a grid. Hence, cells (pockets) with no sensor measurements (shaded in green in Figure 6) may exist. If each cell has just a single measurement associated to it, the job of creating a map becomes simpler. To make a map where each cell has one reading (a sensor measurement or a predicted value) we proceed as follows.

The aim is to assign one measurement y_i to a cell $\mathbf{x}_i$, $i = 1, ...N$, where $N \in \mathbb{N}$ is the number of cells on a map and $\mathbf{x}_i$ is a position vector representing a cell. For cells having only one sensor measurement, y_i equals that particular sensor measurement. For cells having more than one sensor measurement, y_i is assigned to be the average of all sensor measurements in that cell. For example, $y_1 = (y_{1,1} + y_{1,2} + y_{1,3})/3$. For a cell $\mathbf{x}_\star$ having no sensor measurements, given sensor readings $y_1, \cdots, y_n$ for some non-empty neighboring cells $\mathbf{x}_1, \cdots, \mathbf{x}_n$, $n \in \mathbb{N}$ and $n < N$, the predicted value $y_\star$ is to be found.

Similar to [25], the idea here is to approximate a target value y for a given cell at location $\mathbf{x}$ using M basis functions $r_1(\mathbf{x}), \cdots, r_M(\mathbf{x})$, with weights $w_1, \cdots, w_M$

so that

$$f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w} \tag{24}$$

$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon \tag{25}$$

where $\phi(\mathbf{x}) = [r_1(\mathbf{x}), \cdots, r_M(\mathbf{x})]^T$ and $\mathbf{w} = [w_1, \cdots, w_M]^T$ contain unknown parameters which are yet to be specified, and solved for later. The variable ϵ represents measurement noise, which is assumed to have a Gaussian distribution with zero mean and variance σ_n^2, i.e. $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. At this point, σ_n is unknown, and the appropriate value for it will be decided later. The subscript n in the symbol σ_n denotes that the quantity σ_n is related to the measurement noise. This subscript n should not be confused with, and is not related to, the subscript n used to represent measurement y_n at location $\mathbf{x}_n$. Equation (25) is the measurement equation and we assume that $f(\cdot)$ in (24) is the underlying model, i.e., given a location $\mathbf{x}$ on the surface of the lagoon, $f(\cdot)$ provides the depth at location $\mathbf{x}$. Given a location $\mathbf{x}_i$ and a corresponding measurement y_i, the following distribution holds

$$P(y_i(\mathbf{x}_i)|\mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma_n}\exp\left(-\frac{\left(y_i - \phi^T(\mathbf{x}_i)\mathbf{w}\right)^2}{2\sigma_n^2}\right). \tag{26}$$

Since we are not making a map in real-time, all the survey data are available for us to make the required map offline. Therefore, once the weight vector $\mathbf{w}$ is set, it does not need to be updated. This is a major difference between the method used for mapping presented here and the work in [25]. In [25], tidal measurements are updated in real-time at a particular frequency, so the weight vector $\mathbf{w}$ is updated as well. Therefore, here we use a zero mean Gaussian prior with a covariance matrix Σ_p on the weight $\mathbf{w}$, i.e.,

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p). \tag{27}$$

Although common, it is not necessary to assume Gaussian processes which have a zero mean function. Such an assumption is not a limitation, because the posterior process is not necessarily zero mean[31].

The covariance of the Gaussian process $f(x) = \phi(\mathbf{x})^T\mathbf{w}$ is calculated as follows,

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &\triangleq \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})])(f(\mathbf{x}') - \mathbb{E}[f(\mathbf{x}')])] \\ &= \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] \\ &= \phi(\mathbf{x})^T\mathbb{E}(\mathbf{w}\mathbf{w}^T)\phi(\mathbf{x}') \\ &= \phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}'). \end{aligned} \tag{28}$$

Note that $\phi(\mathbf{x}) = [r_1(\mathbf{x}), \cdots, r_M(\mathbf{x})]^T$. Therefore, given a covariance matrix Σ_p, and any set of basis functions $\mathbf{r}_1(\cdot), \cdots, \mathbf{r}_M(\cdot)$, we can compute the covariance function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}')$ according to Equation (28). Conversely, for

every (positive definite) covariance function $k(\cdot,\cdot)$, there exists a (possibly infinite) expansion in terms of basis functions. For details, see Section 4.3 in [31]. Since the explicit forms of basis functions are not directly used in our mapping procedure, we specify the covariance function $k(\cdot,\cdot)$ and omit the calculation of basis functions according to this choice of the covariance function $k(\cdot,\cdot)$. We adopt the standard form for a covariance function [32], which is

$$k(\mathbf{x},\mathbf{x}') \triangleq \sigma_f^2 \exp\left(-\frac{1}{2}\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{l^2}\right). \tag{29}$$

Here l is the length-scale parameter which defines the global smoothness of the function f and σ_f^2 denotes the amplitude or the signal variance.

For notational convenience, let vector $\mathbf{y} = [y_1, \cdots, y_n]^T$, where each available measurement y_i, for $i \in \{1, \cdots, n\}$ is of the form given by Equation (25). Let vector $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T$ represent a vector consisting of locations $\mathbf{x}_i$ corresponding to measurements y_i. Let $\Phi(\mathbf{X}) \triangleq [\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_n)]$. For notational convenience we will use the abbreviation $\Phi \triangleq \Phi(\mathbf{X})$. Now we can write:

$$\begin{aligned} P(\mathbf{y}(\mathbf{X})|\mathbf{w}) &= \prod_{i=1}^{n} P(y_i(\mathbf{x}_i)|\mathbf{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \phi^T(\mathbf{x})\mathbf{w})^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} \exp\left(-\frac{|\mathbf{y} - \Phi^T\mathbf{w}|^2}{2\sigma_n^2}\right). \end{aligned} \tag{30}$$

This implies,

$$\mathbf{y}(\mathbf{X})|\mathbf{w} \sim \mathcal{N}(\Phi^T\mathbf{w}, \sigma_n^2\mathbf{I}), \tag{31}$$

where $\mathbf{I}$ denotes an identity matrix. As a consequence we have,

$$\begin{aligned} P(\mathbf{w}|\mathbf{y}(\mathbf{X})) &= \frac{P(\mathbf{y}(\mathbf{X})|\mathbf{w})P(\mathbf{w})}{P(\mathbf{y}(\mathbf{X}))} \\ &\propto P(\mathbf{y}(\mathbf{X})|\mathbf{w})P(\mathbf{w}) \\ &\propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - \Phi^T\mathbf{w})^T(\mathbf{y} - \Phi^T\mathbf{w})\right) \exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T\left(\frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1}\right)(\mathbf{w} - \bar{\mathbf{w}})\right), \end{aligned} \tag{32}$$

where $\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1})^{-1}\Phi\mathbf{y}$. Let $A = \sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}$. Then, we get

$$\mathbf{w}|\mathbf{y}(\mathbf{X}) \sim \mathcal{N}(\bar{\mathbf{w}}, A^{-1}). \tag{33}$$

Recall that $\mathbf{x}_\star$ represents the location of a cell with no sensor measurements and that the predicted value $y_\star$ is to be found. Here $y_\star = y(\mathbf{x}_\star)$. Since $P(f(\mathbf{x}_\star)|\mathbf{y}(\mathbf{X})) = \int P(f(\mathbf{x}_\star)|\mathbf{w})P(\mathbf{w}|\mathbf{y}(\mathbf{X}))\mathrm{d}\mathbf{w}$, we get

$$f(\mathbf{x}_\star)|\mathbf{y}(\mathbf{X}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_\star)^T A^{-1}\mathbf{X}\mathbf{y}, \phi(\mathbf{x}_\star)^T A^{-1}\phi(\mathbf{x}_\star)\right) \tag{34}$$

$$f(\mathbf{x}_\star)|\mathbf{y}(\mathbf{X}) \sim \mathcal{N}(B, C), \tag{35}$$

where

$$B = \phi(\mathbf{x}_\star)^T \Sigma_p \Phi(\Phi^T \Sigma_p \Phi + \sigma_n^2 \mathbf{e})^{-1}\mathbf{y} \tag{36}$$

$$C = \phi(\mathbf{x}_\star)^T \Sigma_p \phi(\mathbf{x}_\star) - \phi(\mathbf{x}_\star)^T \Sigma_p \Phi(\Phi^T \Sigma_p \Phi + \sigma_n^2 \mathbf{e})^{-1}\Phi^T \Sigma_p \phi(\mathbf{x}_\star). \tag{37}$$

Equation (35) is obtained by simplifying Equation (34). Further simplification of Equation (35) leads to Equation (38) as follows:

$$f(\mathbf{x}_\star)|\mathbf{y}(\mathbf{X}) \sim \mathcal{N}(\tilde{B}, \tilde{C}), \text{ where} \tag{38}$$

$$\tilde{B} = K(\mathbf{x}_\star, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{e})^{-1}\mathbf{y} \tag{39}$$

$$\tilde{C} = K(\mathbf{x}_\star, \mathbf{x}_\star) - K(\mathbf{x}_\star, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{e})^{-1}K(\mathbf{X}, \mathbf{x}_\star), \tag{40}$$

and where K is defined as

$$K(\mathbf{X}, \mathbf{X})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j), \tag{41}$$

$$K(\mathbf{x}_\star, \mathbf{X})_{1,j} = k(\mathbf{x}_\star, \mathbf{x}_j), \tag{42}$$

$$K(\mathbf{X}, \mathbf{x}_\star)_{i,1} = k(\mathbf{x}_i, \mathbf{x}_\star), \tag{43}$$

$$K(\mathbf{x}_\star, \mathbf{x}_\star) = k(\mathbf{x}_\star, \mathbf{x}_\star). \tag{44}$$

Thus, given the sensor measurements $y_1, \cdots, y_n$ for cells at locations $\mathbf{x}_1, \cdots, \mathbf{x}_n$, we have obtained the distribution $f(\mathbf{x}_\star)$ for any (empty) cell $\mathbf{x}_\star$. For details the readers should refer to [31].

Given parameters $\{\sigma_f, l, \sigma_n\}$, we can use Equations (34)-(44) to calculate the distribution of $f(\mathbf{x}_\star)$, i.e., the predicted measurement $y(\mathbf{x}_\star)$ for a given empty cell $\mathbf{x}_\star$. We calculate $\{\sigma_f, l, \sigma_n\}$ using the available measurement data $\{\mathbf{X}, \mathbf{y}\}$. Using the method of maximum likelihood estimation, we choose

$$\{\sigma_f, l, \sigma_n\} = \underset{\{\sigma_f, l, \sigma_n\}}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i). \tag{45}$$

Values for the unknown parameters $\{\sigma_f, l, \sigma_n\}$, maximizing the probability that the measurements $\mathbf{y}$ appear if we use the model in Equations (24)-(25) to predict the target value for cells X, have thus been found. A variety of optimization algorithms can be applied to solve for $\{\sigma_f, l, \sigma_n\}$. For this work a genetic algorithm is used because of simplicity of implementation, effectiveness for both convex and non-convex problems and the ability to avoid being trapped at a local optimum. To reduce the computational cost when predicting the measurement values at

a particular empty cell, we use data from the first 400 cells closest to the required empty cell. For our particular implementation of the genetic algorithm, a population size of 500 is used. The population represents initial guesses of candidates for the parameters maximizing the probability on the right hand side of Equation (45). The search space used is $\sigma_f \in (0, 100], l \in (0, 500], \sigma_n \in (0, 100]$. After the genetic algorithm produces $\{\sigma_f, l, \sigma_n\}$, Equations (34)-(44) give the distribution of the prediction $y_\star$ for any empty cell at a given location $\mathbf{x}_\star$. The mean value $y_\star$ is then assigned to an empty cell's location.

4 Experimental Results and Data Analysis

During the 21-day survey of a tidal lagoon at the Grand Isle, Louisiana, where crude oil was spotted along the beaches, autonomous surveys were carried out. The experiments included parameter identification for ASV-Victoria, and curve following control for both ASV-Victoria and AUV-Fetch1, while simultaneously collecting oil and bathymetry data. In a marine survey, it is very important to know the environment. A bathymetric map is very helpful to decide spots safe for deploying marine vehicles, as vehicles can get damaged if they are deployed in extremely shallow locations. Such needs call for collaboration between autonomous vehicles with varying capabilities. As the EcoMapper is equipped with the sonar system for depth measurements, the EcoMapper was deployed first on autonomous runs over the entire lagoon to acquire bathymetric data. Based on this data, a bathymetric map of the lagoon was obtained using the method described in Section 3.3. Once a bathymetric map was obtained, ASV-Victoria was deployed in a safe region. Initially tests were performed on ASV-Victoria to identify parameters as described in Section 3.1. ASV-Victoria was then launched to perform an autonomous survey. The ASV was driven by the curve tracking controller described in Section 3.2. To measure crude-oil, a survey was performed using ASV-Victoria.

Following the autonomous survey carried out by ASV-Victoria, AUV-Fetch1 was deployed. It performed similar curve-tracking runs during which it also measured oil data. In areas where relatively higher concentrations of oil were noticed, ROV-β was launched to study the underwater habitat in more detail. When on surface, the oil sensors on the ROV also collected data. Such data collected by all three vehicles was used along with the mapping procedure described in Section 3.3 to produce a map of oil concentration in the area surveyed. The following sub-sections present the details about the on-field experiments, and the results obtained.

4.1 Parameter Identification

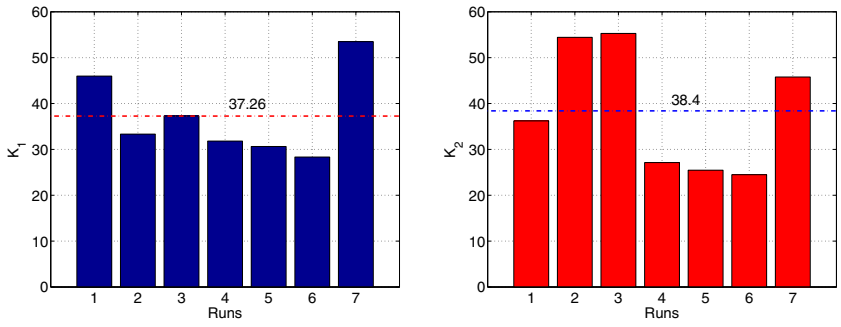
For parameter identification tests, ASV-Victoria was operated in open loop mode. Seven sets of thruster speed commands were sent to the lower level thruster controllers. Based on the linear and angular velocity measurements during these tests, the parameters K_1 and K_2 were calculated using Equations

(10) and (11). Figures 7(a) and 7(b) show the result for the seven test runs. The dotted lines show the average values, i.e. $K_1 = 37.26$ and $K_2 = 38.4$.

4.2 Curve Tracking

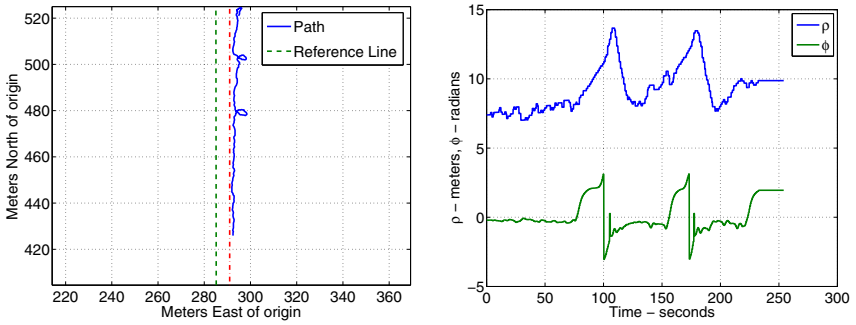
To demonstrate the performance of the proposed curve-tracking control law, both ASV-Victoria and AUV-Fetch1 were commanded to track certain desired curves. Figures 8(a) and 8(b) show the results of a straight line tracking experiment with ASV-Victoria. For this experimental run, $\rho_0 = 6m$ and control gains $\mu = 5$ and $K_p = 1$ were used. The dotted line in Figure 8(a) is the reference line the ASV was trying to follow and the solid curve is the path taken by the ASV. Since $\rho_0 = 6m$, it is expected that ASV-Victoria should maintain a distance of six meters from the reference line. The red dash-dotted line in Figure 8(a) shows the buffer from the reference line that Victoria is expected to maintain. We can see that the ASV follows the line well and maintains the required separation of close to six meters, but there are some intermittent deviations. The details are shown in Figure 8(b), where the solid line represents ρ and the dotted line represents the error in orientation ϕ. From Figure 8(b), we observe that the distance ρ is maintained just above 8 on average, and that ϕ stays close to zero. After disturbances occur, ϕ comes back to zero very quickly. Data analysis suggests that some faults occurred in the electronic thruster-speed controller, thus causing the disturbances. Despite such disturbances, the vehicle recovered very soon and tracked the line, maintaining the required separation of ρ_0. This shows the robustness of the control law.

Circular path tracking experiments were also carried out using ASV-Victoria. For the particular circular path tracking run shown in Figure 9(a) and 9(b), the constants $\rho_0 = 4m$, $\mu = 5$, and $K_p = 1$ were selected. The radius of the desired circle was $R = 1m$. The vehicle motion was clockwise. Figure 9(a) shows the experimental result. The green dashed circle is the reference circle and the red dash-dotted circle is the buffer Victoria is expected to maintain from the refer-



(a) Estimation of the parameter K_1 (b) Estimation of the parameter K_2

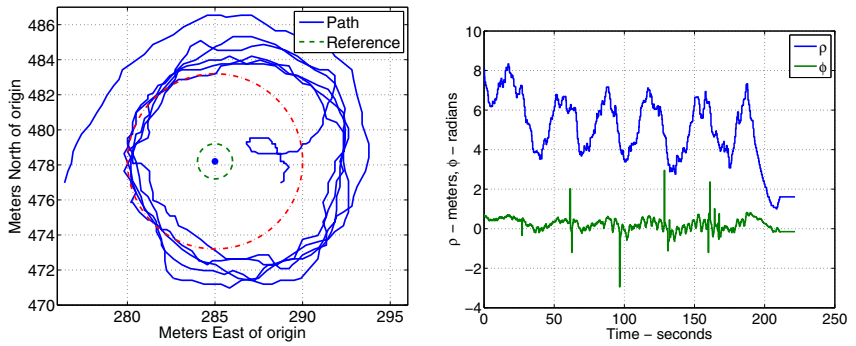
Fig. 7. Results of parameter identification tests



(a) Trajectory of line following using ASV-Victoria

(b) Position and orientation error analysis for line following using ASV-Victoria

Fig. 8. Line following using ASV-Victoria



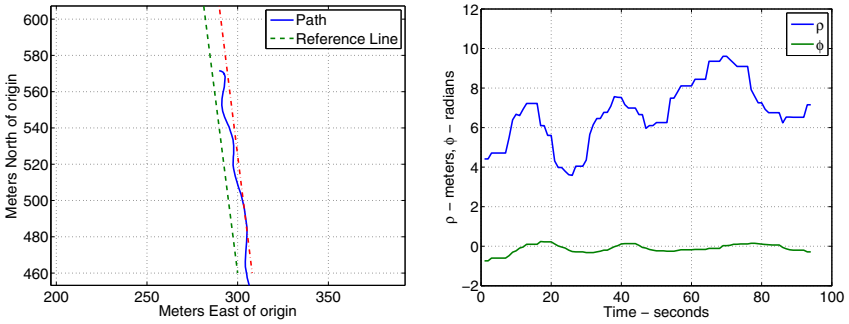
(a) Trajectory obtained while tracking a circular curve using ASV-Victoria

(b) Position and orientation error analysis for circular curve tracking using ASV-Victoria

Fig. 9. Circular curve following using ASV-Victoria

ence circle. The solid blue curve represents the path taken by ASV-Victoria. It is seen from Figure 9(a) that the ASV tracked the desired circle reasonably well, maintaining the required buffer distance, although the path had some displacement to the right. This was caused by the current in the lagoon (approximately $20cm/s$, from southwest to northeast). Figure 9(b) shows separation ρ and the error in orientation ϕ. GPS localization errors close to $3m$ on average at Grand Isle caused some oscillations, but it is seen from Figure 9(b) that ϕ was maintained close to zero and ρ was maintained around $5m$ on average which equals $\rho_0 + R$. This suggests that Victoria tracked the circular curve successfully in the presence of real environmental disturbances and localization error and maintained the required separation ρ_0 from the reference curve.

Straight line tracking experiments were also performed using Fetch1. For the particular results shown in Figure 10(a) and 10(b), the settings used for the



(a) Trajectory of line following using Fetch1

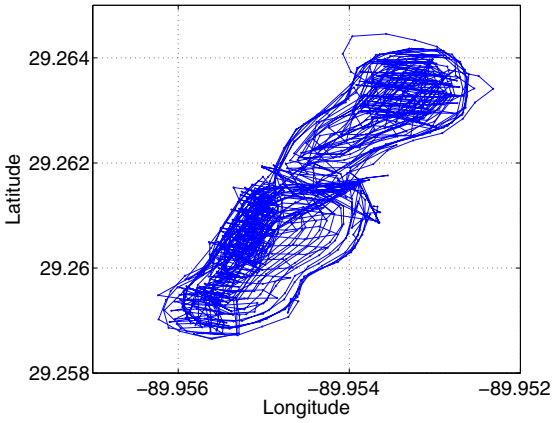
(b) Position and orientation error analysis for line following using Fetch1

Fig. 10. Line following using AUV-Fetch1

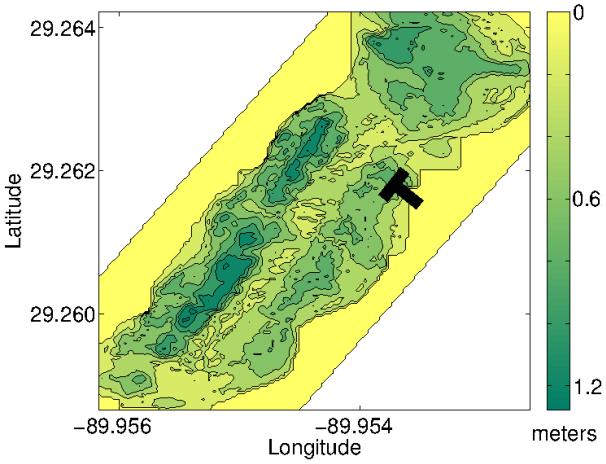
constants were $\rho_0 = 8m$, $\mu = 0.1$, and $K_p = 0.001$. The experimental results are shown in Figure 10(a), where the dashed line is the reference line Fetch1 was trying to follow and the solid curve is the actual path taken by Fetch1. The red dash-dotted line is the buffer distance Fetch1 is expected to maintain from the reference line. Because the dynamics of Fetch1 were notably slower than that of Victoria, smoother convergence is observed in Figure 10(a). Figure 10(b) shows separation ρ and the errors in orientation ϕ. We can see that the separation ρ converges to the desired value, i.e., $\rho_0 = 8m$, and ϕ stays close to zero. From the above experimental results, we can observe that the control laws described in Section 3 are robust in the presence of winds, water currents, tides, and engineering constraints such as sensor inaccuracy, localization errors, and network delays. Note that such line and curve following tests are performed because in a real life situation it is not always possible to find identical robots so that wider areas can be surveyed in formation. Also, different types of robots usually have widely different dynamics which can impede formation control. However, such differences between vehicles can be exploited to map a time varying flow field (e.g., oil dispersed in water affected by sea waves and winds). If a flow field is time varying, different robots can be made to follow specific paths on regular time-intervals. This can help generate a better map because data along a specific path are available at different time instants. This makes disturbance rejection possible and can be used to account for effects of tides and currents. Also, in such a setup, vehicles with different capabilities can move along the same path to measure different types of quantities in a flow (e.g., different chemicals in water). Hence, such path following experiments are performed in this work.

4.3 Bathymetry of the Lagoon

In the bathymetric surveys, the EcoMapper followed a path that spanned the entire lagoon, and collected bathymetric data at a sampling rate of 2Hz along the path. Figure 11(a) shows the path taken by the EcoMapper while it performed a



(a) A compilation of actual paths taken by the EcoMapper while on various bathymetric surveys.



(b) A depth map of the lagoon. The color bar shows the depth of the lagoon from the surface in meters.

Fig. 11. Generating a smooth bathymetric map

complete bathymetric survey of the lagoon. From Figure 11(a), it is seen that the bathymetric data obtained are not uniform (i.e., there are some pockets which are left unsampled). Using the mapping method in Section 3.3, a smooth depth map is produced. For generating such a map, a square grid consisting of 22,500 cells (150×150 cells) was used to overlay the survey area. For the cells in this grid which were unsampled by the EcoMapper, the depth was predicted by using Equations(34)-(44). To reduce the computational cost when predicting the depth in an unsampled cell, only data from 400 cells closest to the required empty cell are used. This makes the process faster than using the entire set of measurements.

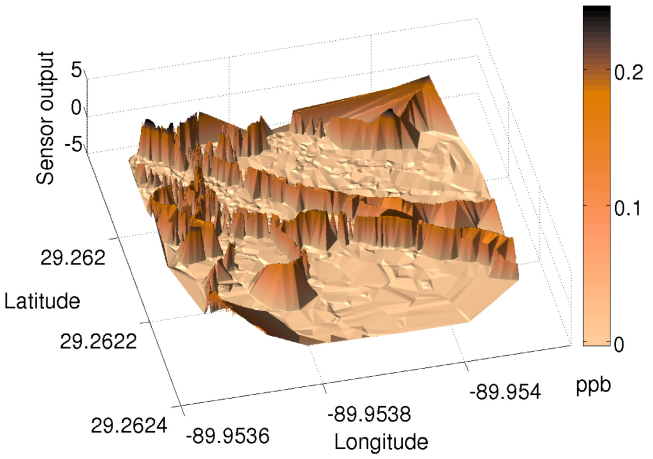
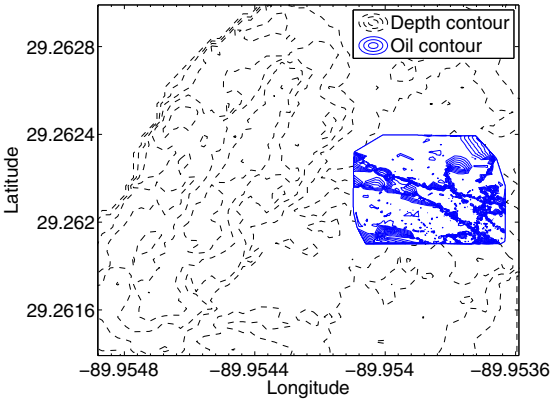


Fig. 12. Crude-oil concentration map

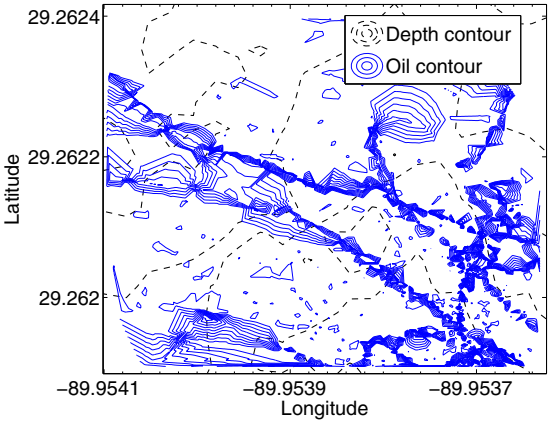
After predicting the depth for all empty cells the depth map obtained for the lagoon is shown in Figure 11(b). The light (yellow) portions represent parts outside of the lagoon. The darker (green) regions are the deepest whereas the lighter yellow areas are relatively shallower. The thick black T-shaped structure shows the location of the dock, where the vehicles were launched. Tides caused increases or decreases in the depth measured by the EcoMapper. Figure 11(b) was produced by compensating for such tidal effects, i.e., the depth map shown in Figure 11(b) is static (in time). If such a map is to be used to check the depth at a given location (on the surface of the lagoon) at given instants of time, tidal effects corresponding to the given time instant need to be added to the depth obtained using the map in Figure 11(b).

4.4 Surface Crude Oil Concentration Map

ASV-Victoria intensely surveyed an area of the lagoon and collected surface crude oil concentration data. Based on this data a crude oil concentration map was generated as shown in Figure 12. The method used is as described in Section 3.3. In order to get a better sense of the distribution of the measured surface crude oil, a contour plot of the concentration map shown in Figure 12 is overlaid on top of the contours from the depth map of the lagoon shown in Figure 11(b). The overlays are shown in Figures 13(a) and 13(b). Dotted contours correspond to the contours on the depth map in Figure 11(b). The solid (blue) contours represent the contours of the oil concentration map in Figure 12. Figure 13(b) provides a zoomed view of Figure 13(a). Field measurements were made to verify the bathymetric and the oil concentration map. The measurements suggest that



(a) The surface crude oil contour overlaid on the depth map of the lagoon.



(b) The surface crude oil contour overlaid on the depth map - a zoomed view.

Fig. 13. Generating a crude-oil concentration map

the results of our analysis are close to reality. This shows the effectiveness of the proposed method in reconciling data from a collaborative autonomous marine survey.

5 Conclusion

The methods presented in this chapter enable collaborative autonomous marine environmental surveys, using a fleet of heterogeneous marine vehicles which forms a distributed sensor and control network. The path following controllers used are robust. This enables vehicles with different dynamics to follow lines and curves in order to reconstruct a time varying field in a region of interest while

on an environmental survey. In our control algorithm, we use simple dynamic models and control laws, which makes it possible to quickly deploy a fleet of autonomous vehicles. This enables coverage of a wide region for an environmental survey, even if cooperative formation control is not possible due to differing vehicle dynamics. The mapping algorithms reconcile survey data from a variety of heterogeneous marine vehicles, and create a high fidelity visual representation of the desired survey data. Such collaboration between heterogeneous vehicles has the potential to assist in finding hot-spots in large environments even if the individual marine vehicles may possess limited battery power.

Using the proposed method, we performed a 21-day survey at the Grand Isle in Louisiana to evaluate the level of crude oil remaining in the area after the Deepwater Horizon oil spill. We presented experimental results on controller performance. The controllers were tested on ASV-Victoria and AUV-Fetch1. The results show the effectiveness and robustness of our control law in the presence of natural disturbances like wind, water currents, and engineering constraints such as sensor inaccuracy, localization errors, and network delays. With the help of the fleet of marine robots, we collected large amounts of survey data, including bathymetric data and crude-oil data. The mapping algorithm successfully reconciled the survey data and created high fidelity visual representations of them.

From the survey data, we see that there was crude oil remaining in the water in the coastal areas in the Gulf of Mexico after cleanup efforts. Although the concentration is low, there may be unknown long-term effects on the entire coastal ecosystem. Knowledge gathered after this oil spill can help us prepare for possible disasters of a similar scale in the future.

Acknowledgement. The authors acknowledge the following individuals and organizations for making this work possible. J. Elliott, D. Godschalk and S. Williams (College of William & Mary) for field assistance. Professor Edward Overton (School of the Coast & Environment) at Louisiana State University (LSU) for analyzing samples collected during the experiments. The Grand Isle State Park for supporting our field experiments. We thank GTSR field team members Steven Bradshaw, Valerie Bazie, Sean Maxon, Phillip Cheng and Brian Redden for their efforts in the field, and all other GTSR members for remote assistance in various ways. This research work was supported by NSF RAPID Grant ECCS-1056253 and ECCS-1056255 with partial support from ONR grants N00014-08-1-1007, N00014-09-1-1074, and N00014-10-10712(YIP), and NSF grants ECCS-0845333(CAREER) and CNS-0931576. We appreciate the support received from the School of Electrical and Computer Engineering at the Georgia Institute of Technology and other sponsors listed at http://gtsr.gtsav.gatech.edu/sponsors.

References

1. Geen, M.: Advances in Marine Survey Products and Platforms. In: OCEANS 2007-Europe, vol. 1-3, pp. 984–989 (2007)
2. Hausler, A., Ghabcheloo, R., Kaminer, I., Pascoal, A., Aguiar, A.: Path Planning For Multiple Marine Vehicles. In: OCEANS 2009-Europe, vol. 1 & 2, pp. 423–431 (2009)

3. Pettersen, K., Egeland, O.: Exponential Stabilization of An Underactuated Surface Vessel. In: Proceedings of the 35th IEEE Conference on Decision and Control, vol. 1, pp. 967–972 (1996)
4. Pettersen, K., Lefeber, E.: Way-Point Tracking Control Of Ships. In: Proceedings of the 40th IEEE Conference on Decision and Control, vol. 1, pp. 940–945 (2001)
5. Do, K., Jiang, Z., Pan, J.: Universal Controllers for Stabilization and Tracking of Underactuated Ships. Systems & Control Letters 47(4), 299–317 (2002)
6. Ghommam, J., Mnif, F., Benali, A., Derbel, N.: Nonsingular Serret-Frenet Based Path Following Control for an Underactuated Surface Vessel. Journal of Dynamic Systems, Measurement, and Control 131(2), 021006(8 pages) (2009)
7. Xiang, X., Lapierre, L., Liu, C., Jouvencel, B.: Path Tracking: Combined Path Following and Trajectory Tracking for Autonomous Underwater Vehicles. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 3558–3563 (2011)
8. Do, K., Pan, J.: Robust Path Following of Underactuated Ships Using Serret-Frenet Frame. In: Proceedings of the American Control Conference, vol. 3, pp. 2000–2005 (2003)
9. Malisoff, M., Mazenc, F., Zhang, F.: Stability and Robustness Analysis for Curve Tracking Control Using Input-to-State Stability. IEEE Transactions on Automatic Control 57(5), 1320–1326 (2012)
10. Pettersen, K., Fossen, T.: Underactuated Dynamic Positioning of A Ship-Experimental Results. IEEE Transactions on Control Systems Technology 8(5), 856–863 (2000)
11. Zhang, F., Justh, E., Krishnaprasad, P.S.: Boundary Following Using Gyroscopic Control. In: Proceedings of the 43rd IEEE Conference on Decision and Control, vol. 5, pp. 5204–5209 (2004)
12. Zhang, F., O'Connor, A., Luebke, D., Krishnaprasad, P.S.: Experimental Study of Curvature-Based Control Laws for Obstacle Avoidance. In: Proceedings of 2004 IEEE International Conf. on Robotics and Automation, vol. 4, pp. 3849–3854 (2004)
13. Kim, J., Zhang, F., Egerstedt, M.: Curve Tracking Control for Autonomous Vehicles With Rigidly Mounted Range Sensors. Journal of Intelligent and Robotic Systems 56(1-2), 177–197 (2009)
14. Zhang, F., Fratantoni, D.M., Paley, D., Lund, J., Leonard, N.E.: Control of Coordinated Patterns for Ocean Sampling. International Journal of Control 80(7), 1186–1199 (2007)
15. Wu, W., Zhang, F.: Robust Cooperative Exploration With A Switching Strategy. IEEE Transactions on Robotics 28(4), 828–839 (2012)
16. Wu, W., Zhang, F.: Cooperative Exploration of Level Surfaces of Three Dimensional Scalar Fields. Automatica, the IFAC Journal 47(9), 2044–2051 (2011)
17. Dasgupta, P.: A Multiagent Swarming System for Distributed Automatic Target Recognition Using Unmanned Aerial Vehicles. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 38(3), 549–563 (2008)
18. Boardman, M., Edmonds, J., Francis, K., Clark, C.: Multi-Robot Boundary Tracking With Phase and Workload Balancing. In: Proc. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3321–3326 (2010)
19. Jin, X., Ray, A.: Coverage Control of Autonomous Vehicles for Oil Spill Cleaning in Dynamic and Uncertain Environments. In: Proc. 2013 American Control Conference (ACC), pp. 2594–2599 (2013)

20. Johnson, B., Hallin, N., Leidenfrost, H., O'Rourke, M., Edwards, D.: Collaborative Mapping With Autonomous Underwater Vehicles in Low-Bandwidth Conditions. In: OCEANS 2009 - EUROPE, pp. 1–7 (2009)
21. Carlési, N., Michel, F., Jouvencel, B., Ferber, J.: Generic Architecture For Multi-AUV Cooperation Based on A Multi-Agent Reactive Organizational Approach. In: Proc. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5041–5047 (2011)
22. Li, H., Popa, A., Thibault, C., Trentini, M., Seto, M.: A Software Framework for Multi-Agent Control of Multiple Autonomous Underwater Vehicles for Underwater Mine Counter-Measures. In: Proc. 2010 International Conference on Autonomous and Intelligent Systems (AIS), pp. 1–6 (2010)
23. Gustavi, T., Dimarogonas, D.V., Egerstedt, M., Hu, X.: Sufficient Conditions for Connectivity Maintenance and Rendezvous in Leader-Follower Networks. Automatica 46(1), 133–139 (2010)
24. Mukhopadhyay, S., Wang, C., Bradshaw, S., Maxon, S., Patterson, M., Zhang, F.: Controller Performance of Marine Robots In Reminiscent Oil Surveys. In: Proc. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), Vilamoura, Portugal, pp. 1766–1771 (2012)
25. Liang, X., Wu, W., Chang, D., Zhang, F.: Real-Time Modelling of Tidal Current for Navigating Underwater Glider Sensing Networks. Procedia Computer Science 10, 1121–1126 (2012)
26. Patterson, M.R., Sias, J.H.: Modular Autonomous Underwater Vehicle System. U.S. Patent 5, 995, 882 (1999)
27. Patterson, M.R., Sias, J.H.: Fetch!® Commercial Autonomous Underwater Vehicle: A Modular, Platform-Independent Architecture Using Desktop Personal Computer Technology. In: Ocean Community Conference 1998 Proceedings, Baltimore, MD, vol. 2, pp. 891–897 (1998)
28. Patterson, M.R.: A Finite State Machine Approach to Layered Command And Control of Autonomous Underwater Vehicles Implemented in G, A Graphical Programming Language. In: Ocean Community Conference 1998 Proceedings, Baltimore, MD, vol. 2, pp. 745–751 (1998)
29. Malisoff, M., Zhang, F.: Adaptive Control for Planar Curve Tracking Under Controller Uncertainty. Automatica 49(5), 1411–1418 (2013)
30. Malisoff, M., Zhang, F.: Robustness of A Class of Three-Dimensional Curve Tracking Control Laws Under Time Delays and Polygonal State Constraints. In: Proc. 2013 American Control Conference (ACC 2013), Washington D.C., USA, pp. 5710–5715 (2013)
31. Rasmussen, C.E., Williams, C.: Gaussian Processes for Machine Learning. MIT Press (2006)
32. Stachniss, C., Plagemann, C., Lilienthal, A.: Gas Distribution Modeling Using Sparse Gaussian Process Mixtures. Autonomous Robots 26(2-3), 187–202 (2009)

Human-Robot Mutual Trust in (Semi)autonomous Underwater Robots

Yue Wang[1], Zhenwu Shi[2], Chuanfeng Wang[3], and Fumin Zhang[2]

[1] Department of Mechanical Engineering, Clemson University,
Fluor Daniel Building, Clemson, SC 29634
yue6@clemson.edu
[2] School of Electrical and Computer Engineering,
Georgia Institute of Technology, Atlanta, GA 30332
{zwshi,fumin}@gatech.edu
[3] George W. Woodruff School of Mechanical Engineering,
Georgia Institute of Technology, Atlanta, GA 30332
cwang329@gatech.edu

Abstract. It is envisioned that a human operator is able to monitor and control one or more (semi)autonomous underwater robots simultaneously in future marine operations. To enable such operations, a human operator must trust the capability of a robot to perform tasks autonomously, and the robot must establish its trust to the human operator based on human performance and follow guidance accordingly. Therefore, we seek to i) model the mutual trust between humans and robots (especially (semi)autonomous underwater robots in this chapter), and ii) develop a set of trust-based algorithms to control the human-robot team so that the mutual trust level can be maintained at a desired level. We propose a time series based mutual trust model that takes into account robot performance, human performance and overall human-robot system fault rates. The robot performance model captures the performance evolution of a robot under autonomous mode and teleoperated mode, respectively. Furthermore, we specialize the robot performance model of a YSI EcoMapper autonomous underwater robot based on its distance to a desired waypoint. The human performance model is inspired by the Yerkes-Dodson law in psychology, which describes the relationship between human arousal and performance. Based on the mutual trust model, we first study a simple case of one human operator controlling a single robot and propose a trust-triggered control strategy depending on the limit conditions of the desired trust region. The method is then enhanced for the case of one human operator controlling a swarm of robots. In this framework, a periodic trust-based control strategy with a highest-trust-first scheduling algorithm is proposed. Matlab simulation results are provided to validate the proposed model and control strategies that guarantee effective real-time scheduling of teleoperated and autonomous controls in both one human one underwater robot case and one human multiple underwater robots case.

Keywords: Human-Robot Interaction, Mutual Trust, Real-Time Scheduling, (Semi) autonomous Underwater Robots.

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014,*
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_6,

1 Introduction

(Semi)autonomous underwater robots have been widely used for military and civilian applications such as seafloor mapping, ocean exploration, fisheries and plankton-acoustics research, underwater mine detection and etc. However, current approaches are limited to multiple human operators controlling a single (semi)autonomous underwater robot. The high level of manpower required to operate such a (semi)autonomous underwater vehicle inevitably leads to high labor costs as well as human errors.

As the autonomy technology advances, the number of human operators per underwater robot will be significantly reduced [7] within the very near future. To enable a single human operator controlling one or multiple robots, it is of particular importance to build mutual trust between a human operator and underwater robots. On one hand, the human operator must trust the capability of robots and allow them to operate autonomously under normal conditions. On the other hand, a robot must also build its trust on a human operator based on the operator's performance so that the robot can follow human guidance accordingly. This is especially crucial for the monitoring and control of underwater robots since underwater communication is more challenging than that of ground and ariel robots.

In this paper, we propose a novel model for the mutual trust between the human operator and (semi)autonomous underwater robots. This model is inspired by the time series trust model proposed in [19, 29]. Different from the unilateral human-to-robot trust in [19, 29], since we consider the reciprocal trust between human and robot partners, our model includes both robot performance, human performance, and overall fault rates of the human-robot systems. The robot performance will be modeled based on impact of human's neglect (respectively, intervention) on the robot and evaluated in autonomous (reps. teleoperated) mode. In particular, we obtain the performance model of a YSI EcoMapper autonomous underwater robot based on its distance to a desired waypoint, which will be used in the simulation section 5. The human performance model establishes a mathematical description of the Yerkes-Dodson law in psychology, which is a function of both human workload and task difficulty.

Based on the mutual trust model, trust-based algorithms are developed to guarantee effective real-time scheduling of teleoperated and autonomous control of the underwater robots. To achieve this objective, we first investigate the case of one human operator controlling a single robot. A trust-triggered control strategy depending on the limit conditions of the desired trust region is obtained. Further, we consider the case of one human operator controlling a swarm of robots. To overcome the limitations of the trust-triggered control strategy for a single robot, the real-time scheduling algorithms [35, 39] are adapted for the allocation between teleoperated and autonomous control mode for multiple robots. A periodic trust-based control strategy with a highest-trust-first scheduling algorithm is proposed such that the mutual trust can be maintained at a desired level for each human-robot pair.

The main contributions of the chapter are two fold. First of all, we systematically model the human robot mutual trust based on a time series model. While the existing works in the literature have been mainly focused on human-to-robot trust and are mostly qualitative [17, 24], our model captures the mutual trust between human and robot partners, which is especially important for underwater robots applications. Second, we

develop trust-based control and real-time scheduling algorithms which guarantee desired trust level for the human-robot team. The mutual trust study is especially relevant for distributed multi-robot systems or mobile sensing networks where one human operator is asked to interact with networked robots.

The outline of the chapter is as follows. In Section 2, we introduce the related works in human robot interaction (HRI), trust in HRI, and related research in supervisory control theory. In Section 3, we propose a time series mutual trust model as a function of robot performance, human performance, and fault rate. The robot and human performance models are then discussed in detail. Section 4 discusses the control strategies to drive the human-robot mutual trust to the desired trust region. We first develop a trust-triggered control strategy based on the mutual trust model for the human operator to control one single (semi)autonomous underwater vehicles in Section 4.1. We then extend the case to multi-robot team in Section 4.2. A periodic control strategy with highest-trust-first scheduling algorithm is developed. Section 5 provides a set of Matlab simulation results. We conclude the paper in Section 6.

2 Related Works

Research in human-robot interaction (HRI) has grown rapidly in the last two decades. The seminal book [32] gives a good introduction of much of the early work in human and computer systems interactions. The book [16] provides a survey on the foundations and trends of HRI. It gives a unified treatment of HRI-related problems, reviews the history and formation of the HRI field, identifies key themes and solution approaches, discusses challenge problems that are likely to shape the field in the near future, and related fields such as teleoperation, automation science, human-computer interaction, artificial intelligence and so on. The book [20] gives an overview of HRI in advanced manufacturing systems with highlights on three broad areas including human factors, safety, design and implementation. Multi-disciplinary problems in HRI such as operator cognitive, robot software, industrial robot capabilities, organizational and management structures, and safety are discussed. The book [33] deals with human-automation systems from design to control and performance of both humans and machines. The special issue on HRI in the IEEE Transactions on Robotics [23] presents a wide range of HRI related research in robotics including multimodal ways by which humanoids interact with human beings, the design of a robot head expressing emotions, navigation algorithms for a mobile humanoid robot, control solutions for humanoid and assistive robots, human robot interfaces, multiple human-robot team interactions, human subject testings, and the evaluation for HRI methods. The special issue on HRI in the International Journal of Social Robotics [18] presents recent advances in understanding the expectations, intentions, and actions of both users and robots, and designing appropriate robot behaviors based on its understanding of the world. Several international conference proceedings are dedicated to HRI research such as [22] for human-robot personal relationships, [15] for social robotics, [1] for robot and human interactive communication, and [21] for general HRI research. HRI research has immersed in almost all aspects of robotics and automation and is of significant importance to real life applications. On one hand, robots can extend human capabilities and compensate for human

limitations, especially in some hazardous environments. On the other hand, human has superior abilities to handle unanticipated and poorly addressed scenarios, and hence provides good guidance to semiautonomous and autonomous robots. Therefore, the integration of human and robot capabilities will greatly improve system performance in demanding tasks.

In particular, the mutual trust between humans and robots is fundamental for effective HRI in order to fulfill tasks cooperatively and successfully [3, 17]. Humans and robots act like partners in this shared relationship and their mutual trust affects how each of the members behaves. The human robot mutual trust is a reciprocal relationship and includes both human-to-robot trust and robot-to-human trust. Human-to-robot trust involves understanding capabilities of a robot and allowing a robot to perform certain tasks autonomously. It affects the willingness of the human operator to accept robot-produced information and thus benefit from the advantages inherent in robotic systems [17]. The analysis on human-to-robot trust is especially useful for operating multiple robots simultaneously [3]. The trust level affects the difference between neglect time and activity time of a robot, which has an important impact on multiplexing human attention among multiple robots. Robot-to-human trust is based on understanding the proficiency of human operator, establishing trust according to this understanding, and following guidance while maintaining a certain level of autonomy. Humans and robots will either gain or lose trust based on the progress of the task [24]. Over-trust often leads to misuse of the robot and human resources and hence causes decreasing effectiveness during the execution of a desired task. Contrarily, under-trust will put limitation on the autonomy of robots and underestimates human's capability. Hence, ensuring an appropriate level of human robot mutual trust can be a challenge.

Existing studies on the trust in HRI have mostly been qualitative and descriptive. More generally, trust in robot is very much related to trust in automation [24, 33]. Qualitative analysis has been performed in [17] to evaluate the effects of human, robot and environment factors on perceived trust in HRI. A collaborative performance model is developed in [14] to capture the critical performance attributes of human-robot trust. Furthermore, a decision-analytical based measure of trust is developed, which is given by the ratio of the expected loss to the number of operator overrides.

Despite of the modeling effort, most of the literature has been focused on unilateral human's trust to robots. However, in a collaborative operation where humans and robots work together with each other as partners to complete a task efficiently, the trust between humans and robots should be bilateral and an appropriate level of mutual trust will eventually yield the best collaborative performance. To model human-robot mutual trust, the analogy on human-animal trust has been drawn to model human-robot trust based on experimental data. In [6], two separate notions are employed to determine human-robot trust by analogous to human-dog trust, i.e., (i) knowing how a partner will respond and (ii) trusting oneself to interpret a partner's behavior. The paper [10] argues that different contexts and perception can be applied to study human-robot relations by comparisons with human-animal relations. Related research includes mixed initiative interaction where the key is not only the ability of the human to understand and predict robot performance, but also the robot's ability to identify human needs and select intervention points to assume different levels of initiative [8].

Apart from the above qualitative works, a time series trust model has been proposed to characterize the quantitative dynamic relationship between human-to-robot trust and corresponding factors. A time-series model of human-to-robot trust and the human operator's self-confidence is proposed in [29]. In [19], the relationship between the changes in human's control strategy and trust to a semi-automatic pasteurization plant is investigated using subjective rating scales.

Related works in supervisory control refer to intermittent human interaction with a remote, automated system in order to manage a controlled process or task [34]. Several works discuss about the supervisory control for teleoperation of multi-agent coordination, which involve the control of a lead agent or the center of mass of the formation by an external human operator [4, 13]. In [9], a human supervised robot forage problem is proposed where the two alternative, forced choice human decision-making task is considered. The use of human supervisory control for swarming networks was discussed in [12]. Multi-operator supervisory control has been studied in [27, 37].

However, there still lacks systematic modeling of the dynamic evolution of mutual trust between human and robot partners, especially for the case when a human operator is required to monitor and control multiple robots at the same time. This then motivates the current chapter. The multi-robot case further raises the problem of real-time scheduling and allocation of human interaction with each robot, which we will address in more details in Section 4.

3 Dynamic Model of Mutual Trust

In this section, we first model the mutual trust between the human operator and one (semi)autonomous robot. Inspired by the time series human-to-robot trust model [19, 25, 29], we propose the following dynamic model for human-robot mutual trust

$$T(t) = AT(t-1) + B_1 P_R(t) + B_2 P_R(t-1) + C_1 P_H(t) + C_2 P_H(t-1) + D_1 F(t) + D_2 F(t-1), \quad (1)$$

where $A, B_i, C_i, D_i,\ i = 1, 2$ are real constant coefficients and their values depend on the tasks to be performed as well as the specific robot and human operator. The current mutual trust level $T(t)$ is determined by the previous trust level $T(t-1)$, current and previous performance of the robot $P_R(t), P_R(t-1)$, current and previous performance of the human operator $P_H(t), P_H(t-1)$, and current and previous fault rate $F(t), F(t-1)$ under autonomous or teleoperated mode, respectively. The faults may include machine malfunction or human error. Let T_d be the trust level adopted by the expert human operator, which is gained from experiments. T_u and T_l are the upper and lower limit of the desired trust region such that $T_l < T_d < T_u$.

3.1 Robot Performance Model

The robot performance model adopted in this paper is inspired by the qualitative analysis in [11]. The model is based on the following assumptions derived from experimental

data: i) the likely performance of a robot will gradually degrade due to accumulated error when a human operator neglects the robot (as shown in Figure 12(a)), and ii) the likely performance of a robot will increase as a human operator interacts with the robot (as shown in Figure 12(b)). Hence, we can assume that a robot working in two modes: the autonomous mode, and the teleoperated mode, with different performance models given by the following two difference equations.

$$P_R(t) = \begin{cases} (1-k_R)P_R(t-1) + k_R P_{R,\min}, & \text{autonomous mode} \\ (1-k_H)P_R(t-1) + k_H P_{R,\max}, & \text{teleoperated mode} \end{cases}, \tag{2}$$

where $P_{R,\max}, P_{R,\min} \in [0,1]$ stand for the maximum and minimum robot performance, and $k_R, k_H \in (0,1)$ is the performance coefficients for autonomous mode and teleoperated mode, respectively. The robot performance model (2) guarantees that P_R is bounded between $[P_{R,\min}, P_{R,\max}]$. We acknowledge that this model may not apply to all robots and all applications. More sophisticated models can be derived which depend on specific robots and applications. Nevertheless this model provides a starting point to study mutual trust between human and robots. Next, we will show that this model can be used to model the waypoint-based navigation performance of the YSI EcoMapper.

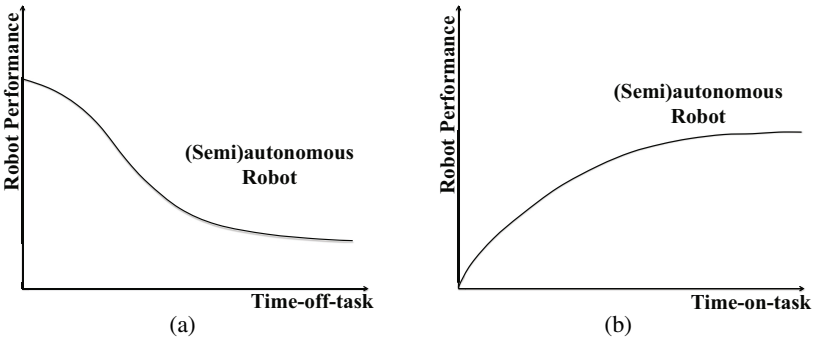


Fig. 1. (a) Impact of human neglect (time-off-task) on the robot performance, and (b) Impact of human intervention (time-on-task) on the robot performance. This figure is inspired by the case of semiautonomous robots in [11]. However, in [11], the performance for autonomous robots is independent of human control.

The EcoMapper, as shown in Figure 2, is a versatile autonomous underwater vehicle (AUV) equipped with water quality sensors for aquatic environmental monitoring applications. It can operate in autonomous mode or teleoperated mode [30]. A dynamical model of the EcoMapper has recently been obtained though computational fluid dynamics analysis and experimental efforts [36]. Motion of AUVs like the EcoMapper are subjected to disturbances from ocean current.

When an AUV navigates autonomously to a waypoint, the error between the desired and actual position will be driven to zero so that the AUV can arrive at the destination point. Although the autonomous navigation is more accurate, the AUV usually takes

Fig. 2. YSI EcoMapper AUV

more time to adjust its orientation and thus spends more time to reach the destination. Therefore, human interventions are often necessary.

The dynamics of the EcoMapper can be expressed by the following equations

$$\begin{aligned} M\dot{\mathrm{V}} + C(\mathrm{V})\mathrm{V} + D(\mathrm{V})\mathrm{V} &= \tau \\ \dot{\eta} &= J(\eta)\mathrm{V} \end{aligned} \tag{3}$$

where $\eta = [x, y, z, \phi, \theta, \psi]^T$ represents the AUV position and Euler angles, $V = [u, v, w, p, q, r]^T$ is the vector of linear and angular velocities, τ is the vector of external forces and moments. M is the inertia matrix, $C(\mathrm{V})$ is the matrix of Coriolis and centripetal terms, $D(\mathrm{V})$ is the damping matrix, and $J(\eta)$ is a transformation matrix from the body-fixed frame to the earth-fixed frame [36].

For the sake of simplicity, we consider only the planar motion of the EcoMapper. The first order Euler approximation can be used to obtain the discrete-time dynamics. Assume that the EcoMapper is initially at some point A and its destination is point B. Define the distance between the current position at time t and the destination as $D(t)$. We can use either the autonomous or teleoperated mode to drive the EcoMapper to its destination point B.

In the teleoperated mode, the human operator can quickly adjust the heading of the EcoMapper as well as stop the EcoMapper when it is close to the destination point B. Therefore, the human operator is able to control the EcoMapper so that it goes towards the destination at full speed $u_{\max}$. However, because human's observation of speed and distances are not very accurate and that there is no automatic compensation to the external disturbances (e.g., wind, water current) applied to the EcoMapper, the actual speed executed by the EcoMapper may not be constant. Hence, we assume that the actual speed of the EcoMapper is given by $u_{\max} - \delta_1$, where δ_1 represents a random noise. In other words, the actual speed of the EcoMapper fluctuates around $u_{\max}$, and δ_1 represents the perturbation applied to the EcoMapper speed. Hence, in this case, we have $D(t) = D(t-1) - (u_{\max} - \delta_1)$. When the distance $D(t) < 2$m, the human operator will deem that the EcoMapper has reached its destination and does not apply control on it any more. Consequently, the position of the EcoMapper will only be affected by some disturbances. Therefore, perturbation is added to the velocity components of the AUV dynamics to simulate the disturbances and we use δ_2 to denote this perturbation. In summary, the distance function $D(t)$ under the teleoperated mode is given as follows:

$$D(t) = \begin{cases} D(t-1) - (u_{\max} - \delta_1), & \text{if } D > 2 \\ D(t-1) + \delta_2, & \text{if } D \leq 2 \end{cases}$$

In the autonomous mode as shown in Figure 3, the EcoMapper is controlled by a self-docking controller which controls the vertical fin angle $\alpha(t)$ and the propeller rotation speed $n(t)$. Assume at some time t, the position of the EcoMapper is $[x(t); y(t)]$ and the heading angle is $\Phi(t)$. Let the desired heading angle at time t be $\Phi_d(t) = \arctan\frac{y(t)-y_d(t)}{x(t)-x_d(t)}$. That is, the heading of the EcoMapper needs to be turned by $\Phi_d(t) - \Phi(t)$. Therefore, the vertical fin angle should be $\alpha(t) = k_1(\Phi_d(t) - \Phi(t))$, where $k_1 > 0$ is a control gain. Because we want to make sure that the EcoMapper stops when it reaches the destination point B, the propeller rotation speed is set as $n(t) = k_2 D(t)$, where $k_2 > 0$ is another control gain.

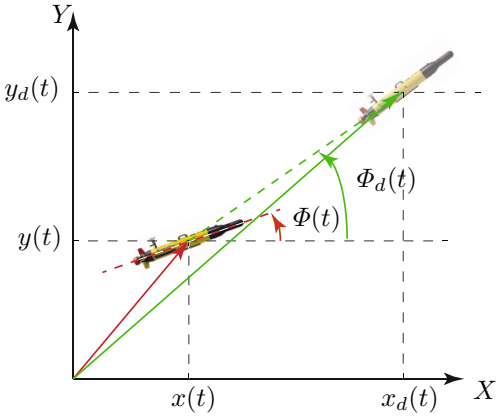


Fig. 3. Illustration of the autonomous mode

The path $D(t)$ traveled by an EcoMapper can be computed from dynamics (3). Let the initial position be $[0, 0]$ and destination be $[x_d; y_d] = [50\text{m}; 0]$. For the teleoperated mode, we set $u_{\max} = 2$m/s, $\delta_1 \in (-0.1, 0.1)$, and $\delta_2 \in (-0.05, 0.05)$. For the autonomous mode, we set $k_1 = 0.5$ and $k_2 = 60$. Because there are some constraints on the vertical fin angle and propeller rotation speed, we assume that the limits of the vertical fin angle are 35° and -35° and the limit for the propeller rotation speed is 2000rpm.

Based on the data $D(t)$, we can determine the parameters in the robot performance model which result in a best match. Under the teleoperated mode, define $P_R(t) = \frac{D_{\max}-D(t)}{D_{\max}}$, where $D_{\max}$ is the maximum distance traveled by the EcoMapper. We can determine the robot performance model under the teleoperated mode as represented by Equation (2) by setting $P_{R,\max} = 0.96$ and $k_H = 0.09$. Under the autonomous mode, define the normalized distance to the destination point B as the robot performance, i.e., $P_R(t) = \frac{D(t)}{D_{\max}}$. The robot performance model (2) under the autonomous mode can be determined by setting $k_R = 0.05$ and $P_{R,\min} = 0.15$. Figures 4(a) and 4(b) plot the actual robot performance and its corresponding model approximation under the teleoperated and autonomous mode, respectively.

3.2 Human Performance Model

The Yerkes-Dodson law [38] describes human performance as an empirical model with respect to human arousal, which is always assumed to be proportional to the workload

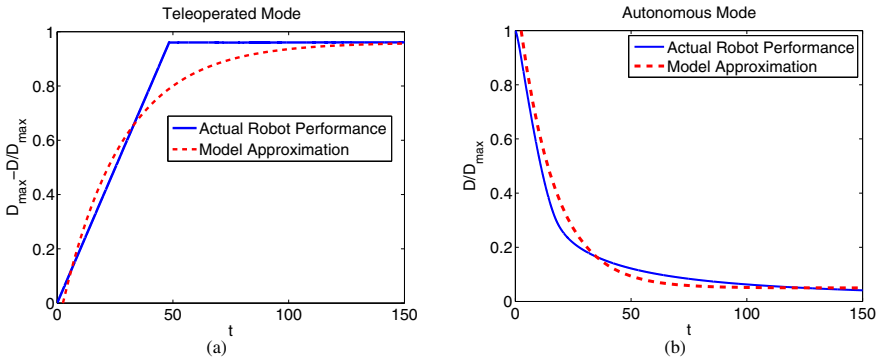


Fig. 4. The YSI EcoMapper performance under (a) the teleoperated mode, and (b) the autonomous mode

[5]. Here human performance means the capability and efficiency of the human operator to perform a given task. In this paper, we represent the human workload as an utilization function that is formally defined as follows.

Definition 1. *At any time t, the utilization $r(t)$ is defined as the ratio of the total time when a human operator is performing some task (versus free) within the time interval $[0, t]$ to the entire time length t.*

Based on the above definition, we can obtain the following mathematical model for human performance

$$P_H(t) = \left(\frac{r(t)}{\beta}\right)^{\beta} \left(\frac{1 - r(t)}{1 - \beta}\right)^{1-\beta}, \tag{4}$$

where $\beta \in (0, 1)$ represents the difficulty of the task for human and a smaller value of β represents a more difficult task. Figure 5 illustrates the effect of utilization r and difficulty of the task β on human performance P_H. Note that the human performance model (4) guarantees that P_H is bounded between $[0, 1]$. We can observe from the inverted U-shape curve that the performance of human increases with utilization at the beginning. However, when the level of utilization r exceeds β, i.e., a moderate level of workload and stress, the human performance will drop. Furthermore, comparing the human performance under different task difficulty β, we can observe that the level of utilization for optimal performance decreases when the difficulty of task increases [28]. For example, for a relatively difficult task with a small value of β, the peak human performance occurs with a low workload and arousal level. However, as the time increases, the human operator easily gets more stressed with more workload and the corresponding human performance decreases.

According to Definition 1, the total "busy" time, i.e., the workload of a human operator within $[0, t-1]$ is $(t-1)r(t-1)$. If the robot is in the autonomous mode within

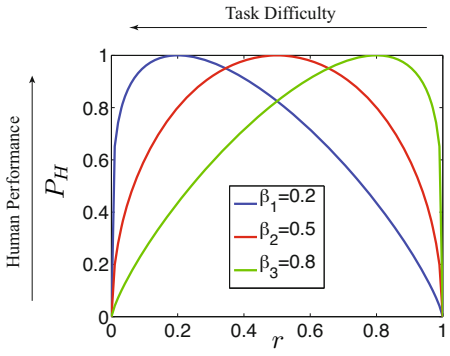


Fig. 5. Human Performance

$[t-1, t]$, the workload of a human operator within $[0, t]$ will remain $(t-1)r(t-1)$. Thus, we have the utilization $r(t)$ in the autonomous mode as

$$r(t) = \frac{(t-1)r(t-1)}{t} = (1 - \frac{1}{t})r(t-1). \tag{5}$$

On the other hand, if the robot is in the teleoperated mode within $[t-1, t]$, the workload of a human operator within $[0, t]$ will be $(t-1)r(t-1)+1$. Thus, we have the utilization $r(t)$ in the teleoperated mode as

$$r(t) = \frac{(t-1)r(t-1)+1}{t} = (1 - \frac{1}{t})r(t-1) + \frac{1}{t}. \tag{6}$$

According to Equations (5) and (6), the utilization $r(t)$ in both modes is as follows

$$r(t) = r(t-1) + \frac{u(t) - r(t-1)}{t}, \tag{7}$$
$$u(t) = \begin{cases} 1 & \text{teleoperated mode} \\ 0 & \text{autonomous mode} \end{cases},$$

where $u(t)$ denotes the control mode of the robot. Note that the utilization $r(t)$ increases in teleoperated mode and decreases in autonomous mode. However, it is always bounded between the minimum utilization ratio 0 and the maximum utilization ratio 1.

This model is especially useful in the case when one human operator supervises multiple robots and the workload of the human is given by the summation of all the control efforts allocated to different robots.

4 Trust-Triggered Control and Real-Time Task Allocation Strategies

4.1 Single Robot Control Strategy

In this section, we first develop a simple trust-triggered control strategy based on limit conditions to drive the mutual trust between the human operator and one single robot

to a desired trust region determined by the expert trust level. As shown in Figure 6, based on the EcoMapper dynamics (3) and the current mode, the robot performance (2) can be determined. The human performance (4) is based on the current utilization r and task difficulty β. The mutual trust (1) can be computed from the robot and human performance in the presence of faults. We design trust-triggered control strategy to dynamically switch between the teleoperated mode and autonomous mode based on the information from the mutual trust.

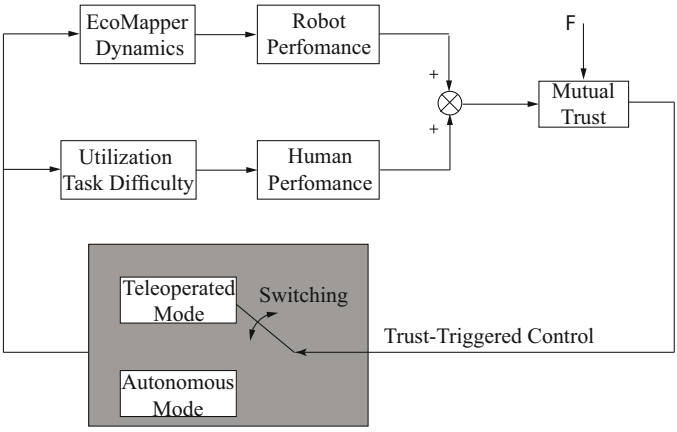


Fig. 6. Trust-Triggered Control Strategy

Define a desired trust region determined by the expert trust level T_d as $[T_l, T_u]$. Let the triggering conditions correspond to the upper limit T_u and lower limit T_l of the trust region $[T_l, T_u]$. When the trust level exceeds the upper limit, i.e., $T(t) \geq T_u$, the system reaches the 'over-trust' situation and teleoperation $u(t) = 1$ will be adopted. On the other hand, when the trust level goes below the lower limit, i.e., $T(t) \leq T_l$, the system reaches the 'under-trust' situation and automatic control $u(t) = 0$ will be adopted. Otherwise, the control scheme at the previous time step will be used. Hence, the above trust-triggered control strategy can be represented as

$$u(t) = \begin{cases} 1, & T(t) \geq T_u \\ 0, & T(t) \leq T_l \\ u(t-1), & \text{Otherwise} \end{cases}. \tag{8}$$

The trust-triggered control strategy in Equation (8) is designed to drive the human-robot mutual trust to the desired trust region in the presence of faults. Figure 7 shows an illustrative example of the above trust-triggered control strategy when the desired trust level is within range of $T(t) \in [1.5, 2]$ with a constant desired trust level $T_d = 1.75$ and fault rate modeled as a Gaussian white noise with standard deviation 0.004.

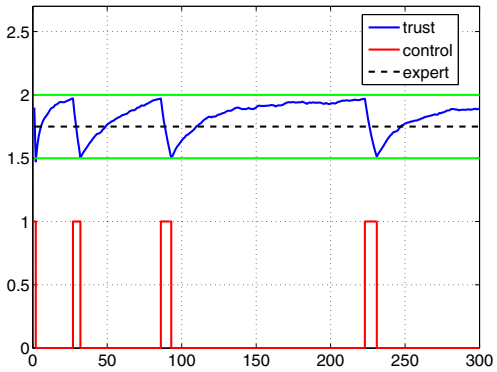


Fig. 7. Illustration of trust-triggered control strategy based on limit conditions

Table 1 summarizes the trust-triggered control strategy.

Table 1. Summary of trust-triggered control strategy

	Autonomous Mode $u(t) = 0$	Teleoperated Mode $u(t) = 1$
Robot PRFM. $P_R(t)$ (Eqn. (2))	$(1 - k_R)P_R(t-1) + k_R P_{R,\min}$	$(1 - k_H)P_R(t-1) + k_H P_{R,\max}$
Human PRFM. $P_H(t)$ (Eqns (4), (7))	$P_H(t) = \left(\frac{r(t)}{\beta}\right)^{\beta}\left(\frac{1-r(t)}{1-\beta}\right)^{1-\beta}$ $(1 - \frac{1}{t})r(t-1)$	$(1 - \frac{1}{t})r(t-1) + \frac{1}{t}$
Mutual Trust $T(t)$	Eqn. (1)	
Triggering Condition	$T(t) \leq T_l$	$T(t) \geq T_u$

4.2 Multi-robot Allocation and Real-Time Scheduling

We now extend the above results to the case when a human operator monitors and controls a team of robots $\{R_1, ..., R_N\}$. Let R_n denote an individual robot in the team. According to the mutual trust dynamic model in Equation (1), the mutual trust level between the human operator and each robot R_n can be easily defined as follows

$$\begin{aligned} T_n(t) = & A_n T_n(t-1) + B_{n,1} P_{n,R}(t) + B_{n,2} P_{n,R}(t-1) + C_{n,1} P_H(t) + \\ & C_{n,2} P_H(t-1) + D_{n,1} F_n(t) + D_{n,2} F_n(t-1) \end{aligned} \tag{9}$$

where the subscript n denotes the index of robots, $T_n(t)$ the mutual trust between the human operator and R_n, $P_{n,R}(t)$ the performance of R_n, P_H the human performance, $F_n(t)$ the fault rate of R_n, and A_n, $B_{n,1}$, $B_{n,2}$, $C_{n,1}$, $C_{n,2}$, $D_{n,1}$ and $D_{n,2}$ are constant coefficients depending on characteristics of each robot R_n. Moreover, for each robot R_n, we use $T_{n,d}$ to denote the trust level adopted by the expert human operator, $T_{n,u}$ denote the upper limit of the desired trust region, and $T_{n,l}$ denote the lower limit of the desired trust region. As disccused in Equation (4), the human perfomrnace $P_H(t)$

is determined by the utilization ratio $r(t)$. In the case of multiple robots, the utilization ratio $r(t)$ depends on the interactions with all robots. Therefore, we extend the definition of $r(t)$ as follows

$$r(t) = r(t-1) + \frac{\sum_{n=1}^{N} u_n(t) - r(t-1)}{t}, \tag{10}$$
$$u_n(t) = \begin{cases} 1 & \text{teleoperated mode} \\ 0 & \text{autonomous mode} \end{cases},$$

where $u_n(t)$ denotes the control mode of robot R_n. As shown in Equation (10), the utilization ratio $r(t)$ is determined by control modes of all robots. It is assumed that the human operator can only interact with one robot at a time, therefore, the utilization ratio $r(t)$ is still bounded between 0 and 1.

The goal of controlling a team of robots is similar to the goal of controlling a single robot, as introduced in Section 4.1. The human operator must use appropriate control strategy to achieve the desired mutual trust with each robot in the team. Since the human operator can only interact with one robot at a time, the trust-triggered control strategy developed for a single robot in Section 4.1 cannot be directly applied here for the following two reasons. First, in the trust-triggered control strategy, the human operator will start to interact with the robot once the mutual trust level reaches the upper limit of the desired trust region. However, for the case of multiple robots, this control strategy will fail when more than one robot reach their respective upper limits at the same time. Second, in the trust-triggered control strategy, once the human operator starts to interact with a robot, this interaction will not stop until the mutual trust level falls below the lower limit of the desired trust region. However, for the case of multiple robots, such long time of interaction with one robot could waste the capability of human operator, which should have been spent on other robots that are in more urgent need of interaction. To overcome the limitations of the trust-triggered control strategy for a single robot, the human operator needs a new trust-based control strategy for controlling multiple robots. This new strategy must satisfy two requirements. First, it should guarantee that each robot can receive a fair share of interaction with the human operator. Second, no more than one robot will reach the upper limit of desired trust region at the same time.

Here we propose a periodic trust-based control strategy. In this new control strategy, the human operator must interact with each robot R_n for I_n amount of time within every period of L_n. For each robot R_n, the choice of L_n is a fixed value that depends on the desired trust region $[T_{n,l}, T_{n,u}]$. A wider trust region allows for a larger choice of L_n. The choice of I_n is a little bit involved and will dynamically change according to the mutual trust level within the previous period. Algorithm 1 discusses the detailed implementation for updating I_n at the beginning of each period. First, we calculate the maximum and minimum mutual trust level within the previous period, as shown in Line 1 and Line 2. When the mutual trust goes too high, we increase the amount of interaction time I_n, as shown in Line 3 and Line 4. Note that the value of I_n cannot go beyong L_n. On the other hand, when the mutual trust goes too low, we decrease the amount of interaction time I_n, as shown in Line 5 and Line 6. Note that the value of I_n cannot be smaller than zero. Note that ϵ_1 and ϵ_2 are arbitrarily small values guaranteeing

that I_n is adjusted before the mutual trust level goes beyond upper and lower limit. Figure 8 shows an example of periodic trust-based control strategy between the human operator and one robot. As we can see, the mutual trust can always stay within the desired region $[1.5, 2]$. Moreover, comparing Figure 8 with Figure 7, we can see that the mutual trust under the periodic trust-based control strategy is closer to the expert level than trust-triggered control strategy. This is because the human operator under the periodic trust-based control strategy may switch between the autonomous and teleoperated mode frequently even before the mutual trust reaches the limits of the desired region. Note that the value of I_n depends on the range of desired trust region. The higher the trust region is, the smaller the value of I_n is.

Algorithm 1. Update_Task

Data: $T_n(t)$, $\{I_n, L_n\}_{n=1}^N$
Result: $\{I_n, L_n\}_{n=1}^N$

for *each* $\tau_n \in \Gamma$ **do**
  $T_{\mathrm{n,max}} = \max_{t-L_n \le t \le t} T_n(t)$;
  $T_{\mathrm{n,min}} = \min_{t-L_n \le t \le t} T_n(t)$;
  if $T_{\mathrm{n,max}} > T_{n,u} - \epsilon_1$ or $T_{\mathrm{n,min}} > T_{n,d}$ **then**
    $I_n = \min\{I_n + 1, L_n\}$;
  else if $T_{\mathrm{n,min}} < T_{n,l} + \epsilon_2$ or $T_{\mathrm{n,max}} < T_{n,d}$ **then**
    $I_n = \max\{I_n - 1, 0\}$;
return $\{I_n, L_n\}_{n=1}^N$;

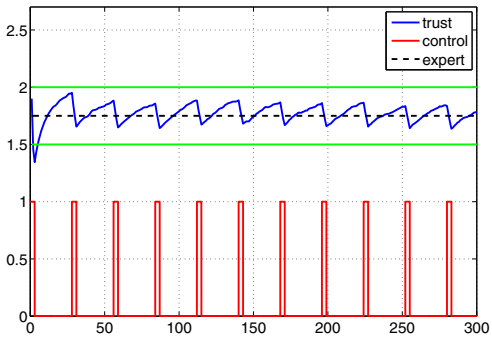


Fig. 8. An example of periodic trust-based control strategy between the human operator and one robot

Based on the above analysis, we can conclude that controlling a team of robots shares similarities to scheduling a set of periodic tasks on a single processor, which is a classic research topic in real-time scheduling [2, 26, 31]. More specifically, the control of each robot R_n can be understood as the execution of an individual task τ_n, and the human operator can be understood as the single processor as he/she can only control one robot at a time. Therefore, controlling a set of robots $\{R_1, \cdots, R_n\}$ corresponds to executing a set of tasks $\Gamma = \{\tau_1, ..., \tau_N\}$. However, the new challenge here is that the time I_n required to interact with a robot R_n within each period L_n will dynamically change according to the robot and human performance. This case has not been studied in real-time scheduling to the best of our knowledge. Because the human operator can only interact within one robot at a time, we now propose a highest-trust-first scheduling algorithm to schedule the interaction with different robots. In the highest-trust-first scheduling algorithm, the human operator always chooses to interact with the robot that has the highest mutual trust level and non-zero remaining interaction time. According to Equation (9), we know that the calcuation of mutual trust level depends on the human performance $P_H(t)$, which in turn depends on the utilization ratio $r(t)$ as show in Equation (4). The value of $r(t)$ reflects the accumulative operation of the human operator from time 0 to current time t. Therefore, the value of $r(t)$ needs to be continuously updated as time propagates. Moreover, the remaining interaction time of each robot also needs to be dynamically updated as time propagates. To achieve these goals, we adopt the dynamic timing model for real-time scheduling proposed in [35, 39]. The dynamic timing model contains a set of state variables that represent the dynamic updates of system status according to the evolution rules decided by the scheduling policy. The remaining interaction time of each robot can be defined as one of the state variables in the model and gets updated dynamically. After updating these state variables in the dynamic timing model, we obtain the human utilization ratio r accordingly. We now define three state variables that completely represent the current status of robots at any time t.

Definition 2. *Define the dynamic arrival as $Q(t) = [q_1(t), ..., q_N(t)]$, where any $q_n(t) \in Q(t)$ denotes how long after t the next instance of τ_n will arrive. For any task τ_n, we have the evolution of $q_n(t)$ as follows*

$$q_n(t+1) = \begin{cases} q_n(t) - 1, & q_n(t) > 0 \\ L_n, & q_n(t) = 0 \end{cases}. \tag{11}$$

Definition 3. *Define the residue as $S(t) = [s_1(t), ..., s_N(t)]$, where any $s_n(t) \in S(t)$ represents the remaining interaction time required after time t by the current instance of τ_n. For any task τ_n, we have the evolution of $s_n(t)$ as follows*

$$s_n(t+1) = \begin{cases} s_n(t) - 1, & u_n(t) = 1, q_n(t) \neq 0 \\ s_n(t), & u_n(t) = 0, q_n(t) \neq 0 \\ I_n, & q_n(t) = 0 \end{cases}. \tag{12}$$

Definition 4. *Define the gap as $E(t) = [e_1(t), ..., e_N(t)]$, where any $e_n(t) \in E(t)$ represents the difference between the current trust level $T_n(t)$ of the robot R_n and its desired trust upper limit $T_{n,u}$, i.e., $e_n(t) = T_{n,u} - T_n(t)$.*

Note that the higher the trust is, the smaller the value of $e_n(t)$ is. The choice of I_n is designed in Algorithm 1 such that $T_n(t)$ will not exceed $T_{n,u}$ and hence $e_n(t)$ is always positive. When the mutual trust level associated with the robot is higher, the human operator has the trend to over trust the robot. Therefore, the highest-trust-first scheduling algorithm states that the smaller $e_n(t)$ is, the more urgent the human needs to manually control the robot R_n. Based the state variables $Q(t), S(t), E(t)$, we can schedule the human operator's interaction with a team of robots according to Algorithm 2. At each time step t, the inputs to Algorithm 2 are the current system status including the human and robot performance $P_H(t)$, $\{P_{n,R}(t)\}_{n=1}^N$, human utilization ratio $r(t)$, mutual trust $\{T_n(t)\}_{n=1}^N$, state variables of the dynamic timing model $Q(t), S(t), E(t)$, and the period and interaction time $\{L_n, I_n\}_{n=1}^N$ obtained from Algorithm 1. We define a set G which is a set of tasks with the non-zero remaining interaction time s_n. If G is not an empty set, the robot with minimum difference e_n will be chosen for teleoperation. The outputs of Algorithm 2 are the updated system status $P_H(t+1)$, $\{P_{n,R(t+1)}\}_{n=1}^N$, $r(t+1)$, $\{T_n(t+1)\}_{n=1}^N$, $Q(t+1)$, $S(t+1)$, $E(t+1)$ and the scheduling decision $\{u_n(t+1)\}_{n=1}^N$. If the human operator is scheduled to interact with the robot R_n at time t, $u_n(t+1) = 1$ and if the human operator is scheduled NOT to interact with the robot R_n at time t, $u_n(t+1) = 0$.

We now summarize the periodic trust-based control strategy composed of Algorithm 1 and 2. At every time step t, we first use Algorithm 1 to evalute the choice of I_n for each task τ_n by checking whether the mutual trust $T_n(t)$ falls within the desired trust region $[T_{n,l}, T_{n,u}]$ during the time interval $t \in [t - L_n, t]$. If not, the value of I_n will be adjusted accordingly. To be more specific, Algorithm 1 can guarantee the proper choice of I_n so that $T_n(t)$ will not exceed $T_{n,u}$ or go below $T_{n,l}$. Hence, $e_n(t)$ is always positive. Given the output $\{I_n, L_n\}_{n=1}^N$ of Algorithm 1, we use Algorithm 2 to compute the mutual trust $T_n(t+1)$ for each robot and then decide which is the next robot the human operator will interact with. The above process repeats as time propagates.

5 Simulation Results

In this section, we will show that the periodic trust-based control strategy can effectively maintain the mutual trust between the human operator and robots within a desired region.

5.1 Simulation Setup

We consider the case when a human operator controls three EcoMapper AUV robots $\{R_1, R_2, R_3\}$. All three robots are asked to perform station keeping using switched autonomous and teleoperation strategy. Their performances are measured by the distance (in meters) to the goal points as described by Equation (2). Since the AUVs may have different sensor packages installed, we assume the three robots have different parameters as listed in Table 2. Each robot has its initial performance as $[P_{1,R}(0), P_{2,R}(0), P_{3,R}(0) = [0.18, 0.25, 0.21]$. The human operator has his/her performance as described by Equation (4). We assume that the task difficulty for the human operator is $\beta = 0.8$. The initial human performance is $P_H(0) = 0.25$ and the initial utilization ratio is $r(0) = 0.1$.

Algorithm 2. Highest-Trust-First Scheduling Algorithm

Data: $r(t)$, $P_H(t)$, $\{P_{n,R}(t), T_n(t), q_n(t), s_n(t)\}_{n=1}^N$, $\{L_n, I_n\}_{n=1}^N$

Result: $\{u_n(t{+}1)\}_{n=1}^N$, $r(t{+}1)$, $P_H(t{+}1)$, $\{P_{n,R}(t{+}1), T_n(t{+}1), q_n(t{+}1), s_n(t{+}1)\}_{n=1}^N$

/* Update $P_{n,R}$, $P_{n,H}$ and T_n*/

for *each* $\tau_n \in \Gamma$ **do**
- $P_{n,R}(t+1) \xleftarrow{\text{Eq.}(2)} \{P_{n,R}(t), u_n(t)\}$;
- $r(t+1) \xleftarrow{\text{Eq.}(10)} \{r(t), \{u_n(t)\}_{n=1}^N\}$;
- $P_H(t+1) \xleftarrow{\text{Eq.}(4)} r(t+1)$;
- $T_n(t+1) \xleftarrow{\text{Eq.}(9)} \{T_n(t), P_{n,R}(t+1), P_{n,R}(t), P_H(t+1), P_H(t)\}$;

/* Update state variables */

for *each* $\tau_n \in \Gamma$ **do**
- $q_n(t+1) \xleftarrow{\text{Eq.}(11)} \{q_n(t), L_n\}$;
- $s_n(t+1) \xleftarrow{\text{Eq.}(12)} \{s_n(t), u_n(t), q_n(t), I_n\}$;
- $e_n(t+1) = T_{n,u} - T_n(t)$;

/* G is a set of tasks with the non-zero remaining interaction time*/

$G = [\,]$;

for *each* $\tau_n \in \Gamma$ **do**
- **if** $s_n(t+1) > 0$ **then**
  - $G = [\,G,\, \tau_n\,]$;

/* Select the next task for execution*/

for *each* $\tau_n \in \Gamma$ **do**
- $u_n(t+1) = 0$;

if *G is not empty* **then**
- $i = \min_{\tau_n \in G} e_n(t+1)$;
- $u_i(t+1) = 1$;

return $\{u_n(t{+}1)\}_{n=1}^N$, $r(t{+}1)$, $P_H(t{+}1)$, $\{P_{n,R}(t{+}1), T_n(t{+}1), q_n(t{+}1), s_n(t{+}1)\}_{n=1}^N$

Table 2. Coefficients in Robot Performance Model

	k_R	k_H	P_{Rmin}	P_{Rmax}
R_1	0.15	0.09	0.05	0.96
R_2	0.12	0.09	0.02	0.85
R_3	0.15	0.12	0.05	0.95

The mutual trust between the human operator and each robot R_n follows the dynamic model discussed in Equation (9). The constant coefficients in Equation (9) are chosen as $A_n = 1, B_{n,1} = -1, B_{n,2} = 1, C_{n,1} = -1, C_{n,2} = 1, D_{n,1} = 0.002, D_{n,2} = 0.001$ and the fault rates follow the normal distribution. The initial mutual trust between the human operator and three robots are assumed to be $[T_1(0), T_2(0), T_3(0)] = [1.83, 1.8, 1.88]$. The goal of the human operator is to make sure that the mutual trust $T_n(t)$ with each robot R_n stays within a desired trust region as time propagates. In this simulation, we choose a desired trust region with the lower bound $T_{n,l} = 1.5$, the upper bound $T_{n,u} = 2$, and the ideal expert level $T_{n,d} = 1.75$ for each robot.

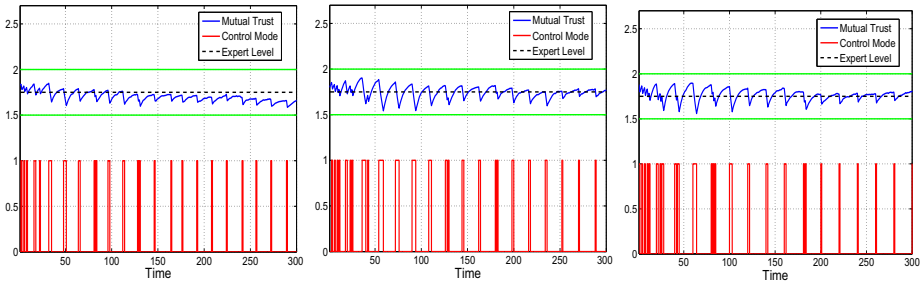
To achieve the above goal, the human operator applies our proposed periodic strategy to switch control among three robots. As discussed in Section 4.2, we choose the initial parameters in the periodic strategy as

$$[I_1, L_1] = [3, 16]\text{s} \quad [I_2, L_2] = [4, 18]\text{s} \quad [I_3, L_3] = [5, 12]\text{s} \tag{13}$$

where each pair $[I_n, L_n]$ for $n = 1, 2, 3$ denotes that the human operator must interact with the robot R_n for I_n seconds within every L_n seconds. Note that the value of I_n will dynamically change according to Algorithm 1 and Algorithm 2. If the mutual trust $T_n(t)$ is too close to the lower bound, the human operator will interact less with this robot and thus the value of I_n will decrease. On the other hand, if the mutual trust $T_n(t)$ is too close to the upper bound, the value of I_n will increase.

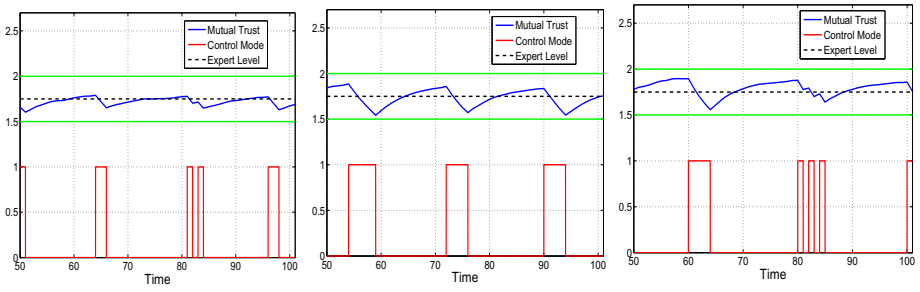
5.2 Results and Discussions

Figure 9 shows the evolution of the mutual trust $T_n(t)$ $(n = 1, 2, 3)$ for three robots within the time interval $t \in [0, 300]$ seconds, under the periodic control strategy. The green lines represent the upper bound and lower bound of the desired trust region, the black dashed lines represent the ideal expert level, the blue lines represent the mutual trust between the human operator and robots, and the red lines represent the control of the human operator. For the red line, "1" means that the human operator is interacting with the robot; and "0" means that the human operator is NOT interacting with the robot. It can be observed that the mutual trust $T_n(t)$ of each human-robot pair is consistently bounded between $[1.5, 2]$. Moreover, by comparing the mutual trust in Figure 9 with that in Figure 7, we can see that the mutual trust under the periodic control strategy in Figure 9 stays closer to the ideal expert level (black dashed line) than that under the trust-triggered strategy in Figure 7.



(a) $T_1(t)$ between human and R_1 (b) $T_2(t)$ between human and R_2 (c) $T_3(t)$ between human and R_3

Fig. 9. Mutual Trust within $[0, 300]$ seconds, under the periodic control strategy



(a) The First Robot (b) The Second Robot (c) The Third Robot

Fig. 10. Mutual Trust within $[50, 100]$

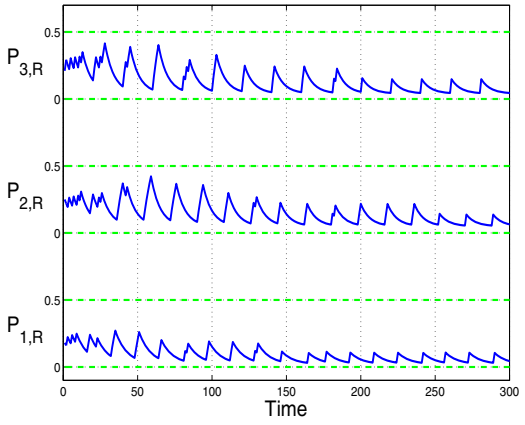


Fig. 11. Robot Performance

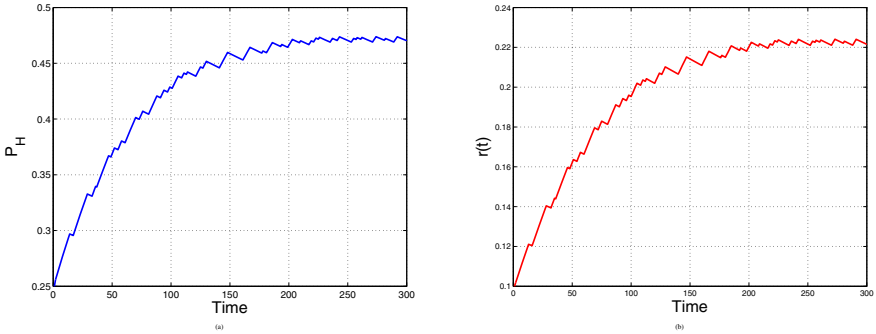


Fig. 12. (a) Human Performance, and (b) Utilization

For the sake of illustration, we further zoom in the evolution of the mutual trust within a smaller time interval $[50, 100]$ in Figure 10. We can clearly see that the human operator can only interact with one robot at a time. Moreover, for each robot, the amount of interaction time will dynamically change according to the mutual trust.

Figure 11 shows the evolution of robot performance within $[0, 300]$. As seen from Figure 11, due to the different robot performance at each time step, the teleoperated and autonomous control allocation for each robot is dynamically changing. Figure 12 shows the evolution of human performance and utilization within $[0, 300]$, respectively. Because the utilization is always less than the optimal value $\beta = 0.8$, the evolution of human performance has the same trend as the utilization. All the performance is bounded between $[0, 1]$.

6 Conclusion

In this paper, we propose a mutual trust model to capture the interactions between human and underwater robots. A set of trust based control strategies are developed to allocate the teleoperated and autonomous mode in real-time. Especially, we present a real-time scheduling algorithm for the multi-robot case. The mutual trust level depends on both robot and human performance. More specifically, we investigate the robot performance of the YSI EcoMapper AUV and establish the human performance model based on Yerkes-Dodson law. A trust-triggered control strategy is first developed for single robot and human pair. We further extend the results to multi-robot case and propose a periodic trust-based control strategy with highest-trust-first scheduling algorithm for real-time task allocation. Simulation results are presented to show the effectiveness of the proposed strategies.

Although we present a novel and attractive tool to study HRI, in particular, human-robot mutual trust, this preliminary work is a first step towards systematic study of the model and control of the dynamical evolution of HRI in collaborative human and AUV teams. Further works will focus on the extensions on both performance models and control strategies for human-robot mutual trust in underwater applications. Human performance model specific to AUV operators will be investigated. We will also consider

non-periodic trust-based scheduling for multiple human multiple robot case and the stability proof for the switched control strategies. Since the execution time of each task is dynamically changing according to the performance of the robots, we will develop algorithms to predict the robot performance within finite time horizon and also integrate the highest-trust-first scheduling algorithm into the schedulability test.

References

1. IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, France (September 2012)
2. Arzen, K.E., Cervin, A., Eker, J., Sha, L.: An introduction to control and scheduling co-design. In: Proceedings of IEEE Conference on Decision and Control, pp. 4865–4870 (2000)
3. Bainbridge, W.A., Hart, J., Kim, E.S., Scassellati, B.: Fan-out: Measuring human control of multiple robots. In: The 17th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN, pp. 701–706 (August 2008)
4. Balch, T., Arkin, R.: Behavior-based formation control for multirobot teams. IEEE Transactions on Robotics and Automation 14(6), 926–939 (1998)
5. Bertrand, J.W.M., van Ooijen, H.P.G.: Workload based order release and productivity: a missing link. Production Planning and Control 13(7), 665–678 (2002)
6. Billings, D.R., Schaefer, K.E., Chen, J.Y., Kocsis, V., Barrera, M., Cook, J., Ferrer, M., Hancock, P.A.: Human-animal trust as an analog for human-robot trust: A review of current evidence. Technical report, DTIC Document (2012)
7. Naval Studies Board. Autonomous vehicles in support of naval operations. National Research Council, Washington, DC (2005)
8. Bruemmer, D.J., Marble, J.L., Dudenhoeffer, D.D.: Mutual initiative in human-machine teams. In: Proceedings of the 2002 IEEE 7th Conference on Human Factors and Power Plants, pp. 7–22. IEEE (2002)
9. Cao, M., Stewart, A., Leonard, N.E.: Integrating human and robot decision-making dynamics with feedback: Models and convergence analysis. In: 47th IEEE Conference on Decision and Control, pp. 1127–1132 (December 2008)
10. Coeckelbergh, M.: Humans, animals, and robots: A phenomenological approach to human-robot relations. International Journal of Social Robotics 3(2), 197–204 (2011)
11. Crandall, J.W., Goodrich, M.A., Olsen Jr., D.R., Nielsen, C.W.: Validating human-robot interaction schemes in multitasking environments. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 35(4), 438–449 (2005)
12. Cummings, M.: Human supervisory control of swarming networks. In: 2nd Annual Swarming: Autonomous Intelligent Networked Systems Conference (2004)
13. Ding, X., Powers, M., Egerstedt, M., Young, S., Balch, T.: Executive decision support: Single-agent control of multiple uavs. IEEE Robotics & Automation Magazine 16(2), 73–81 (2009)
14. Freedy, A., DeVisser, E., Weltman, G., Coeyman, N.: Measurement of trust in human-robot collaboration. In: International Symposium on Collaborative Technologies and Systems, pp. 106–114 (2007)
15. Ge, S.S., Khatib, O., Cabibihan, J.-J., Simmons, R., Williams, M.-A. (eds.): ICSR 2012. LNCS (LNAI), vol. 7621. Springer, Heidelberg (2012)

16. Goodrich, M.A., Schultz, A.C. (eds.): Human-Robot Interaction: A Survey (Foundations and Trends(r) in Human-Computer Interaction). Now Publishers Inc. (2008)
17. Hancock, P.A., Billings, D.R., Schaefer, K.E., Chen, J.Y.C., de Visser, E.J., Parasuraman, R.: A meta-analysis of factors affecting trust in human-robot interaction. Human Factors: The Journal of the Human Factors and Ergonomics Society 53, 517–527 (2011)
18. Hanheide, M., Lohse, M., Zender, H.: Special issue on expectations, intentions & actions in human-robot interaction. International Journal of Social Robotics 4(2) (April 2012)
19. John, L., Moray, N.: Trust, control strategies and allocation of function in human-machine systems. Ergonomics 35(10), 1243–1270 (1992)
20. Karwowski, W., Rahimi, M. (eds.): Human-Robot Interaction. Taylor & Francis (1992)
21. Kuzuoka, H., Evers, V., Imai, M., Forlizzi, J.: In: Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction, Tokyo, Japan (March 2013)
22. Lamers, M.H., Verbeek, F.J.: HRPR 2010. Revised Selected Papers Series: LNICST, vol. 59 (June 2010)
23. Laschi, C., Breazeal, C., Nakauchi, Y.: Guest editorial special issue on human-robot interaction. IEEE Transactions on Robotics 23(5) (2007)
24. Lee, J.D., See, K.A.: Trust in automation: Designing for appropriate reliance. Human Factors 46, 50–80 (2004)
25. Lewandowsky, S., Mundy, M., Tan, G.: The dynamics of trust: Comparing humans to automation. Journal of Experimental Psychology: Applied 6(2), 104 (2000)
26. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard- real-time environment scheduling algorithms for multiprogramming. Journal of the Association for Computing Machinery 20, 46–61 (1973)
27. Majji, M., Rai, R.: Autonomous task assignment of multiple operators for human robot interaction. In: American Control Conference, pp. 6454–6459 (June 2013)
28. Mendl, M.: Performing under pressure: stress and cognitive function. Applied Animal Behaviour Science 65(3), 221–244 (1999)
29. Moray, N., Inagaki, T., Itoh, M.: Adaptive automation, trust, and self-confidence in fault management of time-critical tasks. Journal of Experimental Psychology: Applied 6(1), 44 (2000)
30. Murphy, R.R., Steimle, E., Hall, M., Lindemuth, M., Trejo, D., Hurlebaus, S., Medina-Cetina, Z., Slocum, D.: Multi-robot port-controlled hamiltonian systems under quantization. In: IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR) (2009)
31. Sha, L., Arzen, K.E., Abdelzaher, T., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., Mokd, A.K.: Real time scheduling theory: A historical perspective. Real-Time Systems 28, 101–155 (2004)
32. Sheridan, T.B.: Telerobotics, Automation, and Human Supervisory Control. The MIT Press (1992)
33. Sheridan, T.B. (ed.): Humans and automation: System design and research issues. Wiley (2002)
34. Sheridan, T.B.: Supervisory control of remote manipulators, vehicles and dynamic processes: Experiments in command and display aiding. Technical report, DTIC Document (1983)
35. Shi, Z., Zhang, F.: Predicting time-delays under real-time scheduling for linear model predictive control. In: International Conference on Computing, Networking and Communications. Workshops Cyber Physical System, pp. 205–209 (2013)
36. Wang, C., Zhang, F., Schaefer, D.: Dynamic modeling of an autonomous underwater vehicle: the ecomapper. In: Journal of Marine Science and Technology (2013) (under review)

37. Whetten, J., Goodrich, M., Guo, Y.: Beyond robot fan-out: Towards multioperator supervisory control. In: IEEE International Conference on Systems Man and Cybernetics (SMC), pp. 2008–2015 (2010)
38. Yerkes, R.M., Dodson, J.D.: The relation of strength of stimulus to rapidity of habit-formation. Journal of Comparative Neurology and Psychology 18(5), 459–482 (2004)
39. Zhang, F., Shi, Z., Mukhopadhyay, S.: Robustness analysis for battery supported cyber-physical systems. ACM Transactions in Embedded Computing Systems 12(3), Article 6 (27 pages) (2013)

Part III
System Engineering

Integrating Wireless Sensor Nodes in the Robot Operating System

Philipp M. Scholl[1], Martina Brachmann[2], Silvia Santini[2], and Kristof Van Laerhoven[1]

[1] ESS, Technische Universität Darmstadt
{scholl,kristof}@ess.tu-darmstadt.de
[2] WSN Lab, Technische Universität Darmstadt
{martina.brachmann,santinis}@wsn.tu-darmstadt.de

Abstract. The Robot Operating System (ROS) is a popular middleware that eases the design, implementation, and maintenance of robot systems. In particular, ROS enables the integration of a large number of heterogeneous devices in a single system. To allow these devices to communicate and cooperate, ROS requires device-specific interfaces to be available. This restricts the number of devices that can effectively be integrated in a ROS-based system. In this work we present the design, implementation, and evaluation of a ROS middleware client that allows to integrate constrained devices like wireless sensor nodes in a ROS-based system. Wireless sensor nodes embedded in the environment in which a robot system is operating can indeed help robots in navigating and interacting with the environment. The client has been implemented for devices running the Contiki operating system but its design can be readily extended to other systems like, e.g., TinyOS. Our evaluation shows that: in-buffer processing of ROS messages without relying on dynamic memory allocation is possible; message contents can be accessed conveniently using well-known concepts of the C language (structs) with negligible processing overhead with respect to a C++-based client; and that ROS' message-passing abstraction facilitates the integration of devices running event-based systems like Contiki.

1 Introduction

Both wireless sensor networks (WSN) and robot systems can support a plethora of different application scenarios. In disaster recovery and remote monitoring scenarios, for instance, robots can be used as reliable instruments to access and explore dangerous environments [2]. In particular, robots can perform critical interventions that would otherwise put human lives at risk, for example during the incident at the Fukushima nuclear plant [12]. Wireless sensor networks can also be used in emergency response scenarios [10]. Deployed by air over a disaster area, sensor nodes can provide fine-grained sensory data to support first responder teams [8]. While robots can perform actions and move autonomously, sensor networks are typically static but can provide continuous, detailed observations about the environment around them. In many scenarios, the combined use of both robots and WSNs is desirable. This has led to several approaches enabling communication between sensor nodes of a WSN and mobile robots active within

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014,*
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_7,

the same environment. For example, Suzuki et al. propose to use mobile robots to drop sensor nodes in critical areas where sensor measurements are needed [16]. The deployed sensor nodes form an ad-hoc wireless communication infrastructure that can in turn be used by the robots. In this paper, we investigate how to enable a seamless integration of wireless sensor nodes and mobile robots within the same system. In particular, we focus on robot systems developed using the Robot Operating System (ROS) [11,14] and wireless sensor nodes running Contiki [4], one of the most used programming environment for WSNs. Our approach can however be easily ported to other WSN platforms.

ROS is a software framework that supports the development of robot systems [11,14]. Robots usually consist of different components, including sensors (e.g., cameras, laser scanners, etc.) and actuators (e.g., servo motors). To work properly as a single unit, each of these components requires driver and other software written for its specific operating system or embedded programming environment. To implement the behavior of the robot in a flexible way, these components need to be loosely coupled with each other through communication channels. In other words, a *middleware* that allows to abstract from the peculiarity of the implementation of each component is needed. ROS provides such a middleware and uses C++ language constructs to create and relay messages from and to the heterogeneous devices. On the contrary, most WSN programming environments – like the above-mentioned Contiki or TinyOS [9] – rely on C or C-like languages to make the code more efficient and thus able to run on resource-constrained WSN hardware.

In this work we describe the design, implementation, and evaluation of a ROS client that enables the seamless integration of Contiki-based sensor nodes in a ROS-based system. The challenges posed by this integration are manyfold. First, ROS messages are transmitted in a format that is not accessible using common C-structs. Therefore, ROS messages need to be converted into a C-compliant, WSN-compatible format at runtime and vice versa. Second, sensor nodes have limited memory. To convert the format of exchanged messages, these need to be stored in an efficient manner. Third, ROS only supports TCP for transporting messages. This adds a significant communication overhead that becomes critical when resource-constrained WSN nodes are involved in the message exchange. Adding UDP support allows to cope with this problem but requires an additional software component, as detailed in Section 4. Last but not least, to support a large number of sensor nodes within a WSN, IPv6 is preferably used. Since ROS only supports IPv4, a proxy that converts IPv6 to IPv4 and vice versa is needed.

To translate ROS messages to a machine-native C-struct memory layout we resort to a code generation approach. Using a simple, centralized, proxy-based approach to resource discovery we can achieve robust synchronization despite the typically unreliable transmission links in WSN. Our approach is suitable for platforms that have no C++-support – which is instead required by standard ROS clients –, for platforms where dynamic memory allocation is prohibitive or support for predictable execution speed (e.g., real-time application) is necessary. While we evaluated our approach on Contiki-based sensor nodes[1], its extension to other C-based WSN operating systems – like, e.g., TinyOS – is straightforward.

[1] A publicly available implementation of our middleware can be found at: `https://www.github.com/pscholl/contiki_rosnode`

The main contribution of this paper is to show that a translation and access to ROS messages format using common C constructs is feasible and that access to these messages is possible without inducing a large overhead. We furthermore show that a simple periodic message exchange is feasible only for small networks.

In the following, we first briefly introduce the ROS platform and discuss related work. We then present the design of our integration middleware detailing how resource discovery and message en- and decoding works. Finally, we show the results of a simulation study documenting the performance of our middleware in terms of memory footprint and communication overhead.

2 Background and Related Work

In this section we outline the main abstractions used in ROS and discuss their applicability to WSNs. We further present established WSN middleware systems and analyse their compatibility with ROS. Finally, we review already existing ROS middleware clients for embedded systems.

2.1 The Robot Operating System (ROS) Middleware

A ROS system is a loosely coupled conglomeration of *components*. Each component represents a running process within ROS and is connected to other processes through *topics* or *services*. Components provide functionalities to the robot system like navigating to a certain position or providing raw sensor data. These functionalities are advertised through a central component, called the *ROSmaster*, through the topic and service names. Accessing components is achieved by subscribing the advertised topics or services at runtime. A robot system implemented using ROS thus consists of a compound of components that exchange messages published on topics or services.

The distinction between topics and services is as follows. Topics provide a publish/subscribe mechanism that enables unreliable many-to-many, one-way message transport. When a component publishes a message on a specific topic, the message is then instantaneously transported to all components that subscribed to that topic as well as to other interested parties. Services, on the other hand, provide one-to-one remote procedure call interactions based on the topic mechanism. A service call is only completed once an answer has been received. Or put differently, it provides the means for reliable communication.

Both topics and services rely on the exchange of *messages*. Messages in ROS are defined as nested data structures containing primitive types (integers, floating points, etc.) and arrays. These messages are defined at compile time and attached to topics/services at runtime. This leads to a statically typed message exchange. Topics, services, and messages alike are identified by strings, which also allows for hierarchical ordering. Following the taxonomy of Eugster et al. [5], ROS can be called a *hierarchically-ordered*, *topic-based*, *publish/subscribe middleware system with static component deployment*.

As an example, let us consider an application that makes a robot move to a randomly chosen location and keeps track of its movements. A ROS component that allows to dynamically determine the position of the robot from sensor measurements and to control

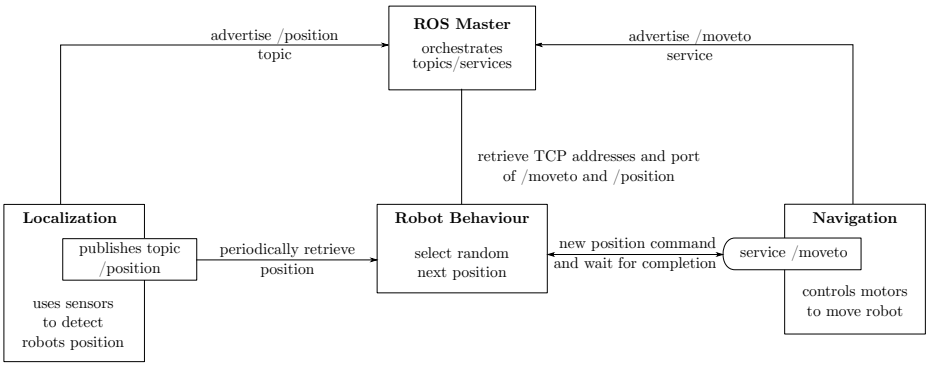


Fig. 1. The connections of a simple, exemplary ROS system. The ROSMaster resolves topics/services to TCP connections, through which the Robot Behaviour component retrieves the current robots position and sends commands to move the robot.

the robot's motors is necessary. To operate within a ROS system, this component could provide a *position* topic on which the current position of the robot is published. Also necessary to support the application is a *service* that allows to move the robot towards the target position. The application could *subscribe* to the position topic and retrieve the position periodically or whenever necessary. Figure 1 depicts this simple hypothetical system with the corresponding connections between components. A topic is used here since there is no need to acknowledge the reception of a new position. Instead, for moving the robot to a new position a service is used. This allows to let the application obtain an acknowledgment when the robot starts to move and when it has arrived at the new position. The software that provides this acknowledged and non-acknowledged communication is called a *ROS client*.

In this work, we present a ROS client that allows integrating wireless sensor nodes in a ROS system. To this end, the following functionalities must be provided: (i) A component that resolves topic/service names to network connections and establishes these connections. This can be achieved through communication with the ROSmaster. (ii) A message en-/decoder that translates the ROS message on-wire format to one that is compatible with the target system. Our implementation targets sensor nodes running the Contiki operating system and is henceforth referred to as the *Rostiki* client.

2.2 Middleware Systems for Wireless Sensor Networks

A number of middleware systems able to cope with the specific requirements of WSNs have been proposed in the literature [15,17]. These also include several publish/subscribe middleware systems like, for instance, LooCI by Hughes et. al. [6]. LooCI provides several useful services like runtime deployment of components, runtime introspection, dynamic reconfiguration, distributed resource discovery, and an event bus system. LooCI also supports multiple platforms including the Contiki operating system. Adapting LooCI to make it able to interact with ROS is in principle possible. Indeed, LooCI inspired our work on *Rostiki*. However, many of LooCI's features cannot be

mapped directly to ROS middleware concepts. Integrating LooCI and ROS is thus unnecessarily complex. In comparison, our Rostiki client is lightweight and can be used seamlessly with a ROS system.

The Global Sensor Network (GSN) is a middleware platform, developed by Aberer et al., that "*provides a scalable infrastructure for integrating heterogeneous sensor network technologies using a small set of powerful abstractions*" [1]. GSN allows to discover, integrate, and share sensing devices as well as sensor data stream in a single system. To integrate a specific device or stream, a corresponding *wrapper* must be available. However, GSN is specifically tailored to WSNs and does not provide the functionalities needed to support robot systems. Another related approach is IrisNet, a software infrastructure aimed at facilitating the development of "*sensor-enriched Internet services, which combine traditional data sources with information collected from live sensor feeds*" [13]. Although related to our work, IrisNet focuses on enabling sensor data to be easily retrieved from and made accessible to heterogenous sources. Unlike ROS, Irisnet does not provide mechanisms to allow the implementation of robot systems.

Like ROS, the TinyOS operating system for WSNs also relies on a component-based model [9]. TinyOS is written in nesC, a C-dialect that supports TinyOS's event-based programming paradigm. For communication TinyOS relies on the Active Messages abstraction [3]. To optimize code size and memory usage the components used by a TinyOS application are loaded and linked at compile time. This makes TinyOS incompatible with ROS middleware concepts, which requires re-wiring of components at runtime.

While it is not a complete middleware system, the Constrained Application Protocol (CoAP) is also related to our work. CoAP is an IETF effort to standardize web services for resource-constrained devices. It provides a component model, a mechanism for wiring components and a resource discovery component. It is based on an adaption of well established web service concepts like Representational State Transfer (REST). Since CoAP can be integrated directly into modern web browsers [7] it could have been used to map ROS topics easily. However it lacks a concept of streaming sensor data, often found in ROS systems.

2.3 ROS Clients for Embedded Devices

Besides our work there exist other approaches that focus on enabling embedded devices to be directly connected to ROS systems. These include: ROSSerial[2], μROSNode[3] and ros/C[4]. The goal of these efforts is the same as ours: enabling systems composed of resource-constrained devices to interact directly with the ROS middleware.

To the best of our knowledge, ROSSerial is the first approach that pursued this goal. It was originally designed to enable communication from Arduino devices through serial lines. Recently, it has been enhanced to support generic embedded systems through TCP/IP. Rostiki's resource discovery mechanism is compatible to the one used by

[2] Available at: `https://github.com/ros-drivers/rosserial`
[3] Available at: `https://github.com/openrobots-dev/uROSnode`
[4] Available at: `https://github.com/synapticon/rosc`

ROSSerial and certain components from both are interchangeable. However the major drawback of ROSSerial is its C++-based message de-/encoder. While this provides object-oriented access to ROS messages it also decreases the number of supportable target systems. The message decoder also relies on dynamic memory management, which is not available on many resource-constrained platforms like wireless sensor nodes.

The goal of µROSNode is to provide a lightweight ANSI-C compatible ROS middleware client. Also in this case, however, the targeted embedded systems are assumed to be more powerful than typical WSN platforms. The message de-/encoder heavily relies on dynamic memory management to process received messages and it is therefore only partially useful on WSN platforms. µROSNode's message parser relies on function calls to retrieve message members and it can thus only conditionally be defined as object-oriented.

Ros/C is the most recent effort started to enable the ROS middleware to run on embedded platforms. Nonetheless, it already implements the same resource discovery scheme as the standard ROS client. However, it does not (yet) provide an object-oriented interface for accessing ROS messages. Instead, it rather relies on a push-based approach, i.e., it triggers a callback for each member in a ROS message. This puts a considerable burden on the application programmer, who is left in charge of parsing the structural information of a message. This also creates a large maintenance effort when message definitions change. However, it allows to handle messages that do not fit into the main memory of the target system.

Unlike these related approaches, Rostiki allows for in-buffer processing of received messages while still providing an object-oriented interface to access message content. Furthermore, Rostiki is platform-agnostic and can also be easily modified to support different resource discovery schemes.

3 System Design

The main requirements driving the design of Rostiki are: (i) the need to avoid unnecessary runtime and API complexity while maintaining a low memory/code footprint, (ii) portability, and (iii) robustness against transmission failures. These requirements stem directly from the characteristics of wireless sensor network platforms. In particular, microcontrollers used on WSN hardware platforms are typically endowed with only few kilobytes of internal memory. Furthermore, they do not support dynamic memory allocation. To cope with these challenges we present, in Section 3.1, the design of an "in-buffer" en-/decoding scheme for ROS messages. The code generator, which creates the message de-/serializer component, is further described in Section 3.2. Another important factor driving the design of Rostiki is the unreliable nature of wireless communication, which not only influences the actual data transmission but also the overhead of maintaining a comprehensive view of the available resources in the network. In Section 3.3, we describe two strategies that make Rostiki able to cope with this challenge.

Figure 2 shows the design of Rostiki. The message en-/decoder is included in the Rostiki client and runs directly on the sensor nodes. A border router forwards IPv6 message to the ROS network, which relies on a WSN proxy to gather messages from the border router. This proxy takes care of publishing advertised topics to the ROSmaster. The ROSmaster is the central orchestration unit. It keeps track of published topics

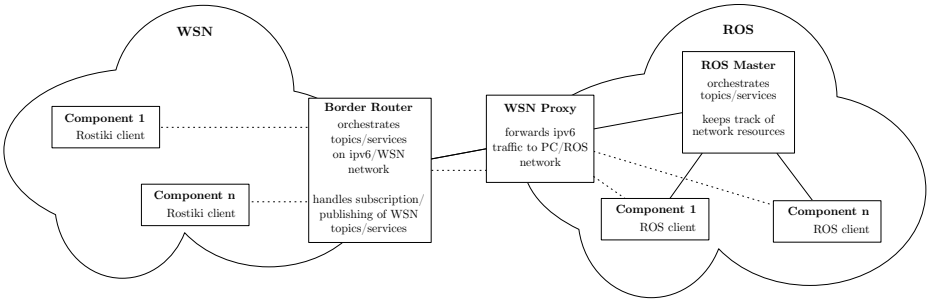


Fig. 2. The design of the Rostiki middleware system. A ROS-based system (right) and a WSN (left) are shown. Dashed lines designate established data connection from WSN components to ROS components. Solid lines represent management information flows, like advertised topics and services. The WSN proxy handles the management of this information for the WSN and proxies subscriptions and advertisement to the ROSmaster.

and services and it can be queried for the network address of these services. This can be used by the ROS and Rostiki client to connect to other topics and services. After these network addresses have been resolved a direct connection to the queried topics or services are established. If a direct connection between a ROS component and WSN component is not possible (this happens for instance when the ROS network runs on IPv4), the WSN proxy will forward those messages.

3.1 En-/Decoding ROS Messages in C

A central part of our Rostiki client is the encoding and decoding component of ROS messages. Since ROS messages are defined at compile time, we adopt a code generation approach to map the on-wire ROS message format to the machine-dependent C-struct memory layout. The generated code should minimize the amount of additional memory that is needed to hold the message and the memory required to run the en- and decoding processes. To this end, especially due to the absence of dynamic memory allocation, most operations are done inside the transmission buffer with only a small fraction done via temporarily allocated memory on the stack. This approach allows to adapt the endianess of primitive types as well as to align the message to the machine's natural data access width by reordering the elements inside the message.

The ROS on-wire message encoding works by sequencing the elements of a message one after another. Primitive types (like integers, floats, etc.), arrays of fixed size, nested messages, arrays with a dynamic size as well as strings are supported. Dynamic arrays and strings are preceded by their length, i.e. they follow the Pascal string encoding convention. Figure 3 shows an example of an encoded message. The left hand side shows the message declaration of two nested messages. The shaded areas in show the message as it would be encoded in a transmission buffer. It can be seen that the elements are concatenated in the buffer while the structural information has been stripped.

The straightforward ROS message encoding is not directly mappable to a C-struct memory layout, because there is no language construct to allocate memory for dynamic

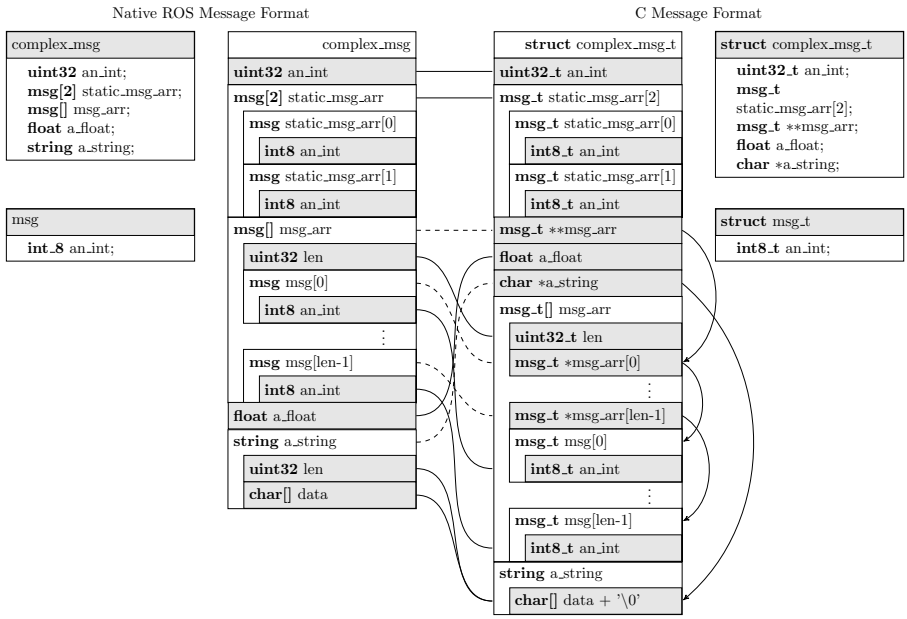


Fig. 3. The memory layout (and declaration) of an on-wire ROS message and the corresponding C-struct layout (and declaration). Shaded areas depict actual used memory, while non-shaded areas depict structural information that is not stored or transported. Lines in this figure represent the operation during en-/decoding, solid lines are copy operations while dashed line show when management information needs to be added (and accordingly further memory needs to be allocated) or can be removed.

sized arrays (or strings) within structs. However, at the price of a slight increase in storage space the memory can be reorganized to allow for natural access (i.e. using available C language concepts, like pointers). To this end, we conceptually split the message into a *static* and *dynamic* part. As its name implies the static part holds all elements for which the size is known at compile time. These include primitive types and static arrays as well as indirections to the dynamic sized elements. The data corresponding to these static parts is copied to the head of the memory buffer, while the data corresponding to dynamic elements is moved to the dynamic part (i.e. to the tail) of the memory buffer and can only be accessed through the aforementioned indirections. This approach allows to define a ROS message in the most natural C-language struct, as depicted in Figure 3. The right-hand side of this picture shows the respective C-struct declaration and its memory layout. The figure also shows where necessary indirections are created. Thus, all primitive types and static arrays are allocated at the head of the message buffer, dynamic sized elements will be replaced by indirections and their data moved to the tail of the message buffer. It should be noted that this approach cannot be easily applied to nested messages, since these again can contain dynamic sized elements. The code generator (see Section 3.2) copes with this problem by "unrolling" nested message definitions. For each nested message, there is a special part of code that en-/decodes a particular nested message inside the containing message. By reorganizing

the static elements to the head of the message buffer, moving dynamic elements to the tail of the buffer and adding according indirections it is possible to map ROS messages to the CPU-specific C-struct memory layout.

In order to decode a ROS message to a C-struct we propose the following two-step algorithm. The algorithm allocates memory on the stack only temporarily and places the result in the original transmission buffer.

1. Allocate memory on the stack for all static elements and copy all static element into this memory region. While copying, alignment and endianess of primitive types are fixed. Additionally, indirections to dynamic arrays and strings are initialized since information on their size is now available.
2. Iterate over all message elements in reverse order. The data of each dynamic element is then *moved* to its respective position at the tail of the message buffer. Once this movement has been completed, the static elements are copied from the stack into the transmission buffer and the decoded message is returned.

In step 1 the mentioned storage space overhead is introduced. With respect to standards ROS encoding, the C-struct encoding needs additional management information in order to support dynamic elements. For the dynamic arrays an additional indirection has to be added, i.e. the number of bytes needed for one pointer on the target system. For strings the overhead can be calculated with the following formula: $sizeof(char*) - sizeof(uint32\_t) + 1$, which represents the size of the indirection needed to access the string and the space reduction by converting from a length-prefixed string to a null-terminated C-string.

Encoding messages from a C-struct representation to the ROS message representation takes less effort. The encoding process indeed simply requires iterating over all elements in the message and copying their data, again fixing endianess and alignment issues, and prefixing by length for dynamic elements. This of course only works when transmission buffer and message buffer do not overlap. However, we have shown that a little overhead in terms of memory consumption allows to access the static and dynamic elements of ROS message with C constructs.

3.2 Code Generator

As mentioned in section 3.1 the message en-/decoder components are generated during compile time from the standard ROS message description files. After this step has been performed, the generated message en-/decoder C code has to be compiled into the firmware to allow for message en-/decoding. This code generation step is one of the core contributions of our work and can be divided into three steps: (i) building a syntax tree by parsing ROS message definitions; (ii) generating the struct definitions; and (iii) generating the message en-/decoder code.

Parsing the ROS message definition into a syntax tree is the simplest step during code generation. Message definitions like the exemplary one in Figure 3 are read line-by-line, and type as well as name definitions are stored. This type can either be a primitive type, string, static or dynamic list, or another message definition which is recursively resolved. This step result in a topologically sorted tree. A path in this tree then contains the information on how the ROS message will be encoded on the wire and its structure.

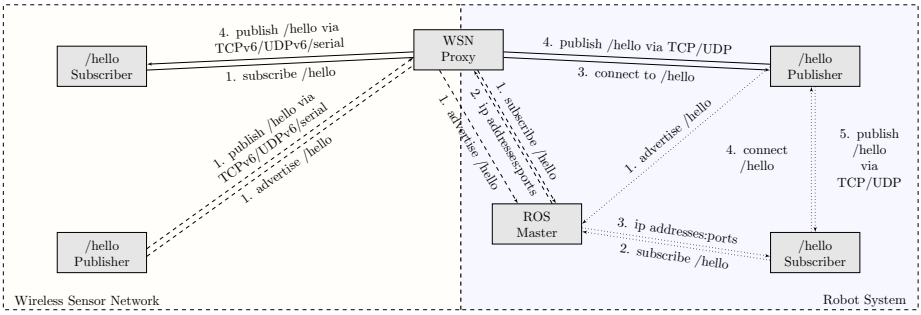


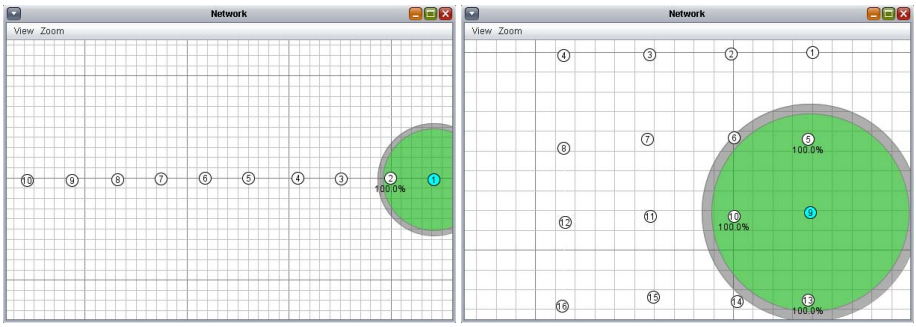
Fig. 4. The figure shows the resource discovery and connection establishment of ROS components. Each block represents a component running either on the robot (right side) or on a wireless sensor node (left side). The dotted lines constitute the establishment of a connection between two PC components on the /hello topic. Dashed lines show the communication of the WSN components with the ROSMaster via the WSN proxy. And solid lines represent the connection between a WSN component and a PC component (in this scenario the WSN component connects to a PC component).

The next step consist of creating the struct definition from the syntax tree. To this end, we employed the visitor design pattern, i.e. each node in the tree is visited in a breadth-first manner and the stored type of this message member is translated to its C counterpart. Primitive types as well as static lists are directly translated. References to other messages are resolved to the just created "struct"-counterpart. Dynamic list and strings are replaced by a pointer of their respective type. This results in a struct definition like the one depicted on the right-hand side of Figure 3.

The last step consist in generating the code for decoding or rather translating the ROS on-wire message to a C-struct compatible memory layout, as already outlined in Section 3.1. To perform this step the message definition tree is traversed twice in depth-first manner. In the first phase only members of static size are considered and code is generated to copy those temporarily on the stack. In the second phase *memmove*-calls, which move dynamic sized members to the tail of the transmission buffer are generated. Afterwards code is generated that copies static members back into the transmission buffer and that fixes the pointers to the dynamic message parts. This way the transmitted messages are accessible through the previously defined C-structs.

3.3 Resource Discovery

In this work, the expression *resource discovery* is used to address mechanisms that allow string descriptions to be resolved to actual data sources, like network connections. In the case of our proposed ROS client, strings can be used to address *topics* as well as *services*. The central ROSmaster keeps track of the networking addresses of individual nodes, of the respective *topics* they publish on or subscribe to and their advertised *services*. In our implementation the WSN proxy performs this task for the WSN. This centralized proxy is used to keep transport-agnostic information about the advertisement/subscription of the connected WSN nodes. For example, whenever a WSN node



(a) Linear layout with 2-10 nodes. (b) Rectangular layout with 4,9,16 nodes.

Fig. 5. Chosen multi-hop network topologies for simulative evaluation

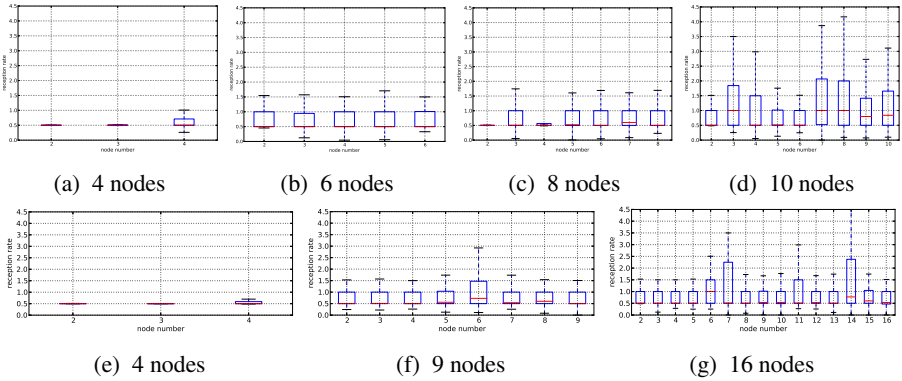
wants to subscribe to a specific *topic* it sends a request to the WSN proxy, which then negotiates this request with the ROSmaster, creates a connection to the remote node and brokers the data exchanged on this topic. Figure 4 shows the message exchange of the WSN proxy, ROSmaster and participating nodes during topic negotiation (i.e. subscribing to/publishing on a *topic*) inside the ROS network, and over the WSN boundary.

While a centralized approach to resource discovery, like the one presented here, constitutes a single point of failure it has several advantages. First of all, multiple transport protocols like TCP/IPv6, UDP/IPv6, serial transmission lines or the XBee protocol can be supported. While for most of the commonly used protocols a direct connection between the nodes of the ROS network and WSN network is not possible (e.g., one node communicates via the XBee protocol), it is still possible to directly connect those that do. For example, if both a WSN node and a ROS node communicate on the same IP link a direct connection can be established instead of brokering all message through the proxy, mitigating the single point of failure. The same applies to connections inside the WSN network as well. In this case, the WSN proxy only provides a central register of resources and their respective network addresses.

Furthermore, assuming an unreliable transport channel allows to cover a large number of different transport channels without increasing implementation complexity – resource discovery needs to be as reliable as possible. To keep this complexity on a manageable level we adopt a stubborn communication strategy, i.e. nodes communicate their advertised/subscribed topics and services to the WSN proxy in a periodic fashion. This strategy does not scale well and is not resource-efficient, but can be customized to the specific environment by adapting its period. Also, such a periodic exchange of resource information provides a robust discovery mechanism for unreliable channels.

4 Implementation and Simulation

Our implementation is based on the Contiki operating system [4]. Besides allowing to program WSN platforms, Contiki also provides the means to communicate with other devices through its TCP/IPv6/6LoWPAN stack. Also, the event-based nature of Contiki harmonizes well with the ROS publish/subscribe middleware concept – messages



(a) 4 nodes (b) 6 nodes (c) 8 nodes (d) 10 nodes

(e) 4 nodes (f) 9 nodes (g) 16 nodes

Fig. 6. Reception rates for the linear (upper) and rectangular (bottom) network topology

published on a topic are delivered to Contiki processes as events and are also published to topics in such manner. It should be noted that while our implementation is based on Contiki, it is also general enough to be ported to other platforms like TinyOS [9] or even to be uses as an alternative ROS client on personal computers. Besides an event delivery mechanism, for the reception of messages, only a timer and the message transport needs to be implemented.

In this section we compare our implementation to the ROSSerial implementation, which also targets embedded platforms. We quantify the message translation and memory overhead of both solutions, as well the achievable throughput in presence of unreliable transmission links. Our WSN proxy (called hector_serialization[5]) implementation is compatible to the ROSSerial one and can be used in place, but supports more transport protocols.

4.1 Communication Overhead

To validate our resource discovery approach and the chosen UDP message transport, we simulate multiple sensor network topologies and measure the achievable message throughput. Since our implementation is based on Contiki, we use the RPL/ROLL[6] multi-hop routing protocol and Cooja for simulating Sky/TelosB sensor nodes running our proposed Rostiki client.

For evaluating the communication performance of the proposed middleware, we chose to use a subscription scenario. Besides the subscription delay, this scenario also shows the probable performance of publishing on a ROS topic from the sensor nodes. In the subscription scenario, all participating sensor nodes are subscribed to the same ROS topic, where messages are published with a fixed rate of 2Hz. This rather slow rate has been chosen to avoid overloading the network, which can easily happen due to the nature of message transport. Each node is connected to the published ROS topic

[5] Available at: `https://github.com/tu-darmstadt-ros-pkg/hector_serialization`

[6] Using the default RPL configuration of Contiki.

Table 1. This table shows the code size in bytes, as reported by the compiler-generated memory map file, of the ROSSerial and our implementation on various WSN platforms

	C++ ROSSerial			C our solution			
	discovery	de-/encoder	total	discovery	de-/encoder	total	compiler
TMote Sky	996	2502	3498	806	3774	4580	msp430-gcc 4.6.3
Zolertia Z1	1356	2332	3688	806	3814	4620	msp430-gcc 4.6.3
RedBee EconoTag		-		936	3052	3988	arm-gcc 4.3.2
Jennic-based jNode		-		788	3583	4371	ba2-gcc 4.1.2
Intel Core i7	1205	2015	3220	1074	2222	3296	gcc 4.7.2

via an unicast channel, i.e. for each published message there will be at least n network messages, where n is the number of subscribed nodes not counting the increase in traffic due to the multi-hop forwarding.

We deliberately chose a linear and a rectangular network topology, as shown in Figure 5. As can be seen, each node in the linear layout (Figure 5a) has two neighbours, respectively two to four in the rectangular layout (Figure 5b). This varies the *message forwarding load* for each node. Furthermore the *number of hops* or the path length to reach each node is varied by these layouts via the number of participating nodes. We measured the arrival time of each published message to compare the *rate of message arrivals* to our fixed rate of 2Hz. Each scenario repeated five times for different numbers of participating nodes. In these scenarios node number 1 is the special border router, which relays messages from the WSN to the robot.

The results of these experiments are depicted in Figure 6 for the linear layout and rectangular layout. The baseline of those experiments are visible in the scenarios with a low number of participating nodes (see Figure 6a and 6e). There, the median delay is exactly at 0.5s with a low standard deviation, which is what we expect by a message delivery rate of 2Hz. However, it is also visible that as the number of nodes increases the message delivery rate increases. Surprisingly, this is not correlated to the node's path length or the total number of nodes. While the steady increasing delay in Figure 6a-Figure 6d suggests that the total number of nodes is the limiting factor, this does not show for the maximum number of nodes in Figure 6d, where the total delay is not evenly distributed anymore. Similarly Figure 6e-Figure 6g show that increasing the number of nodes does not evenly distribute the message delivery delay, but suggests that the forwarding load of each node, i.e., the number of neighbours, has a much higher influence on the communication performance. This hypothesis is confirmed by the fact that node number 6, 7, 11 and 10 are among the ones with the highest message delay in the rectangular layout, which in turn also have the highest neighbour count. We therefore assume that our proposed unicast communication model works only for small scale networks and that it is mandatory to move to multicast communication support for networks of larger scale.

4.2 Memory Overhead

We evaluate two different types of memory overhead. The first one originates from the necessary code to handle resource discovery and to de-/encode the ROS message

format. Table 1 shows the size of our proposed solution split into a discovery and a de-/encoder part compared to the C++-based ROSSerial solution. The specific messages for which these de-/encoders were generated for are shown in Figure 3. We compiled our solution for multiple typical WSN platforms that are supported by Contiki, like the TMote Sky, Zolertia Z1, RedBee EconoTag, the Jennic-based jNode and for a native environment based on an Intel Core i7. It can be seen that our C-based solution consumes more code memory for the de-/encoder when compared to the C++-based ROSSerial solution. This increase in memory consumption is mainly due to the fact that our solution supports arrays of nested messages, which needs code to unroll this nesting. This feature is not supported by the ROSSerial implementation and leads to a lower code memory consumption. For messages which are not using these nested arrays similar space is required for both solutions. It can also be seen that discovery has comparable code complexity.

The second overhead we can look at is the memory required to store the message. For encoding messages, this overhead is nearly the same. In both cases the message needs to be stored either on the stack or in static buffers. Additional memory, i.e. information that is not transferred, is needed for accessing strings and dynamic arrays via indirections. This additional memory is also needed when storing a decoded message. Our solution however decodes in the transmission buffer, which allows to save the memory for almost the whole message. If we let n_s be the number of strings, n_a the number of dynamic arrays, m be the number of elements in an array with nested messages, and $sizeof(x)$ be the bytes required to store an element without string and array data, we can calculate the memory required for storing a decoded message. For ROSSerial this equals $sizeof(msg) + n_s * sizeof(char*) + n_a * sizeof(void*)$, i.e. the whole message plus additional space for storing indirections to strings and arrays. For our solution the additional required space is only $n_s * sizeof(char*) + (n_a + m) * sizeof(void*)$. Thus, our solution requires less additional storage space, while also supporting arrays of nested messages.

4.3 Message Translation Overhead

We define message translation overhead as the time needed to translate a message from the ROS on-wire format to the memory layout used by the programmer's code. In our implementation this overhead stems mainly from the adjustment of the memory layout of ROS messages to the architecture-dependent C-struct layout. Since there is no concept of in-place strings or dynamic arrays in C, these needs to be shifted to the tail of the message buffer in order to be accessible from the message header. Primitive types like floats and integers have to be copied in order to convert them to their machine-native format.

To quantify this overhead we measure the CPU time difference required to encode/decode the two messages shown in Figure 3 with the C++-ROSSerial implementation and our proposed C-based implementation. We ran these experiments on a Linux machine with a quad-core Intel Core i7 CPU running at 2.8GHz. Each measurement was repeated $10,000$ times and, since ROSSerial does not support nested dynamic arrays, we varied the length of the included array of strings as the worst case scenario for both implementations. Figure 7 shows the result of this experiment. While the encoder

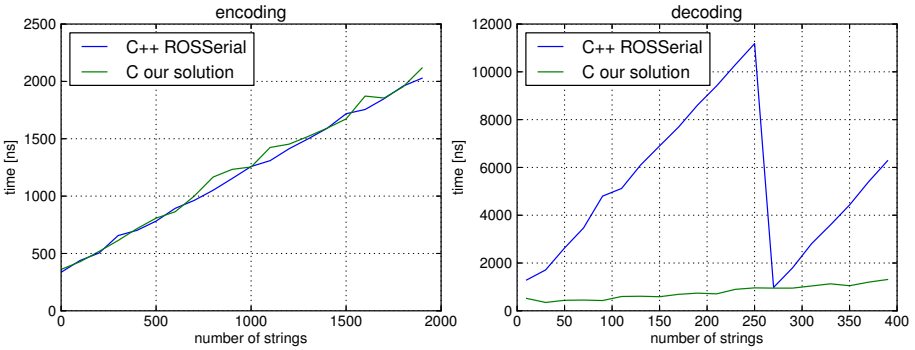


Fig. 7. Time needed for the two en-/decoder implementation to encode/decode a "worst-case" message on a PC. It's clearly visible that the use of dynamic memory management (via realloc) is a performance problem during decoding.

performance is virtually the same, the ROSSerial decoder shows almost a 10-fold increase in runtime compared to our C-based solution. This mainly stems from the use of dynamic memory management (via realloc) in this decoder to support dynamic arrays. In our solution this is solved by reordering the memory layout and moving dynamic arrays to the tail of the buffer. One can also see, when the memory needs to be moved around by the operating system during the *realloc* operation and when enough memory is still available (the sharp decline in time consumption during decoding for ROSSerial is an artifact of this). Our solution can therefore decode messages faster while not relying on the availability of a dynamic memory management unit.

5 Conclusions and Future Work

Connecting wireless sensor networks to the Robot Operating System is currently hampered by the lack of ROS-compatible components able to support resource-constrained devices like sensor nodes. In this paper we have described one of the first approaches enabling the integration of WSNs and ROS. We have shown that it is possible to efficiently translate ROS message (encoded by sequencing all message elements) into Contiki messages and vice versa. Furthermore, we have demonstrated that a simple resource discovery approach can achieve robust functionality. One drawback of our proposed message decoder is that transmitted messages must be stored entirely into the memory of the microcontroller. Further, we have left open to verify whether and how our approach scales to large networks. Future work should look into different resource discovery schemes. For example, the topic-based publish/subscribe concept could be mapped to multicast approaches. In particular, we believe that multicast DNS for topic resolving and multicast communication for message exchange could prove to be a viable alternative to a centralized approach. Furthermore, a message encoding that explicitly separates static and dynamic parts of a message might also allow for further optimizations.

Acknowledgments. The authors would like to thank the anonymous reviewer for their constructive comments on early versions of this paper. This work has been partially supported by the LOEWE research initiative of the state of Hesse, Germany, through the Priority Program Cocoon and by the German Research Foundation (DFG) through the Collaborative Research Center MAKI (SFB 1053) and the Graduate School on Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments (GRK1362).

References

1. Aberer, K., Hauswirth, M., Salehi, A.: Global sensor networks. Technical Report LSIR-REPORT-2006-001, EPFL (2006)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)
3. Buonadonna, P., Hill, J., Culler, D.: Active message communication for tiny networked sensors (2001)
4. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004), pp. 455–462. IEEE Computer Society, Washington, DC (2004)
5. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The Many Faces of Publish/Subscribe. ACM Computing Surveys 35(2), 114–131 (2003)
6. Hughes, D., Thoelen, K., Horré, W., Matthys, N., Cid, J.D., Michiels, S., Huygens, C., Joosen, W.: LooCI: A Loosely-coupled Component Infrastructure for Networked Embedded Systems. In: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2009), pp. 195–203. ACM, New York (2009)
7. Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power CoAP for Contiki. In: Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), Valencia, Spain. IEEE (October 2011)
8. Kumar, V., Rus, D., Singh, S.: Robot and Sensor Networks for First Responders. IEEE Pervasive Computing 3(4), 24–33 (2004)
9. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., Culler, D.: TinyOS: An Operating System for Sensor Networks. In: Weber, W., Rabaey, J., Aarts, E. (eds.) Ambient Intelligence, ch. 7, pp. 115–148. Springer, Heidelberg (2005)
10. Lorincz, K., Malan, D.J., Fulford-Jones, T.R., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M., Moulton, S.: Sensor networks for emergency response: challenges and opportunities. IEEE Pervasive Computing 3(4), 16–23 (2004)
11. Martinez, A., Fernández, E.: Learning ROS for Robotics Programming. Packt Publishing (Septeber 2013)
12. Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., Kawatsuma, S.: Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. Journal of Field Robotics 30(1), 44–63 (2013)
13. Nath, S., Ke, Y., Gibbons, P.B., Karp, B., Seshan, S.: IrisNet: An Architecture for Enabling Sensor-Enriched Internet Services. Technical Report IRP-TR-02-10, Intel Research Pittsburgh (December 2002)
14. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T.B., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (May 2009)

15. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. ACM SIGMOBILE Mobile Computing and Communication Review (MC2R) 6(4), 59–61 (2002)
16. Suzuki, T., Kawabata, K., Hada, Y., Tobe, Y.: Deployment of Wireless Sensor Network using Mobile Robots to Construct an Intelligent Environment in a Multi-Robot Sensor Network (July 2008)
17. Yu, Y., Krishnamachari, B., Prasanna, V.K.: Issues in designing middleware for wireless sensor networks. IEEE Network Magazine (2003)

Mobile Sensing Platforms for Implementing Mobile Sensor Networks

Kemal Akkaya, Izzet Senturk, and Shadi Janansefat

Department of Computer Science
Southern Illinois University, Carbondale, Illinois, USA
{kemal,isenturk}@cs.siu.edu, shadi.janan@siu.edu

Abstract. Wireless Sensor Networks (WSNs) have received significant attention from the wireless networking research community within the last decade due to their potential to be deployed in many real-life applications where human accessibility is limited. While early deployments of WSNs utilized stationary tiny sensors, later, due to wide range of application possibilities, deployment of mobile sensors within WSNs were considered. These sensors have been used for both sensing and mobility purposes which led to a new type of WSNs called Mobile Sensor Networks (MSNs). In this chapter, we examine various mobile sensors from the perspective of their design, implementation and how they form a network. In particular, we investigate the characteristics of existing MSN testbeds and categorize them based on these characteristics. We also provide future research issues in regards to MSN testbeds and their realization in real-life applications.

1 Introduction

Different from wireless sensor network (WSNs), MSNs employ special mobile sensing platforms and form a network of mobile nodes [15]. Thus, the nodes can both do sensing and move. These nodes are typically formed either from robots or from scratch. In this chapter, our focus will be on these special mobile sensing platforms and the MSNs formed with the networking of these nodes.

Specifically, we first look at the design and implementation of mobile sensing platforms. We evaluate the sensing platforms developed at different universities or available on the market using several metrics such as cost,functionality, availability and interoperability. Different mobile sensing platforms have been proposed in recent years to enable experimental research in MSNs. Our main goal in evaluating these mobile sensing platforms is to provide options to the application designers in terms of selecting the right mobile nodes for their needs. The secondary goal is to promote more research by pointing out their deficiencies. The design challenges in terms of hardware will also be discussed in our chapter.

After the evaluation of mobile sensing platforms, we turn our attention to the MSNs that are formed using these platforms. These are referred to as MSN

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014,*
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_8,

testbeds. The study of MSN testbeds is crucial in understanding the actual performance of the existing protocols designed for MSNs. As well known in the research community, majority of the studies on MSNs were simulation-based (or emulation-based at best) which did not take the issues in real-life contexts into consideration. These issues may include the evaluation of the node/network energy consumption characteristics, consideration of nodes' restrictions in moving in the region and evaluation of their performance in a testbed in terms of node movement and data collection.

As a result, there has always been critics in the research community that studies involving MSNs may not be accurate in terms of the conclusions they present. To address these concerns, recently some initiatives have been taken to facilitate the development and accessibility of MSN testbeds. In this chapter, we will also survey these existing MSN testbeds in terms of their research goals, functionalities, accessibility and applications where they can be employed. Finally, we will conclude the chapter by providing some future issues in the design, development and evaluation of MSN testbeds. We believe that these issues will be beneficial for the newcomers into this research area.

The main contributions of this chapter are as follows: 1) We introduce the existing mobile sensing platforms along with their characteristics to be used for MSNs; 2) We present the existing MSN testbeds using these platforms and where they are deployed; 3) We compare the advantages and disadvantages of the mobile sensing platforms and testbeds so that the application designers will be able to pick the right platform for their needs.

This chapter is organized as follows. Section II is dedicated to mobile sensors. In Section III, we describe the mobile sensing testbeds along with their categorization. Finally, in Chapter IV, we conclude the chapter along with some future research issues.

2 Mobile Sensing Platforms

While there has been a number of designed mobile sensors in the past, there was no standardization and no comparison among these designs. Most of the designs are motivated based on the applications' needs, research goals and availability of resources. Because of this wide variety, the impact of MSN research stayed minimal since there were very few applications which could adapt the deployment of these mobile sensors. In this chapter, by listing and categorizing these mobile sensing platforms, we aim to facilitate the use of MSNs which will lead to more research and more applications.

2.1 Metrics for Evaluation

Before we start describing the mobile sensing platforms, we need to come up with some evaluation metrics to compare and contrast the existing designs. Below is a non-exhaustive list of metrics for our evaluation:

1. **Cost**: Cost is one of the major design considerations which was addressed in the previous works. In particular, the tendency of mass deployment in WSNs compels to minimize the cost for each single unit. This will be shown as High and Low in the table.
2. **Functionality**: This is another important aspect which needs to be visited while designing the robot. Adaptability of different experimental settings, computational power and camera properties are part of this functionality. This will be shown as High, Medium and Low in the table.
3. **Navigation**: This feature is about the ability to avoid obstacles or collisions and thus crucial for the success of the application-level goals.
4. **Communication**: This is the ability to communicate with other devices in the network via standard protocols. For instance, being able to communicate with COTS sensor motes and utilize the same communication protocols are crucial for convenient development.
5. **Availability**: Finally, an important factor is the availability of the robot or its parts commercially to facilitate large-scale testbed designs.
6. **Localization**: This indicates the ability of the node to determine its location and direction to move.
7. **Size**: The size of the mobile platform is also an important parameter that will affect the testbed design as well as the navigation.
8. **Application**: The application for which the mobile sensing platform is developed also affects the design in the sense of navigation, energy consumption, communication characteristics, sensing and cost.

Next, we describe the existing platforms by considering the above metrics.

2.2 Ragobot

Ragobot [10] is a platform that was developed for research purposes at UCLA. The design is based on providing mobility for a mote as shown in Fig. 1. Ragobot employs a design paradigm that balances modularity and efficiency. The mobile platform is very small (60.0mm wide and 133.5mm long) compared to other mobile platforms that will be described shortly. This provides the opportunity to use them in large-scale settings. The communication will be through the Mote's radio component (900MHz) although this was not mentioned in the paper [10]. Despite its small size, Ragobot is heavily instrumented with video capture, audio capture, processing, and playback, IR collision avoidance, IR cliff detection, RFID read/write, inertial navigation and more. Navigation and the coordination of

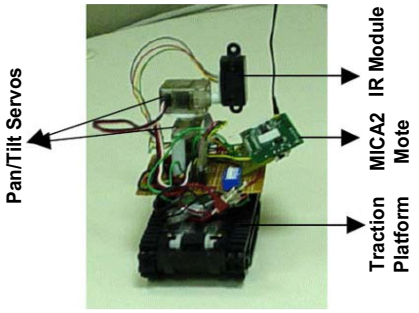


Fig. 1. Ragobot, a mobile sensor developed at UCLA. Taken from [10].

Ragobot are ensured through leaving "hint" on RFID tags and infrared sensors are exploited for obstacle avoidance. While these capabilities were tested with two Ragobots, no testbeds were created. The proposed design also enables adding additional sensors or equipment if demanded by the application; namely light sensor, humidity and temperature sensor, GPS, etc.

The main drawbacks with this robot are its cost and commercial availability. The nodes should be developed in-house and requires special hardware which may not be available. In addition, the small hardware makes it vulnerable to damages and may reduce the ability to navigate in rough environments. Finally, the speed of the node will also be less compared to other COTS robots.

2.3 Robomote

Robomote [23] is another mobile robot platform addressing the problems in large-scale distributed robotics and WSNs. This robot is designed in an attempt to find a trade-off between three main design concerns: size, cost and functionality. The Robomote is a single printed circuit board with dimensions 3.81cm x 2.23cm based on an Atmel AT9OS8535L bit micro controller as seen in Fig. 2 . This board and the robot component connect to a mote which is named Robomote.

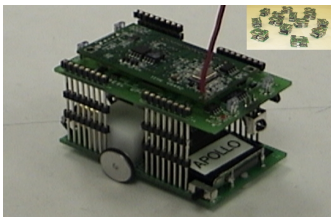


Fig. 2. Robomote: a mobile robot platform for large-scale WSNs. Taken from [23].

The Robomote offers the radio communications interface and controls the robot platform via RS232 serial commands. Robomote is equipped with a wireless network interface (i.e., Mote2 radio) for communication and accurate odometry and compass (2-axis Honeywell HMC1022IC) for navigation. In order to detect objects, infra-red and bump sensors are used. Lifetime of robomote is extended by a solar cell and a smart charging lithium-ion battery. No localization is supported with Robomote.

Despite all unique features and capabilities offered by this robot due to its miniature size and its low cost (e.g., $150), it is not possible to build Robomote using COTS hardware as in the case of Ragobot. This hinders its use in MSNs.

2.4 Micabot

MICAbot [16] exploits Berkeley MICA platform for its central processing and communication. It is claimed to be both inexpensive (e.g., $350) and flexible which makes it useful for a wide range of experimental goals. While trying to emphasize on the importance of size and cost, the design philosophy of MICAbot is an attempt to increase modularity, functionality and structural stability. The dimensions of the MlCAbot base are only 8.6cm x 6.1cm x 2.1cm.

The design of MICAbot uses an expandable sensor board with a large array of sensing capabilities, which provides a flexible basis for changing experimental needs. The body of the MICAbot is one single unit and supports the interface board, battery pack, and motors (see Fig. 3). The interface board was developed to connect the motor driver and sensors to the MICA platform. Communication is accomplished via an RF Monolithic TRIO00 transceiver at rates up to 115 KB.

To solve the problem of localization, a CCD camera is mounted above the testbed and connected to a 650 MHz computer. The camera provides global coordinate frame information for the MICAbots. Each MICAbot has cue on top of it, which can quickly be identified by the camera positioning experiment. The position of the robots is determined relative to a predetermined global coordinate system. While MICAbot is a low-cost and convenient option, it is not commercially available and not easy to re-design, as the wheels need to be developed separately in house.

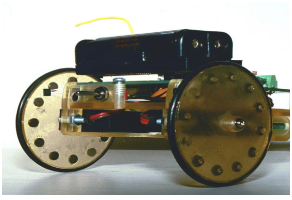


Fig. 3. MICAbot node. Taken from [16].

However, it is important to note that all of the aforementioned platforms, namely Ragobot, Robomote and Micabot are geared purely for MSNs and thus cost and energy-awareness is an important design factor in all of them. These platforms are not expected to perform any actuation or navigate in any terrains. Rather their main mission is sensing and move on demand for performance concerns. Therefore, the choice of application necessitates a closer look at the characteristics of any mobile sensing platform.

2.5 Khepera

In an attempt to develop a mobile robot occupying less than one cubic inch of volume, Khepera [19] as seen in Fig. 4 was built in Microcomputing Laboratory (LAMI) of the Swiss Federal Institute of Technology of Lausanne in 1991. This robot was equipped with a 68HC11 processor, two asymmetric wheels controlled by DC motors with integrated reduction gears, watch batteries, and infrared sensors. Later in 1992, this project was linked with a research on Artificial Intelligence and robotics, resulting in extending the Khepera and adding more features to it. Two DC motors coupled with magnetic incremental sensors, eight analogue infra-red (IR) proximity sensors and the onboard power supply, as well as the possibility of connecting extension modules (e.g., camera, gripper, radio emitter/receiver, etc.) on two different buses are among the additions to the design of this mobile robot.

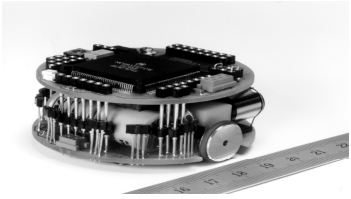


Fig. 4. Khepera, a mobile robot developed in 90's. Taken from [19].

Khepera has served researchers for 10 years, widely used by over 500 universities, and it helped in the emergence of evolutionary robotics. However, Khepera is not a good choice to be used in MSNs, given that it is fairly expensive and not commercially available, and its functionality is limited to a few available extension modules. It does not have 802.15.4 connection capability which limits its direct deployment as a mobile sensing node. Radio extension is mentioned as an optional module but no details on how to do this was not discussed. Nonetheless, its small size enables energy-awareness in movement which can be a plus for MSNs.

2.6 Pioneer 3-AT Robots

In addition to above designed mobile platforms, there has also been a lot of effort to convert the existing robots to mobile sensing platforms that can be used in MSNs. Pioneer 3-DX [20] is one of these robots which is a small lightweight two-wheel two-motor differential drive robot used in Cooperative Testbed designed in [12]. It is equipped with forward-facing and rear-facing sonar sensors and capable of carrying a payload of up to 23 kg (see Fig. 5).

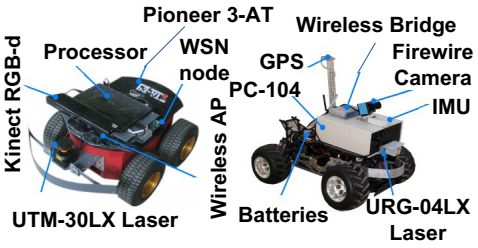


Fig. 5. Pinooer 3-AT mobile robots developed in [12]

The robot's embedded motion controller performs velocity control of the robot and provides robot state & control information including an absolute world position estimate (x, y, θ), battery charge data, and sonar range sensing data. One of the main features of Pioneer 3-DX is that it can be customized and accessorized by choosing from accessories that integrate with the robotic platform.

To program this robot, one can either add an optional internal computer (i.e, one of the available accessories) or connect a personal laptop to the robot. The other important research accessory for Pioneer 3-DX is the 802.11a/b/g Wi-Fi option, which is available for robots already equipped with onboard computers. Several robot motion control functionalities are offered including a low-level velocity control, local position control, trajectory following and random walk. Each of them includes an underlying obstacle avoidance module that ensures a certain configurable distance with the obstacle.

Despite all the functionalities that Pioneer 3-AT provides, as mentioned above, these platforms' initial design motivation was not MSN applications. They are mostly geared for robotics. Therefore, using this robot in MSNs is not cost and energy efficient given its high price and power-hungry features.

2.7 Plantcare Pioneer 2-DX

There has been efforts to also utilize another form of Pioneer robot type, namely Pioneer 2-DX [14]. The characteristics were similar to that of Pioneer 3-AT. PlantCare [14] is an experimental proactive computing platform which builds a zero-configuration and distraction-free system for the automatic care of houseplants using Pioneer 2-DX robots.

The PlantCare system consists of a WSN to measure and report environmental conditions impacting the plants, application logic to monitor these conditions and determine appropriate responsive behavior, and a mobile robot to provide system actuation. In this system, motes are placed both on the robot and in the plants being cared for.

Fig. 6. Plantcare sensor and the mobile platform [14]

The main components of the robot navigation system consist of a reactive collision avoidance module, a module for map building and path planning, and a localization module. The robot is able to communicate with plant sensors via 802.11b. See Fig. 6 for an example. Both the microcontroller and the laser scanner that the robot uses for navigation are connected to a laptop that runs the robot's control and navigation algorithms and is in turn connected to the network via an IEEE 802.11b wireless card. Lastly, the robot has a maintenance bay it uses to automatically charge its own batteries and refill its water reservoir. The bay has a water supply with a spout for dispensing water to the robot, and a charging system matched to the robot's induction coil. All components use probabilistic methods to deal with uncertain sensor information.

Unfortunately, the mobile platform is very expensive (e.g., $1800) due to the high cost of Pioneer 2-DX mobile robot. In addition, the main focus of this platform is to perform actuation. Therefore, it is not suitable for MSNs where collaboration among sensors in terms of messaging is needed. Similar to Pioneer 3-AT, these robots were not designed by considering MSNs and thus their use should be limited (e.g., as a data collector and actuator).

2.8 Acroname Garcia Robots

Acroname Garcia robots are used in some testbeds such as Mobile Emulab [13] because of their relatively reasonable cost ($1100), ease of use and performance characteristics. Using a commercial platform as seen in Fig. 7 avoided the overhead of addressing the many engineering issues inherent in in-house robot design and construction.

In [13] the Garcia robots have an XScalebased Stargate small computer running Linux. A 900MHz Mica2 mote was attached to the robot which can be

controlled by the computer. In this way, sensing as well as communication can be done via the Mica2 mote. However, the robots operate completely wirelessly using 802.11b communication as another option. Motion and steering come from two drive wheels that have a rated maximum of 2m/sec. Six infrared proximity sensors on all sides of the robot automatically detect obstructions in its path and cause it to stop.

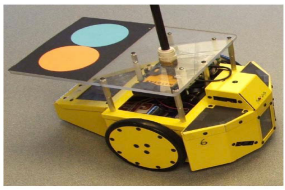


Fig. 7. A sample Garcia robot in [13]

Garcia robots were also used to develop mobile nodes Kansei WSN testbed [9] for sensing at scale. In this case, a TelosB mote was used for communication. Kanse WSN testbed was mainly for stationary sensors (e.g., more than 700 nodes) and thus mobility is not a major component of this testbed. Therefore, the choice of Garcia for the testbed is not motivated by any application. Garcia Robots are mainly produced for robotics applications which makes them inappropriate for MSNs in general. This is because this platform is still big in size and very expensive considering a large-scale deployment. Energy consumption would also be high making it appropriate only for small scale use or employed as a mobile sink to collect data from fixed sensors.

2.9 Sensei-LEGO Mindstorm

This platform is a programmable robotics kit from LEGO. One of the examples is the mobile nodes used in Sensei-UU project [21] which are built with off-the-shelf hardware to make them reproducible and affordable for other researchers. A mobile node in this testbed consists of a Lego Mindstorm robot, a sensor node, and a smartphone. The robot supplies mobility and carries the sensor node and the smartphone. In Fig. 8, the Lego robot carries a smart phone and a TelosB sensor node. The mobile robot uses a laser range finder attached. It is used to measure distances to walls and other objects while the robot moves. By measuring distance, the robots could position themselves when referring to walls and objects using an in-scanned map of a building.

Fig. 8. Sensei-UU mobile node [21]

While sensor node can be used for communication with the other mobile sensor nodes using 802.15.4 standard, the smartphone is used for network management and control via 802.11.

When operating outdoors, Sensei depends on GPS which can be connected through the USB slots or over Bluetooth. For outdoors, a technique based on simultaneous localization and mapping (SLAM) is used to position robots.

LEGO Mindstorm is a robot which is relatively cheaper (e.g., around $300-400) compared to Pioneer or Acroname robots. However, the cost of the smartphone as well as the TelosB motes will increase the price of the whole platform to more than $1000 which is expensive. The main goal of the design is to provide the opportunity for the relocation of the testbed. Therefore, the designers considered lightweight components for navigation which can be an issue compared to other platforms in terms of the ability to navigate. However, such a lightweight design is good for reduced energy consumption.

2.10 iRobot Roomba

Recently developed mobile sensing platforms started relying on the robots from iRobot family due to their cheap costs. Roomba is one of the examples which is normally a mobile vacuum cleaner robot from the company "iRobot" with a diameter of 34cm, a height of 9.4cm and a maximum driving speed of maximum 500 mm/s (1.8km/h). There are various sensors on the robot such as wheel-drop, bumper or infrared sensors to detect obstacles. There is a docking station where Roomba eventually returns with the help of its infrared receiver and a sophisticated algorithm. A PC or microcontroller (e.g., FIT-PC2i in their case) can act as a controller by sending commands via an RS232 to the Roomba.

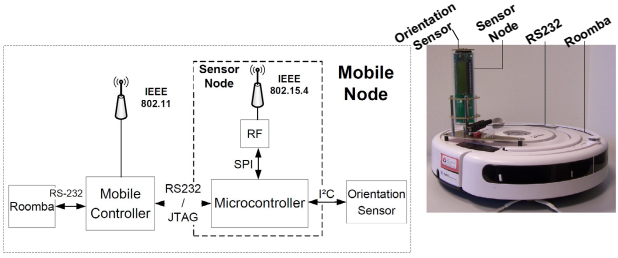


Fig. 9. Roomba along with a sensor [11]

By placing a sensor node on top of Roomba, the researchers in [11] created a mobile sensing platform. The first prototype consisted of a sensor node called TriSOS as illustrated in Fig. 9.

Since the positioning and especially the turning movement of the Roombas is inaccurate and changes with the surface like carpets or ceramics, an additional low power orientation sensor (AMS0805WAH) is used to improve the positioning of the nodes and thus improve the accuracy of the node movement. This orientation sensor is connected to the sensor node and can be read by the mobile controller using RS232 connection.

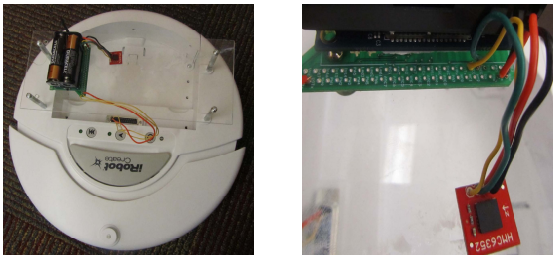
The communication of Roomba nodes is via 802.11 or 802.15.4 connection from the sensor. However, there is also a possibility to have another wireless interface through its mobile controller which is a tiny (27mm x 115mm x 101mm) low power Intel Atom PC running Windows 7 or Linux operation system (FIT-PC2i) placed on top of the Roomba. This will create another interface via 802.11g.

The use of Roomba is motivated from the idea of forming a network among nodes. There is no a specific targeted application and therefore the design mainly focuses on the navigation and communication aspects. COTS and cost are the main motive for the selection of iRobot Roomba.

While Roomba is very economical and easy to develop, there is a limited place for the deployment of sensors on the sensor board. In addition, the cost of mini PC was not mentioned in the original paper.

2.11 iRobotSense

This mobile platform is based on another variant from iRobot family, namely Create [3] which is cost effective ($130) and easily interfaceable with other devices such as sensors and sensor boards. As the sensor and board, it uses IRIS motes from MEMSIC [2] ($99) and an interface board ($15) respectively. The sensor board provides the opportunity to communicate via IEEE802.15.4 with other iRobotSense nodes. The interface board [1] has a variety of models to provide different features. The one that has been used for the design is the simplest and cheapest model, which has a breakout region where all 51 pins are exposed. This board allows easy interfacing of mote devices with peripherals and external sensors via the Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), UART and analog-to-digital converter (ADC) ports. A sample iRobotSense node is shown in Fig. 10.



(a) **(b)**

Fig. 10. (a) A mobile sensor based on iRobot Create platform (b) Wiring with the compass

While iRobotSense cannot do localization and assumes the availability of location information, it can determine the movement direction based on a compass module (HMC6352) from Honeywell ($35). In order to move an iRobotSense node to a certain destination with a known distance, iRobotSense needs to face its wheels towards this destination before it starts moving. To determine the direction of the iRobotSense, it is interfaced with a compass. Nonetheless, for outdoor applications a GPS module can be added to the interface board.

The drive mechanism of iRobotSense consists of two motors controlling each of the two drive wheels while a third wheel in the front provides support. These wheels can be powered and by specifying a turn radius, the robot can move in

any direction. There are limited sensors onboard to detect a cliff, a wheel drop or a forward collision. The robot is programmable via the iRobot Create Open Interface (OI). The OI provides a set of commands and responses for the robot to interact with external devices via the serial interface.

iRobotSense mainly focuses MSN connectivity restoration applications. In this vein, the main focus has been given to the sensing and connectivity components in terms of price. However, the connectivity restoration also consumes a lot of energy which can be taken into consideration in the selection of the robot. For instance, a robot which consumes less energy can be more beneficial in terms of longer lifetime. However, there is a tradeoff here between energy and the navigation. If less energy is consumed, then in most cases the robot cannot move easily in every terrain. This is very crucial in connectivity restoration and thus choose of iRobot also stems from this motivation.

Overall, it is easy to build these nodes by using the COTS products available on the market. The total cost of this platform is roughly $280 which is much cheaper than most of the existing robots. However, more functionality needs to be added such as obstacle avoidance sensors, GPS, etc.

2.12 MiNT-m Roomba

iRobot's Roomba platform has also been used in MiNT project to create mobile nodes called MiNT-mobile (MiNT-m) (see Fig. 11). While there was no sensing hardware as part of this platform, the nodes were able to communicate and move around. These mobile nodes are built from COTS hardware: Routerboard 230 mini PCs with 1-3 Atheros IEEE 802.11a/b/g cards, placed on top of iRobot's Roomba as the mobility platform. One of the wireless cards on each node, operated in RF monitoring mode, is dedicated to collecting traces that are transferred to a central node where they can be visualized in real time.

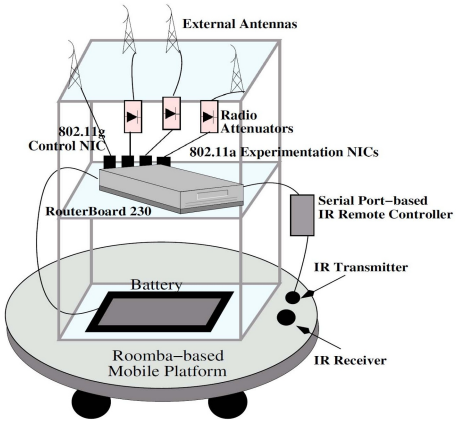


Fig. 11. MiNT-m mobile robotic nodes [6]

Since the custom control GUI enables convenient node configuration, editing and execution of traffic generation scripts, mobility scripts and fault injection scripts. The GUI performs merging of the traces collected by the different nodes and extraction of different network statistics [6].

MiNT-m Roomba nodes were geared for wireless mesh and mobile ad hoc networks and thus the sensing component has not been the focus of the designers. This is also true of the energy consumption characteristics since in general energy

is not a major issue in these networks. The platform is easy to create and has a reasonable price but more research is needed to adapt it for the needs of MSNs.

2.13 Swarmanoid

Swarmanoid [8] presents three robotic platforms (ie. Foot-bot, Hand-bot, Eye-bot) to study heterogeneous robotic swarms comprising various robot types which are able to interact. A uniform hardware architecture is designed and a simulator, ARGoS, is presented.

A common main processor board is designed for the three robot types. A low-level software architecture, ASEBA, is developed to access different modules attached to the robots. Robot behaviors are based on the data obtained from sensors or through communication. A common sensing and communication system is also designed for the robots. The system is based on both infra-red and communication and also provides relative localization.

Fig. 12. Autonomous flying robot, eyebot, to explore indoor environments [8]

While hand-bot can only climb and grasp small objects, foot-bot has autonomous mobility. Thus, foot-bot can interact and provide mobility to hand-bot. On the other hand, eye-bot is an autonomous flying robot which can help the rest of the swarmanoid to detect the objects and direct the actions of other robot types. ARGoS is designed in order to simulate the robots. Therefore, this goal made the designers to focus on the size and ability of the robots rather than sensing and communication. ARGoS provides scalability and flexibility to test the robot controllers.

2.14 Summary and Discussion

In this section, we described the existing mobile platforms that can also do sensing and be employed in MSNs. Table 1 lists all of these nodes with their different characteristics.

From this comparison, we can see that there are two types of trends in the design of mobile sensing platforms. First, trend is to develop nodes that are specifically geared for MSNs and thus focusing on sensing and energy-efficiency. This type of nodes can be used in a lot of monitoring applications with limited mobility but with cost, energy-efficiency and scalability features. The second trend is to employ the existing robots from traditional robotics for application purposes. These designs are not very suitable for MSNs due to their high energy

and price costs. Nonetheless, their navigation capabilities are much better and they are available on the market. This discussion suggests that application requirements play an important role in the selection of mobile sensing platform. A generic mobile node is not possible to design. Rather, based on the needs, a platform should be designed by starting with COTS hardware as much as possible.

Table 1. Existing Mobile Sensing Platforms

Name	Cost	Funct.	Navig.	Comm.	Avail.	Local.	Size	App.
Ragobot	Low	High	Yes	Mica2 900MHz	No	No	Small	Sensing
Robomote	Low	Medium	No	Mica2 900MHz	No	No	Small	Sensing
Micabot	Low	Medium	Yes	Mica2 900Mhz	No	Yes	Small	Sensing
Khepera	High	Medium	Yes	None	No	No	Small	Robotics
Pioneer 3-AT	High	High	Yes	802.11	Yes	Yes	Big	Generic
Plantcare Pioneer 2-DX	High	High	Yes	802.11b	No	No	Big	Agriculture
Acroname Garcia	Medium	Medium	Yes	Mica2 900Mhz	Yes	No	Big	Generic
Sensei LEGO Mindstorm	Medium	Medium	Yes	TelosB/802.11	Yes	Yes	Big	Education
iRobot Roomba	Low	Low	Yes	802.11a/b/g/	Yes	No	Medium	Connectivity
iRobotSense	Low	Low	No	802.15.4	Yes	No	Medium	Connectivity
MiNT-m Roomba	Low	Low	No	802.11a/b/g	Yes	No	Medium	Remote Access
Swarmanoid	High	Low	Yes	802.11	No	No	Medium	Robotic Swarms

3 Mobile Sensor Testbeds

The use of testbeds consisting of the above-mentioned mobile sensing platforms has been limited in the research community. Most of the efforts focused on emulation rather than actual testbeds. A few others were highly specific to the applications and may not be re-deployed for other purposes.

With the development of above mentioned platforms, most of the researchers also created testbeds out of these nodes. In some cases, the design and implementation of the mobile platform also considered the networking options of the nodes. For instance, depending on the radio transmitter on the sink nodes, the nodes are designed to support certain communication protocols. This is also true for sensing hardware selection. The mobile nodes are integrated with well-known sensors that support standard communication protocols such as Zigbee. Cost and energy consumption at the network level may also influence the design of mobile nodes. However, in most cases, existing available radios that can support standard protocols and are available on the market are chosen. In this section, we

describe the testbeds that have used some of the above nodes. We classify these testbeds according to the following metrics:

1. *Research goal*: The goal of the testbed in terms of a research problem.
2. *Application*: The application(s) the testbed can be used.
3. *Communication*: The types of communication among the nodes and other testbed components.
4. *Remote Access*: Whether the testbed provides access to remote users to use the testbed and conduct experiments.

3.1 Cooperative Testbed

A remote testbed merging static WSNs and MSNs is proposed in [12]. This testbed provides an open and modular architecture allowing interoperability between mobile robots and WSN. [12] supports fully distributed approaches as well as centralized approaches. In addition, remote access is possible via a centralized PC.

The testbed WSN consists of static and mobile nodes, with one node mounted on each of the robots as seen in Fig. 13.

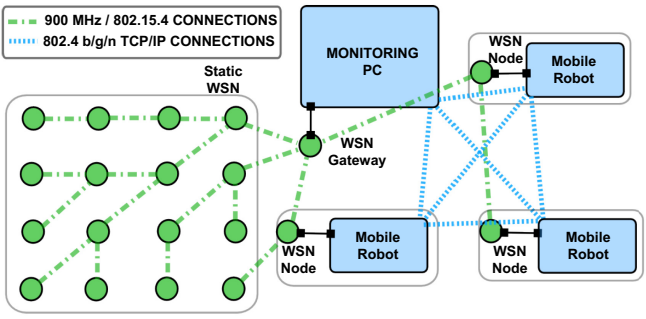


Fig. 13. Coperative testbed with a mobile network talking to a fixed WSN [12]

In this testbed, there are two wireless networks: a wireless LAN (802.11b/g/a) that links the PC that monitors the WSN and the team of robots, and the ad-hoc network used by the WSN nodes via IEEE 802.15.4 protocol.

The sensors are connected to dedicated processor called WSN PC via USB. This WSN PC provides monitoring, reprogramming and logging capabilities and connects the WSN with the rest of the testbed elements through the Local Area Network. Four different models of WSN nodes are available in the testbed: TelosB, Iris, MicaZ and Mica2. TelosB nodes are equipped with SMD (Surface Mounted Devices) sensors whereas all other nodes need to be equipped with MTS400 or MTS300 sensor boards. Also, some have been equipped with embedded cameras such as CMUcam2 and CMUcam3.

The mobile network consisting of mobile nodes is of particular interest to us since we describe their characteristics. In this testbed, the mobile nodes are based on Pioneer 3-AT as discussed earlier. Their communication is via the 802.11 protocol. However, each robot is also equipped with sensors which can communicate with the rest of the sensors via 802.15.4 protocol. The robots can do localization by using cameras and the indoor layout. In addition, time synchronization was also provided. For robots, the well-known Network Time Protocol (NTP) was used [17]. For the WSN nodes, the Flooding Time Synchronization Protocol (FTSP) was implemented.

This testbed was used for several indoor and outdoor experiments/applications. One of the applications was to improve the connectivity of the WSN network by providing redundant paths among the nodes. This is referred to as k-connectivity where k designates the number of available paths between any two nodes. The idea is to identify the locations where mobile robots will act as relays and move them there.

Another application was where the fixed WSN helped the robots to identify their paths when locating objects. Other experiments include localization and multi-robot exploration.

While the testbed provides lots of flexibility via its heterogeneous nodes, the cost of the testbed is very high with the use of Pioneer 3-AT robots as well as the PCs connected to each sensor for processing and communication. This limits the applicability of this testbed outside since the deployment of sensors will be an issue.

3.2 Emulab

Mobile Emulab is a remotely-accessible MSN testbed [13] which was designed for repeatability of experiments. This testbed strives to provide accurate positioning and monitoring, enable automated experiments by both on-site and off-site users, and be built and maintained at relatively low cost using open-source software and COTS equipment.

Mobile Emulab testbed uses six Acroname Garcia robots each of which carries one MicaZ node. The testbed is deployed at an office space of $60m^2$ area. In this space, there are also 25 stationary MicaZ motes as well as 10 motes on the walls. All of the fixed motes are attached to MIB500CA serial programming boards to allow for programming and communication. The 10 wall motes also feature an MTS310 full multi-sensor board with magnetometers that can be used to detect the robot as it approaches. A sample snapshot of this testbed is shown in Fig. 14.

Since there are two ways for communication, one of them is used for control purposes. To this end, a WLAN infrastructure is used for management of the robots (e.g., 802.11b). However, the nodes can communicate with other peers using the sensors attached to robot. This can be Mica2 900Mhz or a TelosB 802.15.4 communication. In the testbed, the authors claim that Mica2 900Mhz was the best option in terms of communication quality.

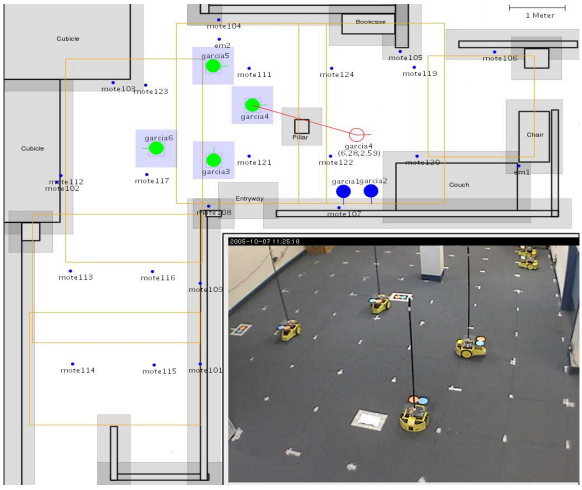


Fig. 14. Mobile Emulab testbed with the layout of the space it was deployed [13]

Mobile Emulab assesses positioning errors. To do this it uses ceiling-mounted cameras for localization in combination with inertial movement measurements. This combination allows the robots to be freely positioned within the designated area.

The main tests conducted with Emulab was on robot localization and the accuracy performance of it. The testbed environment provides cameras for the remote users to be able to watch their tests as they perform experiments remotely.

3.3 MiNT

The Miniaturized Wireless Network Testbed (MiNT) [6][7] at the Stony Brook University is another proposed testbed, consisting of mobile nodes roaming in a 3.66mX1.83m area. First, it provides mobility by means of remotely controlled robots with node hardware mounted on them as seen in Figure 15. Secondly, it supports miniaturization by using RF attenuators to decrease wireless transmission power, thus allowing for a multihop network to be created on an area the size of a large tabletop. The testbed is remotely accessible to outside researchers.

Fig. 15. MiNT Testbed with mobile robotic nodes [6]

While IEEE 802.11a radios are used for experimentation among the nodes, 802.11g is used for robot control commands by the control server. As seen in Fig. 16, there is a control daemon running on a control server which collects inputs from the tracking server and the user, and controls the

movement of mobile robots. It also includes the MOVIE interface for monitoring and control. The tracking server is separate and based on cameras. The server periodically captures images of testbed nodes and processes them to derive the location of each testbed node.

In MiNT, another system is in place to capture packet traces and experiment data. Other novel components of MiNT include automated self battery recharging capability and hybrid simulator interfaces.

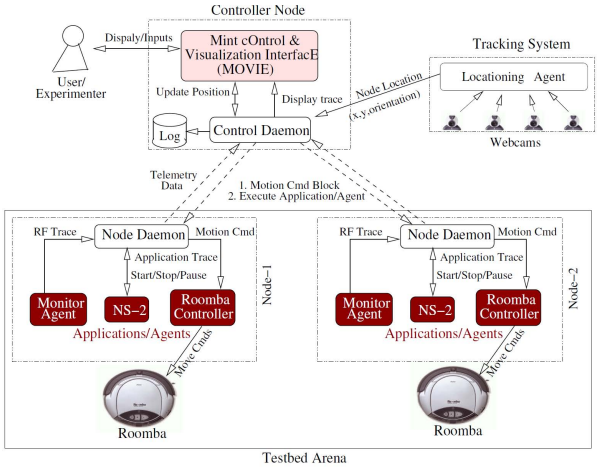


Fig. 16. MiNT Architecture [7]

3.4 Sensei-UU

Another testbed deploying mobile platforms is Sensei-UU testbed [21] that is based on LEGO NXT robots. The main goal of this testbed is to measure the repeatability of the testbed when the experiments are performed several times with similar mobility pattern and also at different times of the day. The testbed is also inexpensive, expandable, relocatable and it is possible to reproduce it by other researchers.

The sensor nodes are attached to sensor hosts. The sensor host communicates with the Site manager which is also the gateway to the testbed and the sensor nodes as seen in Fig. 17. The sensor nodes are normally attached to sensor hosts via their USB interfaces, which are used to observe and control the operation of the sensor nodes using 802.11b/g. Currently IEEE 802.11b/g is used as the control channel. Sensei-UU lets every sensor host keep track of its own position and report its coordinates to the site manager. Mobile sensor nodes use floor markings for navigation and localization. Mobile nodes navigate on a track system that is defined by tape on the floor. The robot follows the track defined by the tape and can be started and stopped arbitrarily by the testbed user. The track system also contains specially marked positions on the track called waypoints, which aid the robot in navigation. The robot is built with two downward facing

color sensors that are positioned so that they can detect the edges of the track. In that way it can follow the tape.

The textbed was mainly used for localization, and interference evaluation (i.e., interference between 802.11 and 802.15.4). It has been shown that the localization of the Sensei-UU is accurate to 1 cm and variations in link characteristics are acceptable to capture fading effects.

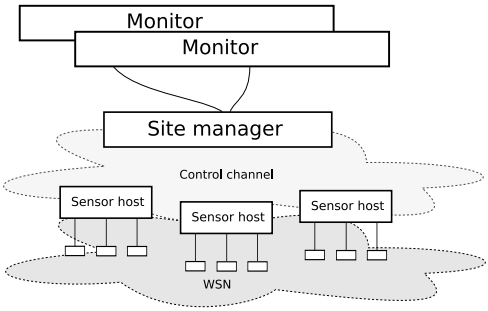


Fig. 17. Sensei-UU Architecture [21]

Through this testbed, one can perform experiments inside and then move them outside for repeatability. It is not remotely accessible but one can re-deploy the testbed using the same configurations via the site manager.

3.5 RoombaNet

Roomba nodes have been used to create a testbed in [11] as depicted in Fig. 18. This RoombaNet testbed has 30 mobile and 30 static nodes and emulates various scenarios such as pedestrian movement or slow car movements. RoombaNet allows for fast and cheap test runs before testing on the final version of protocols for mobile networks in the same way [11].

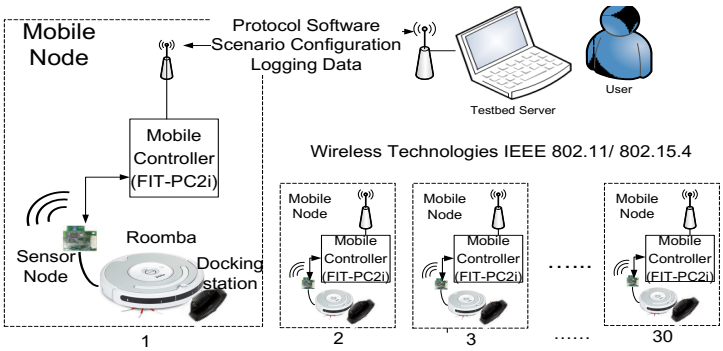


Fig. 18. RoombaNet details [11]

For communication among the Roomba nodes, IEEE 802.11 radio was used as mentioned before. However, the testbed does not support mote like sensors that can sense and communicate via 802.15.4. The main goal of this testbed is to test mobility modeling of humans or vehicles. The testbed is geared for positioning, autonomous movement and collision avoidance which are classical topics in robotics.

RoombaNet is not remotely accessible and there is no management mechanism.

3.6 iRobotSense Testbed

In order to implement and test a partition detection and recovery algorithm in a mobile testbed of iRobotSense nodes, the authors created a network topology of 7 iRobotSense nodes as seen in Fig. 19. Each iRobotSense node is assigned a unique ID and the communication range for each node is set to the lowest value in order to force multi-hopping among the nodes. In this topology, each node is assumed to be broadcasting its readings every second to its neighbors. The neighbors further broadcast these readings until they are received by the sink node which is assumed to be node N_7 in Fig. 19. Since iRobotSense uses a IEEE 802.15.4 based MAC protocol, same protocol was used at the MAC layer in the testbed. However, no specific routing protocol was implemented as flooding was used.

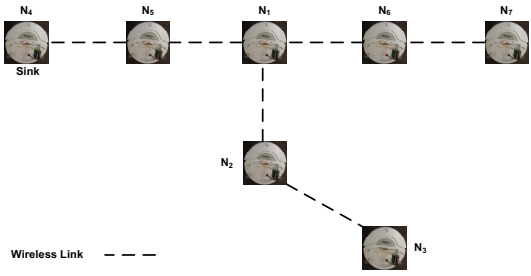


Fig. 19. MSN topology using 7 iRobotSense nodes

This testbed is specific to certain application and does not provide remote access. In addition, the localization is assumed to be available via GPS. The testbed, implements an orientation algorithm for the nodes in order to determine the right directions before their movement starts.

3.7 Kansei Testbed

The Kansei WSN testbed [9] is a large-scale WSN testbed of Ohio State University (OSU). Since it includes five robotic mobile nodes that each carry an Extreme Scale Mote and a Tmote Sky, we also include this testbed within the category of MSN testbed. The robots in this testbed can move within an array of stationary nodes and act as mobile sensor nodes or trigger sensor events in the stationary nodes.

The used robots are from Acroname, Inc. with built in motor-boards and a Stargate single-board computer as an interface. The Stargate on each robot features an 802.11b radio. In addition, each robot contains an XSM (a variant of Berkeley motes developed at OSU) and TMote Sky node to communicate with the stationary arrays as well as to run native code for the XSM and Tmote platforms.

The goal of these mobile nodes in the Kansei testbed is to catch the intruders detected by the sensors. The sensors also provide the tracking for the mobile nodes.

3.8 MOTEL

A robotic-assisted mobile wireless sensor network testbed is presented in [4]. MOTEL addresses two challenges in mobile WSN testbeds: backchannel management and mobility. For backchannel management, a solution called Flexible Runtime Management Software Architecture for WSN (FLEXOR) is offered. On the other hand, Multi-Robot Architecture for Coordinated Mobility (MuRobA) is presented as the solution for mobility.

FLEXOR is a platform-independent architecture for programming and managing WSNs. FLEXOR enables remote function call mechanism through Callback Manager, parameter change of the modules and remote debugging, status inquiries and logging. Software components can also be exchanged at run-time by using Images defined earlier. Images comprises specifications describing software components. Thus, by exchanging a single command, re-programming and reboot of the complete system can be avoided. FLEXOR also has extensive graphical support. MOTEL renders backchannel unnecessary through FLEXOR.

MuRobA enables mobility of the nodes by piggybacking them on mobile robots. The top of the robots are marked by colorful dots to perform localization. One or several cameras are employed overlooking the robots in order to track them. *FleetManager* interprets the information collected from the cameras and decides for movement. The movement commands are sent to robots through bluetooth. There is no limitations on the type and the number of the cameras and the robotic platforms.

The sensor nodes run on FLEXOR which are piggybacked on robots, which run MuRobA. FLEXOR and MuRobA are platform-independent architectures. In [4], TelosB and e-puck [18] are used as the sensor network platform and mobile platform respectively.

3.9 SensLAB

A large scale open WSN testbed, SensLAB, is introduced in [5]. SensLAB offers an open access, multi-user evaluation tool composed of 1024 nodes in 4 sites. Sensors in each site have specific characteristics to provide heterogeneity. Each site hosts 256 nodes which are able to communicate with their neighbors through their radio interface. Furthermore, each node can be configured as a sink node so that they will be able to communicate with other sink nodes.

Testbed is remotely accessible through web-portal without any restrictions on programming language or operating system. Users can setup their experiments by configuring the testbed based on their requirements such as the number of nodes, sensor and radio features, topology, duration of the experiments, etc. Users also have real-time control on the experiment so that the nodes can be turned off to mimic crashes or send fake data to tamper with transmissions.

Localization applications can also be evaluated using SensLAB. A sample tracking application is demonstrated in [5] where 32 out of 256 nodes are mobile. Mobility is provided to the nodes by using electric trains and localization is achieved through RSSI of the signals. Real-time localization computation through web-portal is illustrated in Fig. 20.

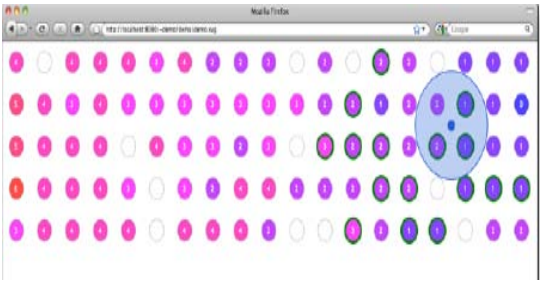


Fig. 20. Real-time localization computation through web-portal [5]

3.10 Summary and Discussion

A summary of the included MSN testbeds based on our evaluation criteria is provided in Table 2. Most of these testbeds are mainly designed for realization of stationary WSNs rather than MSNs. Mobility is, most of the time, provided as an additional feature. In most cases, the size of the testbeds are smaller and they are used for proof-of-concept. Another important observation is that, there are no large-scale testbeds which employ the mobile sensors such as Robomote, Ragobot or Micabot which are designed specifically for WSNs for low-cost, low-energy and scalability purposes. It would be interesting to see the characteristics of a MSN in large-scale that uses these nodes under different goals. While iRobotSense Testbed and RoombaNet are some examples which focuses on WSN applications, they are still not large and they do not provide a comprehensive implementation of all traditional WSN protocols.

Table 2. Existing MSN Testbeds

Name	Goal	Application	Acces.	Communication
Cooperative	MSN-WSN Cooperation	Various	Yes	802.11b/g/n
MinT	Miniarization	Generic	Yes	802.11b/g
Emulab	Repeatability	Localization	Yes	Mica2 900Mhz
Sensei-UU	Repeatability	Localization	No	802.15.4
iRobotSense Testbed	Connectivity Restoration	Generic	No	802.15.4
RoombaNet	Pre-product Runnning	Mobility Modeling	No	802.11
Kansei	Sensor-actor design	Intruder Interception	No	TelosB
MOTEL	Testing and Management	Generic	No	802.15.4
SensLAB	Testing and Management	Generic	Yes	802.15.4

4 Conclusion and Future Issues

In this chapter, we analysed the existing mobile sensing platforms and the MSN testbeds that were created out of these nodes. The analysis indicated several

trade-off among cost, size, functionality and availability. Among the existing mobile platforms, the robots from the iRobot family seem to be more promising than the others in terms of availability and cost. Custom designed mobile platforms are also very attractive but their commercialization is needed for widespread adoption.

Among the testbeds, the issues regarding robotics such as navigation and localization seem to be the main focus. There are not many applications targeted for MSN testbeds. 802.11 standard is still very widely used. Nonetheless, this is not suitable for resource constrained sensors and thus 802.15.4 should be applied.

There are still a number of issues to be researched under MSNs:

1. *Inter-sensor communication*: Inter-sensor communication has not been well investigated. Typically 802.11 has been tested heavily. Testing of 802.15.4 or the new standards such as 6lowPAN [22] should be studied in more details.
2. *Large-scale Testing*: Current testbeds use a few number of mobile nodes but does not look at the issues at the scale. The issue of interference, sensing and routing need to be investigated in a holistic manner.
3. *Energy Consumption*: The mobile nodes are battery operated and their movement consumes the most energy. While energy consumption of messaging has been tested heavily in traditional WSNs, the energy consumption patterns for mobile nodes have not been well explored.
4. *Hardware Design*: Various design strategies to connect sensors and robots have been mentioned in the paper. However, the details of these connections at the hardware and software level have not been well described. The studies should instruct people in such a way that the replication of their design would be easily applicable.

References

1. Interface board, `http://shop.samraksh.com/product.sc?productId=3` (retrieved in May 2012)
2. Iris mote, `http://www.memsic.com` (retrieved in May 2012)
3. irobot, `http://www.irobot.com` (retrieved in May 2012)
4. Forster, A., Forster, A.„ K.G.D.P.S.G., Gambardella, L.: Motel: A mobile robotic-assisted wireless sensor networks testbed. In: 9th European Conference on Wireless Sensor Networks (EWSN) (2012)
5. Burin des Roziers, C., Chelius, G., Ducrocq, T., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., Noél, T., Vandaele, J.: Using senslab as a first class scientific tool for large scale wireless sensor network experiments. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) NETWORKING 2011, Part I. LNCS, vol. 6640, pp. 147–159. Springer, Heidelberg (2011)
6. De, P., Raniwala, A., Krishnan, R., Tatavarthi, K., Modi, J., Syed, N.A., Sharma, S., Chiueh, T.-C.: Mint-m: an autonomous mobile wireless experimentation platform. In: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, MobiSys 2006, pp. 124–137. ACM, New York (2006)
7. De, P., Raniwala, A., Sharma, S.: Mint: a miniaturized network testbed for mobile wireless research. In: Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2005, vol. 4, pp. 2731–2742 (2005)

8. Dorigo, M., Floreano, D., Gambardella, L., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., OGrady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A., Vaussard, F.: Swarmanoid: A novel concept for the study of heterogeneous robotic swarms, vol. PP, p. 1
9. Ertin, E., Arora, A., Ramnath, R., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., Cao, H., Nesterenko, M.: Kansei: a testbed for sensing at scale. In: Proceedings of the 5th International Conference on Information Processing in Sensor Networks, IPSN 2006, pp. 399–406. ACM, New York (2006)
10. Friedman, J., Lee, D., Tsigkogiannis, I., Aghera, P., Dixit, A., Levine, D., Chao, D., Kansal, A., Kaiser, W., Srivastava, M.: Ragobot: A new hardware platform for research in wireless mobile sensor networks. In: Proceedings of International Conference on Distributed Computing in Sensor Systems (2005)
11. Hail, M.A., Pinkowski, J., Teubler, T., Danckwardt, M., Pfisterer, D., Hellbrck, H.: Roombanet - testbed for mobile networks. In: ECEASST (2011)
12. Jimnez-Gonzlez, A., Dios, J.R.M.-D., Ollero, A.: An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. In: Sensors 2011 (2011)
13. Johnson, D., Stack, T., Fish, R., Flickinger, D.M., Stoller, L., Ricci, R., Lepreau, J.: Mobile emulab: A robotic wireless and sensor network testbed. In: Proceedings of the 25th IEEE International Conference on Computer Communications, INFOCOM 2006, pp. 1–12 (April 2006)
14. LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S.B., Sikorski, K., Fox, D., Borriello, G.: Making sensor networks practical with robots. In: Proceedings of the First Int. Conference on Pervasive Computing, Pervasive 2002, London, UK, pp. 152–166 (2002)
15. Ma, Y.T.: Mobility in Mobile Sensor Networks: A study of sensing performance and privacy. PhD thesis, Purdue University (December 2010)
16. McMickell, M.B., Goodwine, B., Montestruque, L.A.: Micabot: a robotic platform for large-scale distributed robotics. In: ICRA, pp. 1600–1605 (2003)
17. Mills, D.: Network time protocol (version 3) specification, implementation (1992)
18. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S.: christophe Zufferey, J., Floreano, D., and Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59–65 (2009)
19. Mondada, F., Franzi, E., Guignard, A.: The development of khepera. In: Proceedings of First International Khepera Workshop, Paderborn (1999)
20. Pioneer P3-DX Robot, `http://www.mobilerobots.com/`
21. Rensfelt, O., Hermans, F., Gunningberg, P., Larzon, L.: Repeatable experiments with mobile nodes in a relocatable wsn testbed. In: 2010 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (DCOSSW), pp. 1–6 (June 2010)
22. Shelby, Z., Bormann, C.: 6LoWPAN: The Wireless Embedded Internet. Wiley Publishing (2010)
23. Sibley, G.T., Rahimi, M.H.A.: Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In: IEEE International Conference on Robotics and Automation (2002)

Part IV
Mobility Management

Wireless Sensor Network Connectivity and Redundancy Repairing with Mobile Robots*

Alberto de San Bernabé Clemente, José Ramiro Martínez-De Dios, Carolina Regoli, and Aníbal Ollero

Robotics, Vision and Control Research Group, University of Seville, Spain
Camino de los Descubrimientos, 41092 Sevilla, Spain
{adesanbernabe,jdedios,aollero}@us.es,
carolina.regoli@gmail.com

Abstract. This chapter presents a method for repairing the connectivity and redundancy of a WSN using mobile robots. It comprises three mechanisms: diagnosis, connectivity repairing and redundancy repairing. During the diagnosis stage the robots survey the scenario learning all required information of the problem. The connectivity repairing mechanism, which takes place after the diagnosis stage, finds the best deployment locations to ensure that the WSN has 1-connectivity. The redundancy repairing mechanism finds the locations where the deployment of new nodes best improves fault tolerance to node failures. The proposed scheme does not use any parameters or assumptions since it acquires all the required information during the diagnosis stage. Besides, its solution is very close to the optimal solution –in all the experiments performed it differed in one node maximum– but requires only a fraction of the time required by the optimal method. The proposed method has been evaluated in simulations and has been validated in experiments carried out in the *CONET Robot-WSN Integrated Testbed*.

1 Introduction

Cooperation among mobile robots and Wireless Sensor Networks (WSN) enhances their individual performance, providing wide variety of complementarities. Robots mobility and their capability of carrying sensors and equipment are useful to enlarge the sensing range and accuracy of static WSN nodes and enlarge the communication ranges of static WSN nodes. Mobile robots have been proposed for WSN localization [3], WSN deployment [21] and WSN data retrieving [19], among others.

Recently, WSN repairing using robots has attracted significant interest. Some of these methods, such as [2,26], detect faults in the network such as coverage holes but do not propose a way to mitigate these faults. Many others focus on solving connectivity failures, see [22,27,10], and most of them propose mobile nodes as tools to repair network connectivity. However, the great majority have been tested only in simulations

* This work was supported by PLANET (European Commission FP7-257649-ICT-2009-5), CLEAR (DPI2011-28937-C02-01) and EC-SAFEMOBILE (European Commission ICT-2011-288082).

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*,
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_9,

and only few, such as [6,15], were used in real experiments. Existing methods aim at repairing only connectivity disregarding the fact that WSN nodes are rather fragile and a connected network can be suddenly disconnected even if only one node fails.

This chapter presents mechanisms for integral WSN repairing. Their objective is not only to repair connectivity but also to ensure a certain level of redundancy. The method can be divided in three mechanisms: diagnosis, connectivity repairing and redundancy repairing. In the first one the mobile robots explore the scenario in order to identify the WSN status, topology and to estimate the location of the deployed nodes. In connectivity repairing, using this information, the method first computes the locations where new WSN nodes should be deployed in order to ensure that the connectivity of the WSN is repaired. Then, mobile robots are commanded to deploy WSN nodes at the computed locations. These new nodes are integrated in the pre-existing WSN. If the connectivity of the resulting WSN is confirmed as repaired, in the next stage redundancy repairing is applied. Redundancy repairing deploys new WSN nodes at specific locations until a certain level of redundancy is achieved.

The proposed scheme does not use parameters nor requires any assumptions on the WSN nor on the WSN nodes being used. It acquires all the required information during WSN diagnosis stage and only needs as input the desired level of WSN redundancy. The proposed method has been implemented in simulations and has also been experimentally demonstrated in the *CONET Robot-WSN Integrated Testbed* [12].

This chapter is organized as follows. Related work is discussed in Section 2. Section 3 provides a general description of the proposed method. The diagnosis, connectivity repairing and redundancy repairing stages are described in Sections 4-6, respectively. Validation experiments are presented in Section 7. Section 8 closes this chapter with the conclusions.

2 Related Work

Fragility of individual nodes has motivated the development of a good number of methods for WSN diagnosis and repairing. Some focus on the identification of faults and misbehaviors in the networks [2,8], whereas others aim to develop a model of the faults that can occur in WSNs [26]. Others focus on detecting specific types of failures in the network, mainly coverage holes, such as in [23] and [28].

There are a good number of works that propose methods to solve connectivity failures. Some of them study the placement of relay nodes to repair the connectivity, see e.g. [22], while others propose using mobile nodes than can be relocated in order to substitute a failing node or to join two disconnected parts of the network, see [27,10], among others. For example, DARPA LANDROIDS program [5] uses mobile radio relay platforms to improve communications connectivity in non-line-of-sight communications environments such as urban settings. Work [4] uses predictive reasoning to compute the relocation of the mobile nodes. In this work sensor nodes can predict the performance of the WSN in terms of coverage when a node moves in a given direction. Others compute locations where WSN nodes should be deployed in order to repair the connectivity while optimizing utilities such as coverage or energy consumption, see [16,9], among others.

Much research has been carried out based on theoretical analyses. Most of them have been tested only in simulations and their results in many cases show to be unrealistic

when experimented in real settings. Only a few works include experimental validation. In [6] a flying robot is employed to deploy nodes on the ground. Once on the ground, the nodes compute their connectivity map in a distributed way. If the network is disconnected, an algorithm determines waypoints for the helicopter to drop additional nodes. In one of the experiments performed in [15] a mobile node was employed to re-establish the link between static nodes.

However, most works focus on connectivity repairing and disregard redundancy, which is critical for WSN robustness in real applications. Several concepts for redundancy in WSN have been defined. Work [1] presented the concept of N-redundancy of a node and proposed: a method to compute N-redundancy, a technique to estimate the repair time and a repairing algorithm that minimizes the repair time. However, like the aforementioned works, its objective was only to repair the connectivity of isolated parts of the network and did not consider redundancy repairing. The concept of redundancy has also been used in works that study the density of deployments, such as [18,17], but these works did not address network repairing.

This chapter proposes a method for integral WSN repairing which involves connectivity and also redundancy repairing. The proposed method makes no assumptions except the approximate region where nodes are deployed. To the best of our knowledge, it is the first work that validates experimentally the use of robots for ensuring a certain level of redundancy.

3 General Description

Consider a scenario where a number of static WSN nodes have been deployed, but their exact position or the WSN topology are not known. This case is realistic even when the nodes have been manually deployed if we consider the inaccuracies in the GPS receiver. For instance, in an environmental monitoring application WSN nodes could have been thrown randomly from an Unmanned Aerial Vehicle (UAV) over the area of interest. We know the approximate area where nodes were deployed but not their exact location. The objective of the WSN is to collect data from each node in a Base Station (BS). However, the nodes will be located at unknown locations and eventually will be disconnected from the base. We want to devise a method that ensures a certain level of connectivity and redundancy among the deployed nodes. WSN repairing can be triggered after nodes deployment or on demand when the BS detects WSN malfunctioning. The approach selected is to deploy new nodes instead of re-locating existing nodes.

The proposed method employs the concepts of N-connectivity and N-redundancy. N-connectivity is a metric used to define the minimum number of nodes that need to be removed in order to partition a graph. If a graph has N-connectivity, it remains connected even if any N-1 nodes are removed. If a network has 0-connectivity it means that there are disconnected nodes.

N-redundancy, defined in [1], represents a local measurement of the goodness of the connectivity among the neighbors of each node. A node *i* is N-redundant if we need to remove at least N nodes neighbors of node *i*, besides removing node *i* itself, in order to break the connectivity between two neighbors of node *i*. 0-redundancy nodes are fragile points of the network: if node *i* fails the network becomes disconnected. We consider a

network robust if all its nodes have at least 2-redundancy. A WSN is N-redundant when all its nodes have N-redundancy or higher.

N-connectivity is a metric for the whole network while N-redundancy is a metric for each node. They offer interesting complementarities. N-connectivity naturally allows analyzing if the network is connected. However, being a global metric, it is not very useful to detect the fragile points of the network. For improving fault tolerance we prefer to use N-redundancy, which is a local metric and naturally finds the nodes in the network that are more critical. Knowing which are the fragile nodes is exploited by our method, which drives the search of the repairing locations using heuristics and reducing computer burden.

The proposed scheme deploys new WSN nodes such that the resulting WSN has at least 1-connectivity and N-redundancy, a desired level N of redundancy. The scheme uses three mechanisms:

- **Diagnosis**: One or several robots carrying onboard WSN nodes survey the scenario to learn the status of the WSN, including the locations of the deployed nodes and the WSN topology. It also learns a model of the distribution of Packet Reception Rate (PRR) with distance for the deployed static nodes.
- **Connectivity repairing**: This mechanism deploys WSN nodes to ensure that the resulting WSN has 1-connectivity. If the network contains nodes or clusters of nodes disconnected from the base station, the method computes the best locations where new nodes should be deployed in order to reduce the number of clusters in the network to one.
- **Redundancy repairing**: It takes place when the WSN has 1-connectivity. It iterates computing the locations where new nodes should be deployed in order to best improve the redundancy of the static nodes that have redundancy lower than N. The iterations continue until all the nodes in the network achieve the desired N-redundancy.

4 WSN Diagnosis with Mobile Robots

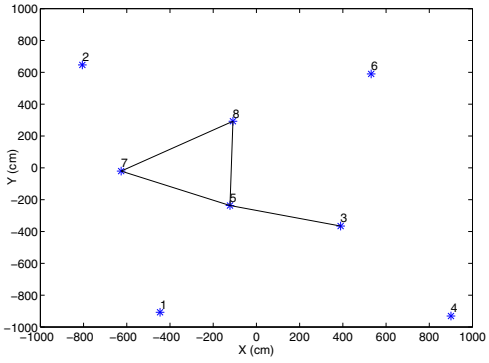
Assume we have available one or more mobile robots that carry one onboard WSN node. Each robot can measure its own location and can communicate with its onboard node using a bidirectional protocol that enables transmission of commands, requests and data. The scenario is discretized into cells and one or more mobile robots are commanded to survey the scenario passing over the centers of each of the cells. During the survey robots make measurements and ask static nodes. As a result, each robot computes the location of the nodes it has discovered, the topology of the static WSN and gathers data to compute a PRR-range model of the static nodes.

During the survey when a robot is close to the center of a cell it commands its onboard node to broadcast a beacon packet. Static nodes that receive the beacon packet respond by transmitting a response packet. When the onboard node receives the packet it measures its Radio Signal Strength Indication (RSSI). The location of each node can be determined from these RSSI measurements. A number of RSSI-based localization methods have been developed including range-based methods, such as multilateration

[24], or range-free methods, such as ROC-RSSI [14], which rely on geometric considerations, or fingerprinting methods, such as [11], which compare measurements with a previously obtained RSSI map. In our case, the location of a node is simply assigned to the cell with the highest RSSI value.

This survey also allows obtaining the topology of the WSN. Every static node dynamically learns which are its neighbor nodes by a simple protocol that periodically broadcasts packets among static nodes. We consider that node *j* is neighbor of node *i*, i.e. *i* and *j* are connected, if node *i* has received more than a certain percentage of responses from node *j* over the last *m* broadcasting periods. Thus, when a static node *i* responses to the robot node, the packet contains the ID of the sender and also of the sender neighbors. Thus, each robot can easily build the local connectivity matrix *LCM* of the static nodes in the area it surveyed.

Each robot transmits its *LCM* to the Base Station, which joins them all. A global connectivity matrix *GCM* of size *n x n* is generated, being *n* the number of nodes discovered. If entry *GCM(i,j)="1"* it means node *j* and *i* are neighbors. Figure 1 shows a simple example and its global connectivity matrix. Neighbor nodes are depicted with a line in the figure.



Global Connectivity Matrix
GCM

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Fig. 1. Random deployment and its global connectivity matrix *GCM* in a simple simulation

The Base node connected to the BS is also one of the static nodes discovered. *GCM* contains the topology of the network and allows us to know if all the nodes are connected, and the network has at least 1-connectivity, or instead there are isolated parts of the network and therefore is divided in two or more clusters. Algorithm 1 shows a simple method that computes a cluster matrix *C* that represents the groups of connected nodes in the WSN topology. Along this paper we will use the term cluster when referring to isolated groups of connected WSN nodes.

C is an *n x n* matrix which entries can be either "1" or "0". If *C(i,j)*="1" it means that node *j* belongs to cluster *i*. Thus, only rows in *C* with at least one entry with value "1" represent a cluster. Rows in *C* with all entries "0" do not represent a cluster. *C* allows expressing cases in which all nodes are disconnected from each other (*n* clusters) and also cases in which all nodes are connected (1 cluster).

Algorithm 1 performs similarly as depth first search to create *C*. First, *C* is initialized with a zero *n x n* matrix. The algorithm searches the first entry in *GCM* with value "1" and declares a new cluster. Then, it finds the nodes belonging to that cluster iteratively analyzing the connected nodes either directly or through other nodes of the same cluster (line 6 of the algorithm). Entries in *GCM* with values "1" already assigned to a previous cluster are not analyzed. The algorithm ends after analyzing all the entries in *GCM*.

At the end of the algorithm every row of *C* that contains at least one "1" represents a cluster of the network. For every row, the columns filled with "1" correspond to nodes that are in that cluster. Figure 2 shows matrix *C* corresponding to the example in Fig. 1. The resulting five clusters are $C_1 = \{1\}$, $C_2 = \{2\}$, $C_3 = \{3,5,7,8\}$, $C_4 = \{4\}$ and $C_5 = \{6\}$: it requires connectivity repairing.

Algorithm 1. Computation of *C* from *GCM*

1. Cluster Matrix *C* is initialized as n x n matrix of zeros
2. j=1, $C(j,1) = 1$
3. **for** k=1:n **do**
4. **if** $\sum_i C(i,k) = 0$ **then**
5. $C(j,k) = 1$
6. $C(j,m) = 1 \; \forall$ node *m* that is neighbor of node *k* in *GCM*
7. j++
8. **end if**
9. **end for**

Cluster Matrix
C

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 2. Cluster matrix *C* computed for the example in Fig. 1

In the diagnosis stage also a PRR-range model for the deployed nodes is computed. Each robot is assumed to know its location during the survey. Also, the location of the static nodes has been estimated as described above. Thus, it is possible to build a PRR-range model using the response packets transmitted by the static nodes and the distances between the robot and the static nodes when the beacon packets were transmitted. In this model range is divided in intervals and the mean percentage of messages received over all the messages broadcasted is taken as the PRR in this interval. Figure 3 shows the PRR-range model obtained in network diagnosis experiments performed in the *CONET Robot-WSN Integrated Testbed*.

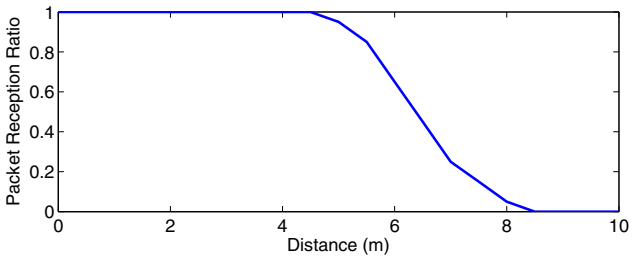


Fig. 3. PRR-range model obtained in the *CONET Robot-WSN Integrated Testbed*

Finally, it is decided if there is need or not to activate the connectivity repairing mechanism. If network is composed of more than one cluster, the connectivity repairing stage is triggered. If the network is composed by only one cluster of nodes, there is no need to activate connectivity repairing.

5 WSN Connectivity Repairing

This mechanism calculates the locations where new nodes should be deployed in order to optimally improve WSN connectivity by joining disconnected clusters. At the end of the previous stage the locations of the nodes and the topology of the clusters are known. The connectivity repairing mechanism is divided in two steps, see Algorithm 2. In the first step new nodes are deployed to join nearby clusters in larger clusters –we call them superclusters. In the second step all superclusters are joined in one cluster covering the entire network, achieving 1-connectivity.

To group clusters in superclusters first we need to know which clusters can be joined into the same supercluster. Using the PRR-range model obtained in the diagnosis stage, R is computed as the range that ensures a desired PRR level, $PRR(d) \geq X, \forall d \leq R$. X is taken as 0.85 in the experiments. If the closest nodes of two different clusters are separated by a distance lower than $2R$, then these clusters can be joined by deploying a new node. These clusters are considered neighbors and can be grouped into the same supercluster.

Consider a WSN with N disconnected clusters. We can compute the cluster connectivity matrix *CCM*. If entry *CCM(i,j)="1"* it means that clusters i and j are neighbors. Otherwise, *CCM(i,j)="0"*. The size of *CCM(i,j)* is *NxN*. *CCM* is also a connectivity matrix. We can also use Algorithm 1 to obtain superclusters of neighbor clusters. The supercluster matrix *SC*, of size *NxN*, is obtained after applying Algorithm 1 to *CCM*. If *SC(i,j)="1"* it means that cluster j belongs to supercluster i. In short, a recursive approach is followed to define how WSN nodes are organized at different levels.

Figure 4 shows an example with 8 nodes, which after the diagnosis stage, are organized in five clusters: $C_1 = \{1\}$, $C_2 = \{2,3,4\}$, $C_3 = \{5,6\}$, $C_4 = \{7\}$ and $C_5 = \{8\}$. C_1 and C_2 are at a distance lower than $2R$, thus they are neighbor clusters. The same applies to C_1 and C_3, C_2 and C_3 and C_4 and C_5. Thus, these clusters are organized in two superclusters: $SC_1 = \{C_1, C_2, C_3\}$ and $SC_2 = \{C_4, C_5\}$. Dashed lines in Fig. 4 shows the

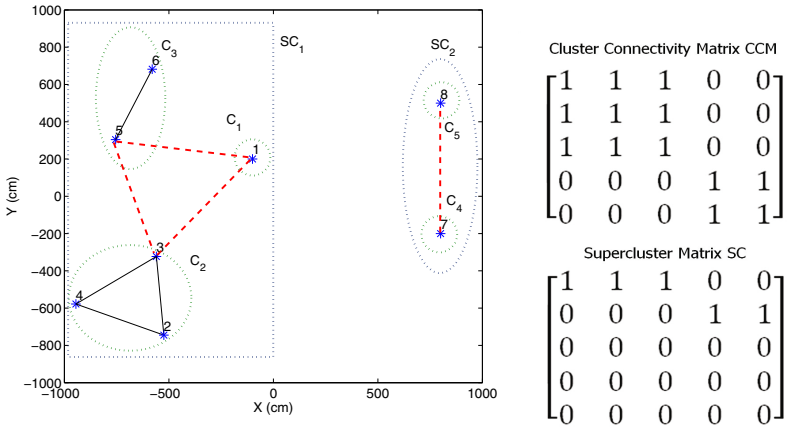


Fig. 4. Superclusters in an example. The links between clusters are represented by dashed lines. *CCM* and *SC* are also shown.

imaginary lines of length $R < d < 2R$ that would join the closest nodes of two different neighbor clusters.

The next step is to compute the locations where to deploy the minimum number of nodes that ensure 1-connectivity within each supercluster. We interpret the graph resulting from considering the WSN topology (in black in Fig. 4) and the imaginary lines between clusters (dashed in Fig. 4). As shown in Fig. 4 two different cases can arise. In SC_1 imaginary lines between clusters, for brevity from now on *ilbcs*, form a polygon (a triangle). In SC_2, the *ilbcs* do not form a polygon.

When the *ilbc* in a supercluster do not form a polygon the repairing location is selected among the cells which distance to both vertices of the *ilbc* are simultaneously lower than R. These locations ensure connectivity between both clusters. Then, the repairing location is selected analysing the effect of deploying a node at each of the selected cells. The location is selected as that in which deploying the node maximizes the repairing impact of the deployment, i.e. creates the highest number of new connections. If case of tie, the quality of the new links is analyzed: the repairing location is selected as the cell with the highest average PRR of the new connections originated by deploying the node in that cell.

In case *ilbcs* form a polygon, the method identifies the locations within the polygon where the minimum number of nodes should de deployed to establish 1-connectivity in the supercluster. First, the method tries to join all the clusters using only one node. If there is at least one cell within the polygon which distance to all the clusters is lower than R, then the supercluster can be connected using only one node. If there are more than one possible cell, the best is selected measuring the repairing impact as described above. Otherwise, the algorithm tries to connect all the clusters using two new nodes. A combination of two cells within the polygon that enable connection among all clusters is selected. If several solutions exist, the best combination using the above criteria is selected. In case no combination of two cells exist, the procedure is iteratively repeated trying solutions with one more node until a solution is found. Notice that a solution is

guaranteed to exist with a number of nodes equal to the number of *ilbcs* that form the polygon. Also the number of *ilbcs* will be lower or equal to the number of clusters in the supercluster.

Figure 5 shows with asterisks the set of deployment locations selected to join the clusters into superclusters SC_1 and SC_2. The connections originated from the new nodes are depicted with dashed lines. In this case the new node 9 joins C_1, C_2 and C_3 into supercluster SC_1 and node 10 joins C_4 and C_5 into SC_2.

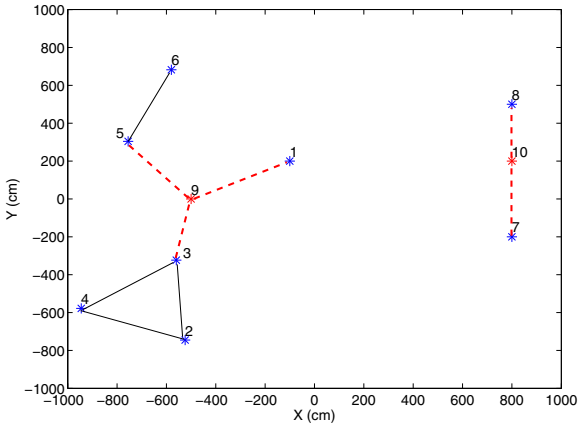


Fig. 5. Solutions selected to join C_1, C_2 and C_3 into supercluster SC_1 and C_4 and C_5 into SC_2. The selected deployment positions for the new nodes are represented with asterisks.

The next step is to join all superclusters in one. The first step is to identify the shortest imaginary lines that join two different superclusters. The length of these lines are higher than $2R$: it is not possible to join two superclusters with only one node. Thus, the approach is to deploy *chains* of new nodes along these imaginary lines. The minimum number of lines needed to join S superclusters is *S-1*. However, there are usually more than one combination of *S-1* lines between superclusters that join all the network. The method selects the combination of lines that join all superclusters with the lowest total length.

For those imaginary lines the optimal locations in which to place nodes is sought between cells which distance to these lines is lower than a certain distance d. The number of nodes Nn that should be deployed depends of the length of the line, $Nn \geq L/2R$, being L the length of the line and R the range that ensures a certain level of PRR according to the PRR-range model computed in the diagnosis stage. The locations of the nodes along the lines are selected using their repairing impact: solutions that join the superclusters with the highest number of new connections are selected; and in case of tie, the solution which new connections have the highest average PRR is selected.

Figure 6 shows the solution selected to join the superclusters in the example in Fig. 5. The shorter lines joining superclusters SC_1 and SC_2 is that between nodes 1 and 10. Two nodes are necessary to join both superclusters. The proposed repairing locations

Algorithm 2. Connectivity repairing algorithm

```
1.  Step1: Join clusters into superclusters
2.  Compute CCM
3.  Obtain SC applying Algorithm 1 to CCM
4.  Determine all imaginary lines between clusters ilbcs
5.  for Each supercluster in SC do
6.     if ilbcs do not form a polygon then
7.        Select the cell close to ilbcs that produces the best connectivity impact
8.     end if
9.     if ilbcs form a polygon then
10.       Determine minimum number of cells that join the clusters
11.       Select the combination of cells that produces the best connectivity impact
12.    end if
13. end for
14.   Step2: Join superclusters
15. Determine the shortest set of lines that joins all superclusters
16. for Every line do
17.    Determine the minimum number of nodes to join the superclusters along each line
18.    Select the combination of cells that produces the best connectivity
19. end for
```

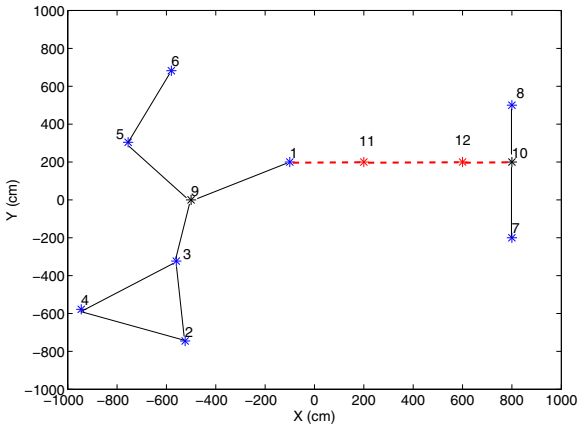


Fig. 6. Solutions adopted to join the superclusters SC_1 and SC_2 in the example in Fig. 4.

selected are marked with red asterisks as nodes 11 and 12 and the new connections are represented as dashed lines.

Finally, robots are commanded to deploy nodes at the repairing positions. New deployed nodes integrate with the existing WSN. If the BS node confirms that it receives packets from all nodes: the connectivity repairing is confirmed and the connectivity matrix *GCM* is updated. If the *BS* does not receive packets from all the nodes a new WSN diagnosis is needed.

6 WSN Redundancy Repairing

The redundancy repairing mechanism is activated if there are nodes with a lower N-redundancy than the desired level. This stage assumes that the network has 1-connectivity, i.e. it is applied after the connectivity repairing stage. This mechanism iteratively calculates the position where the deployment of a new node best improves the N-redundancy of the entire network. The iterations keep on until every node has at least the desired N-redundancy level.

The first step is to compute the redundancy of every node in the network. To do this we follow the concept of N-redundancy as stated in [1] and make use of the *GCM* obtained in the diagnosis mechanism and updated in the connectivity repairing mechanism. A node *i* has N-redundancy when if removed, at least N more nodes need to be removed to break the connectivity between two neighbors of *i*. Nodes with 0-redundancy are critical. A WSN is said to be robust if all nodes have a least 2-redundancy.

Algorithm 3 shows the method to compute the N-redundancy of node *i*. The algorithm can determine if node *i* has redundancy higher or equal to 2. If node *i* has redundancy lower than N, the method computes its redundancy. The algorithm operates with *GCM'*, a copy of *GCM* where the connections of the node *i* have been removed. If a pair of neighbors of *i* are disconnected in *GCM'*, redundancy of node *i* is set to 0. Otherwise, we analyze the impact of removing the connections of neighbors of node *i*. If removing the connections corresponding to any other neighbor of node *i* disconnects two neighbors of node *i* then redundancy of node *i* is 1. Otherwise, its redundancy is 2 or higher.

Algorithm 3. Computation of N-redundancy of node *i*

```
1.  Copy GCM in GCM'
2.  Initial Redundancy 2
3.  Remove all links with node i in GCM'
4.  Compute cluster matrix C' corresponding to GCM'
5.  Redundancy 0 if node i has only one neighbor
6.  for any possible pair of nodes j and k neighbors of i do
7.      if j and k are in different clusters then
8.          Redundancy 0
9.      else
10.         if Removing any other node, nodes j and k are in different clusters then
11.             Redundancy 1
12.         end if
13.     end if
14. end for
```

For example, node 7 in Fig. 7 has 0-redundancy because removing its connections disconnects nodes 3 and 8. On the other hand, node 3 has 1-redundancy: besides node 3 itself we need to remove one node, e.g. node 1, to disconnect two neighbors of node 3, e.g. 2 and 7. The redundancy of every node is shown on top of its identifier.

If the redundancy of a node is lower than the desired N-redundancy level, redundancy repairing mechanism is triggered, see Algorithm 4. Redundancy repairing iteratively computes the location in which to deploy a node that produces the larger increase in redundancy. A brute-force solution that simulates the redundancy increment when deploying a node at each scenario cell would involve high computational cost and result in bad scalability. Many cells are not within the radio coverage of any static node. Also, there are locations that even within the coverage range of a static nodes, are not useful to improve redundancy. Therefore, the proposed mechanism only analyzes the scenario cells that satisfy the following two conditions: a) the cell is within the convex hull of all the deployed nodes and b) the cell is within the radio coverage (distance is lower than R) of at least two static nodes.

The computation of the convex hull of a set of points can be easily performed with algorithms such as [25,20]. Figure 7 shows the convex hull that envelopes the nodes of a cluster. The edges of the polygon can be either connections between static nodes or imaginary lines between disconnected nodes (dashed lines). The N-redundancy of every node is shown on top of the node identifier.

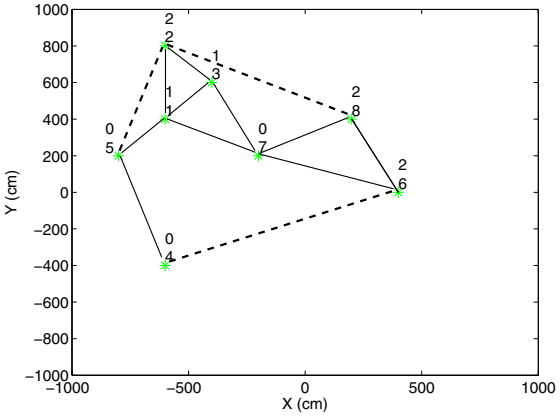


Fig. 7. Example of computation of N-redundancy and convex hull in a node cluster

The cells that do not satisfy both conditions are discarded. The redundancy repairing cells are selected analyzing the impact on redundancy improvement of deploying a node on that cell. The first criterion is the number of nodes that increase their redundancy from 0 to 1 (or higher) as a result of deploying a node at that cell. Nodes with 0-redundancy, such as node 7 in Fig. 7, are critical nodes in the WSN: the proposed method gives priority to solving fragile points. The second criterion is the number of nodes that increase their redundancy from 1 to 2 (or higher). The second criterion is used if there is a tie when using the first criterion. In case the tie keeps, the third criterion is the number of new connections originated by deploying a node at that cell. If necessary, the fourth criterion is to select the cell that obtains the highest average PRR in the connections it originates.

Once a repairing position is selected, *GCM* is updated with the connections produced by the new node. Then, the next iteration of the WSN redundancy repair mechanism starts. The iterations continue until all nodes have at least N-redundancy. Once they have been computed, the robots are commanded to deploy the nodes, provided that the needed number of new nodes are available.

Algorithm 4. N-redundancy repair algorithm

1. Compute the convex hull as in [25,20]
2. Select cells within the convex hull that connect at least two nodes
3. **while** N-redundancy not achieved **do**
4. Analyze the effect of deploying one node in every selected cell
5. Select the cell that produces best impact on redundancy
6. **end while**

Keeping with the example as in Fig. 7, Fig. 8 shows an example that requires two iterations to achieve the desired 2-redundancy in all the nodes. The deployment of the first node, see Fig. 8-left, increases N-redundancy from 0 to 1 or 2 in nodes 7, 4 and 5. N-redundancy is also increased from 1 to 2 in nodes 3 and 1. Hence, deploying only one new node, 1-redundancy is ensured in the whole network. The deployment of the second node, node 10, see Fig. 8-right, increases N-redundancy from 1 to 2 in nodes 9, 5, 7 and 8. Therefore, 2-redundancy is ensured in the whole WSN and the redundancy repairing mechanism finishes.

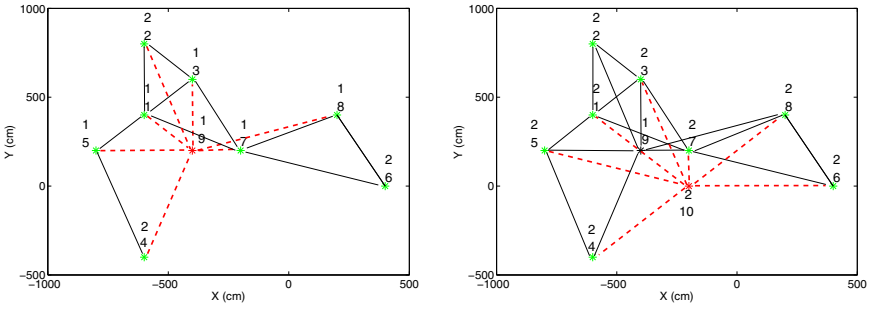


Fig. 8. First and second iterations of the WSN redundancy repairing mechanism in the example of Fig. 7. Left) First iteration. The deployment of node 9 ensures 1-redundancy in the WSN. Right) Second iteration. The deployment of node 10 ensures 2-redundancy in the WSN.

7 Validation

7.1 Simulations

The proposed method is compared with the optimal algorithm in order to assess its performance. The addressed problem is known be NP-complete, [13]. The algorithm

that minimizes the number of nodes consists in trying each possible solution. First, it tries to solve the problem using one node. If no solution is found, all combinations with two nodes are tried. It keeps until a solution is found. Figure 9 shows the computational time and the number of nodes required by both methods to solve the problem with 1-redundancy in 10 different 20x20 m^2 scenarios with 5 nodes. In these simulations both methods were executed in MATLAB. The time is represented in logarithmic scale. In all scenarios the solution found by our method is very near the optimal, it uses the same number of nodes or only one less, and requires times that are only a small fraction of the optimal. In many cases, e.g. scenarios 2, 4, 6, 7 and 8, our method gives the same solution but requiring significantly less time. In scenario 10, our method solves the problem adding one node in 1.88 s, while the optimal method solves it with 4 nodes but requires 221970 s.

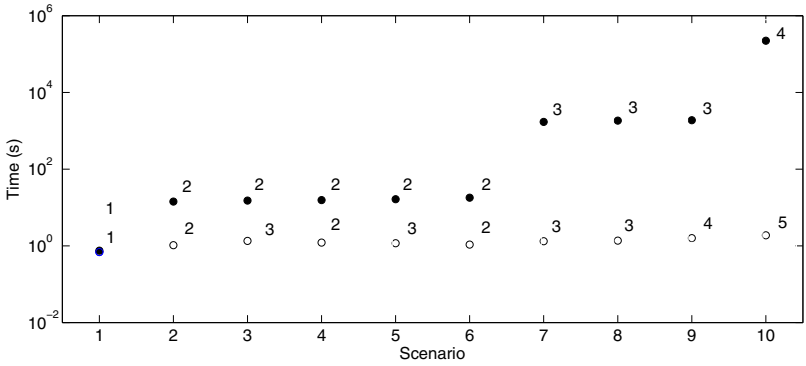


Fig. 9. Computational time and number of nodes required for achieving 1-redundancy in 10 different scenarios. The solutions of the optimal algorithm are represented by full markers while those of the proposed method are represented by empty markers. The number of nodes required to solve each scenario is shown next to every marker.

Figure 10 shows the computational times and number of nodes resulting from solving the problem in the scenario in Figure 4 but scaled in the range [0.5, 3.5]. In scaled scenarios, distances from nodes location to the scenario center were multiplied by a scaling factor. The nodes communication range were kept unaltered. The proposed method scales linearly only with the area of the convex hull that envelopes nodes, which is typically a fraction of the total area.

The proposed method discretizes the scenario into cells. Its dependence with the coarseness of the discretization is also analyzed. Low resolution leads to solutions that use more nodes than needed. High resolution leads to high computing burden. To consider both effects discretization coarseness is selected depending on the nodes range and the size of the convex hull. Notice that the convex hull size is unknown before the redundancy step. The diagnosis and connectivity steps are performed using a fixed resolution, e.g. 1x1 m per cell. Notice that connectivity is very efficient and its burden has low dependence with cell resolution.

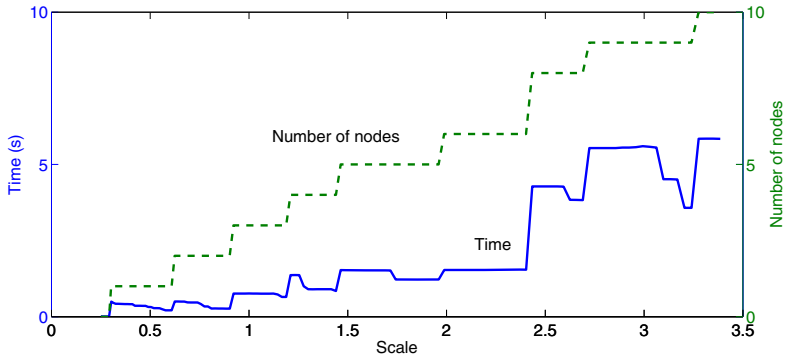


Fig. 10. Computational time and number of additional nodes needed to achieve 1-redundancy in the scenario in Figure 4 at different scales.

Then, once the nodes range and the convex hull size have been computed, the resolution is selected in the range [*R*/2, *R*/5] depending on the convex hull size. The objective is that the convex hull has around 150 cells. Of course, larger scenarios could produce more than 150 cells even if R/2 resolution is selected. In these cases, the method scalability ensures reaching a solution in reasonable times, as analyzed below.

The scenario in Fig. 4 was solved using a discretization in the range [25-625] cells. Figure 11 shows the computational times employed by the proposed algorithm to achieve 1-connectivity, 1-redundancy and 2-redundancy. The computational times in the three cases behave linearly against the number of cells in the discretized scenario. The times to compute 1-connectivity solutions are negligible even in highly discretized scenarios. The times to compute 1-redundancy solutions are lower than 1 second. The times to compute 2-redundancy solutions are larger and they increase linearly with the number of cells.

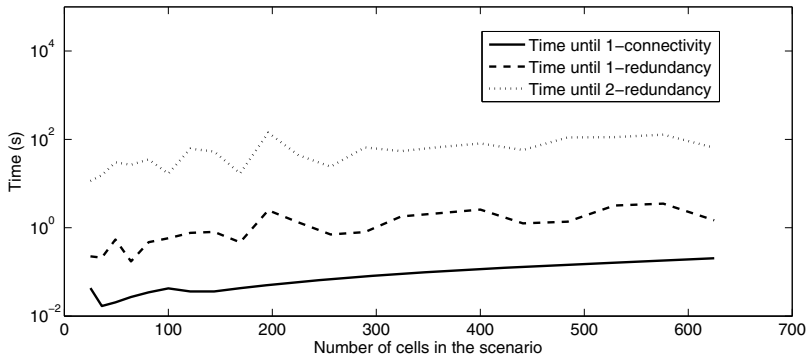


Fig. 11. Computational times to solve the scenario of Figure 4 with a growing number of cells

7.2 Experiments

The proposed method has been also experimented in real settings in the *CONET Robot-WSN Integrated Testbed*. The *CONET Testbed*[1] is a remote open tool to assess and compare multi-robot and WSN methods and algorithms. It is composed by 5 Pioneer 3-AT robots and 140 WSN nodes of different models. Each robot is endowed with a Hokuyo range finder and an Microsoft Kinect camera, GPS and Inertial Measurement Unit, among others. The testbed is installed since 2010 at the basement of the building of the School of Engineering of Seville (Spain). For this experiment one robot was equipped with a robotic arm for node deployment and retrieving, see Fig. 12-left.

In these experiments twenty nodes and a Base node have been deployed in the testbed room. The fleet of Pioneer 3-AT robots cooperatively perform the WSN diagnosis mechanism to compute: locations of the nodes, the connectivity *GCM* and cluster *C* matrices and the PRR-range model. Figure 12-right shows the WSN topology in one experiment after the WSN diagnosis stage. The circles represent the discovered nodes at their estimated locations. The connectivity between nodes is also depicted. In this example, the WSN is fragmented into three clusters: the biggest cluster, $C1$, includes 10 nodes and the Base node, the other two, $C2$ and $C3$, are disconnected from the Base.

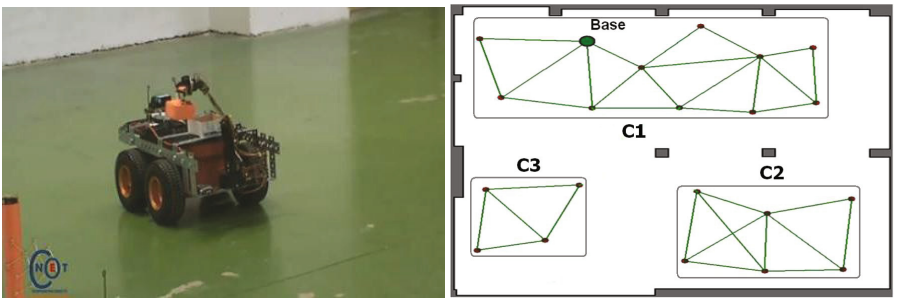


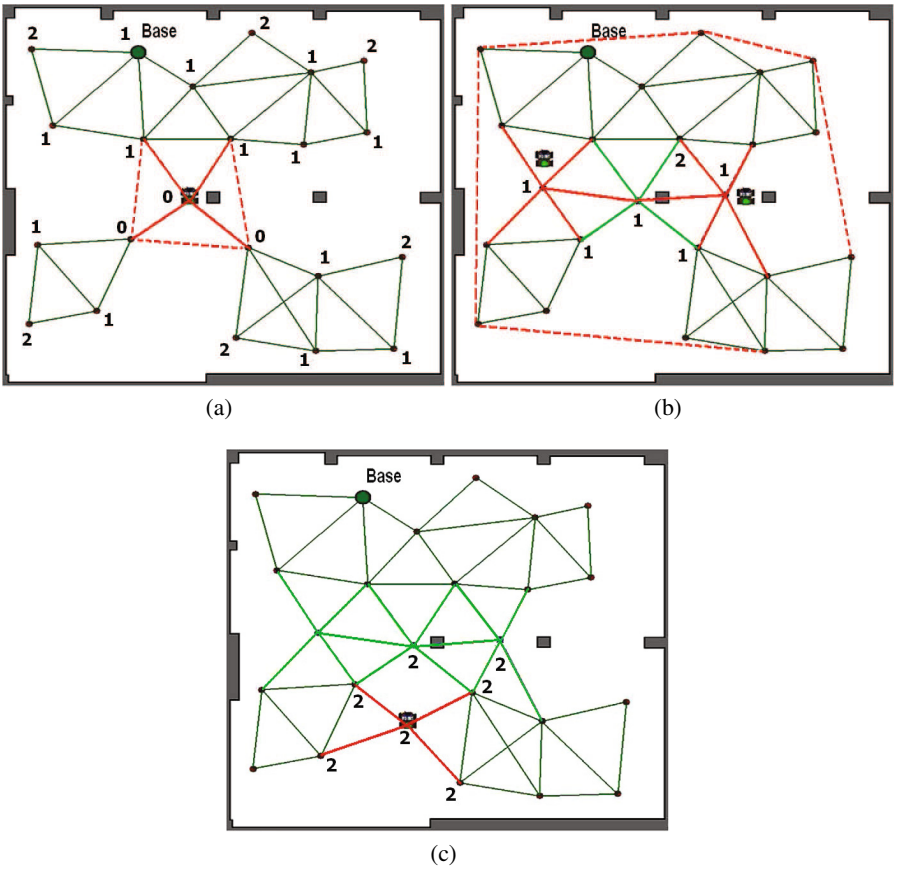
Fig. 12. (Left) Mobile robot deploying a node with its robotic arm. (Right) Connectivity after the WSN diagnosis stage. The WSN is fragmented in three clusters.

Since the network is fragmented the WSN connectivity repairing mechanism is started. It is detected that the distance between the closest nodes is lower than $2R$. Thus, the three clusters can be joined in one supercluster. The method determines that one single node deployed between the three clusters can join them in only one supercluster. Figure 13-a shows with dashed lines the links between the closest nodes of the three clusters. In this case the links between $C1$ and $C2$, between $C2$ and $C3$, between $C3$ and $C1$ form a four sided polygon together with already existing connection between the nodes of $C1$. The solution is sought inside the area delimited by the polygon. The best solution is to deploy one node at the center of the polygon. Then, a robot of the fleet is assigned with the task of deploying a node at the computed position. A simple task allocation based on Market-based auctioning is used, [7]. Once deployed, the new node establishes links with its new

[1] `https://conet.us.es`

neighbors, depicted in red solid lines in Fig. 13-a. The WSN has *1-connectivity* and the connectivity repairing stage is considered finished.

The initial redundancy of every node is shown in Fig. 13-a. There are three nodes with 0-redundancy and thus the WSN redundancy repairing mechanism is triggered. Figure 13-b shows the convex hull that envelopes all the nodes. The edges of the polygon are connections between static nodes or imaginary lines between nodes with no radio connection, shown as dashed red lines in the figure. In this case all the cells inside the convex hull are within the radio coverage of at least two nodes, so in every iteration the WSN redundancy repairing mechanism computes a solution within the convex hull.



(a) (b)

(c)

Fig. 13. Steps in the WSN connectivity and redundancy repairing mechanisms. (a) Network after the WSN connectivity repairing. One node is deployed. The new connections are shown as red lines. (b) Network after achieving 1-redundancy in the WSN redundancy repairing stage, the new connections after deployment of two nodes are shown in red colour. (c) Network state after the deployment of one more node.

After two iterations and the deployment of two more nodes 1-redundancy is achieved in the whole network. After deployment, see Fig. 13-b, ten -10- new connections are created -in red in the figure- and all nodes have at least 1-redundancy. Then, another iteration of the redundancy repairing mechanism was carried out to achieve 2-redundancy. The location selected was computed as described in Section 6. After deployment, four new connections were created and a total of seven WSN nodes increase their N-redundancy from 1 to 2. A summary of this experiment can be seen in this video[2].

8 Conclusions

In this chapter a system for connectivity and redundancy repairing of WSN with mobile robots has been presented. The system consists of three mechanisms: diagnosis, connectivity repairing and redundancy repairing. After the diagnosis stage, the connectivity repairing mechanism finds the optimal deployment locations to ensure that the WSN has 1-connectivity. The redundancy repairing mechanism is executed iteratively. At each iteration, it finds the location where the deployment of a new node best improves N-redundancy of all the nodes in the WSN.

The proposed scheme does not use any parameters. It acquires all the required information during WSN diagnosis stage and only needs as input the desired level of N-redundancy. It has been successfully validated in experiments carried out in the *CONET Robot-WSN Integrated Testbed*[3].

The method uses a iterative approach. It assumes that all necessary nodes are available for deployment. However, when it is not the case, the method uses the nodes available in the best possible way. Each node is deployed optimizing its impact following a step-by-step optimization. The use of globally optimal methods is currently under research. On the other hand, the problems of connectivity and redundancy are coupled but in our method are treated independently. The search for a global solution that simultaneously solves the two problems is under development.

References

1. Atay, N., Bayazit, B.: Mobile wireless sensor network connectivity repair with k-redundancy. In: Chirikjian, G.S., Choset, H., Morales, M., Murphey, T. (eds.) Algorithmic Foundation of Robotics VIII. STAR, vol. 57, pp. 35–49. Springer, Heidelberg (2009)
2. Bourdenas, T., Sloman, M.: Towards self-healing in wireless sensor networks. In: IEEE Sixth Intl. Workshop on Wearable and Implantable Body Sensor Networks, pp. 15–20 (2009)
3. Caballero, F., Merino, L., Gil, P., Maza, I., Ollero, A.: A probabilistic framework for entire WSN localization using a mobile robot. Robotics and Autonomous Systems 56(10), 798–806 (2008)
4. Coles, M., Azzi, D., Haynes, B.: A self-healing mobile wireless sensor network using predictive reasoning. Sensor Review, Emerald Group 28(4), 326–333 (2008)
5. Corbett, D.R., Gage, D.W., Hackett, D.D.: Robotic communications and surveillance-the DARPA LANdroids program. In: Wang, D., Reynolds, M. (eds.) AI 2011. LNCS, vol. 7106, pp. 749–758. Springer, Heidelberg (2011)

[2] `http://www.youtube.com/watch?v=1rKjdT-dCjc`
[3] `https://conet.us.es`

6. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Deployment and connectivity repair of a sensor net with a flying robot. In: Experimental Robotics, I. (ed.) Experimental Robotics IX. STAR, vol. 21, pp. 333–343. Springer, Heidelberg (2006)
7. Dias, M.B., Stenz, A.: Opportunistic optimization for market-based multirobot control. In: Proceedings IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 2714–2720. Lausanne, Switzerland (2002)
8. Fagiolini, A., Babboni, F., Bicchi, A.: Dynamic distributed intrusion detection for secure multi-robot systems. In: IEEE Intl. Conf. on Robotics and Automation, ICRA 2009, pp. 2723–2728 (2009)
9. Ganeriwal, S., Kansal, A., Srivastava, M.: Self aware actuation for fault repair in sensor networks. In: Proceedings of the 2004 IEEE Intl. Conf. on Robotics and Automation, ICRA 2004, vol. 5, pp. 5244–5249 (2004)
10. Gui, C., Mohapatra, P.: SHORT: self-healing and optimizing routing techniques for mobile ad hoc networks. In: Proceedings of the 4th ACM Intl. Symposium on Mobile ad Hoc Networking & Computing, pp. 279–290. ACM (2003)
11. Honkavirta, V., Perala, T., Ali-Loytty, S., Piché, R.: A comparative survey of wlan location Fingerprinting methods. In: 6th Workshop on Positioning, Navigation and Communication, pp. 243–251 (2009)
12. Jiménez-González, A., Martínez-de Dios, J.R., Ollero, A.: An Integrated Testbed for Cooperative Perception with Heterogeneous Mobile and Static Sensors. Sensors 11, 11516–11543 (2011)
13. Lin, G., Xue, G.: Steiner tree problem with minimum number of Steiner points and bounded edge-length. Information Processing Letters 69(2), 53–57 (1999)
14. Liu, C., Wu, K., He, T.: Sensor localization with ring overlapping based on comparison of received signal strength indicator. In: Intl. Conf. on Mobile Ad-Hoc and Sensor Systems, pp. 516–518 (2004)
15. Luthy, K., Grant, E., Henderson, T.: Leveraging rssi for robotic repair of disconnected wireless sensor networks. In: IEEE International Conference on Robotics and Automation, pp. 3659–3664 (2007)
16. Ma, H., Liu, Y.: On coverage problems of directional sensor networks. In: Jia, X., Wu, J., He, Y. (eds.) MSN 2005. LNCS, vol. 3794, pp. 721–731. Springer, Heidelberg (2005)
17. Machado, R., Tekinay, S.: Bounds on the error in estimating redundancy in randomly deployed wireless sensor networks. In: IEEE Intl. Conf. on Sensor Technologies and Applications, pp. 319–324 (2007)
18. Machado, R., Ansari, N., Wang, G., Tekinay, S.: Adaptive density control in heterogeneous wireless sensor networks with and without power management. IET Communications 4(7), 758–767 (2010)
19. Martinez-deDios, J., Lferd, K., de San Bernabé, A., Núñez, G., Torres-González, A., Ollero, A.: Cooperation between UAS and wireless sensor networks for efficient data collection in large environments. Journal of Intelligent and Robotic Systems, 1–18 (2012)
20. Pateiro-López, B., Rodríguez-Casal, A.: Generalizing the convex hull of a sample: The R package alphahull. Journal of Statistical Software 34(5), 1–28 (2010)
21. Popa, D.O., Lewis, F.L.: Algorithms for Robotic Deployment of WSN in Adaptive Sampling Applications. In: Li, Y., Thai, M.T., Wu, W. (eds.) Signals and Communication Technology, pp. 35–64. Springer US (2008)
22. Sitanayah, L., Brown, K.N., Sreenan, C.J.: Fault-Tolerant relay deployment based on length-constrained connectivity and rerouting centrality in wireless sensor networks. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 115–130. Springer, Heidelberg (2012)
23. Watfa, M.K., Commuri, S.: Boundary coverage and coverage boundary problems in wireless sensor networks. International Journal of Sensor Networks 2(3-4), 273–283 (2007)

24. Wang, X., Bischoff, O., Laur, R., Paul, S.: Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications. Procedia Chemistry 1, 461–464 (2009)
25. Wu, W., Li, L., Wang, J.: An improved Graham algorithm for determining the convex hull of planar points set. Science of Surveying and Mapping 35(6), 123–125 (2010)
26. Yoo, G., Jung, J., Lee, E.: Fault Management for Self-Healing in Ubiquitous Sensor Network. In: IEEE Second International Conference on Future Generation Communication and Networking Symposia, FGCNS 2008, vol. 5, pp. 21–25 (2008)
27. Younis, M., Lee, S., Guptam, S., Fisher, K.: A localized self-healing algorithm for networks of moveable sensor nodes. In: IEEE Global Telecommunications Conference, pp. 1–5 (2008)
28. Zhang, C., Zhang, Y., Fang, Y.: Localized algorithms for coverage boundary detection in wireless sensor networks. Wireless Networks 15(1) (January 2009)

Fuzzy Logic Control for Mobility Support in Industrial Wireless Sensor Networks

Zinon Zinonos[1], Chrysostomos Chrysostomou[2], and Vasos Vassiliou[1]

[1] Department of Computer Science,
University of Cyprus, 1 University Ave., 2019 Nicosia, Cyprus
[2] Department of Computer Science and Engineering,
Frederick University, 7, Y. Frederickou Str. Pallouriotissa, 1036 Nicosia, Cyprus

Abstract. Mobility management is a crucial problem for wireless mobile communication, especially in wireless sensor networks (WSN). In this chapter, we show the need of providing an intelligent mobility controller, applicable to any WSN industrial environment or testbed setting with mobility requirements. In particular, we utilize fuzzy logic control, due to its reported strength in controlling nonlinear systems using linguistic information, to build an efficient mobility controller that aid sensor mobile entities to decide whether they have to trigger the handoff procedure and perform the handoff to a new connection position or not. Based on real industrial setting experiments, the fuzzy logic-based mobility controller has shown significant benefits compared to the RSSI-based conventional mobility solution, fulfilling basic performance requirements.

1 Introduction

With the advancements in wireless communications and with the rapid growth in the number of mobile entities, mobility management is one of the most important and challenging problems for wireless mobile communication. Mobility management deals with all actions that must be taken in a network to support the movement of mobile users without losing connectivity. This is true both in infrastructure-based technologies (cellular, WLAN) and infrastructure-less types (ad-hoc, vehicular, sensor). In all cases when a mobile user/node moves to a new location it has to establish a new radio link with the target base-station/access-point/neighbour and release the connection with the previous, in a process called handoff. A basic handoff process consists of two main phases: (a) measurement phase, dealing with the mechanics of measuring important parameters and (b) the decision/execution phase, dealing with the algorithm parameters and handoff criteria [1].

While mobility management is a well understood and researched topic in cellular telephony and WLANs it is still a challenging topic in wireless sensor networks (WSN). WSNs promise fine-grain monitoring in a wide variety of environments and are expected to be deployed in difficult and often inaccessible environments, which from the communication perspective are also usually harsh. WSNs are expected, in many cases, to be densely deployed, with a large number of nodes

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*,
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_10,

within communication range, which exacerbates the communication problems. An emerging application area that combines all the issues mentioned above and in addition poses significant performance requirements on the networks and protocols is the use of WSNs in industrial and manufacturing settings, or in general in settings where critical information needs to be transmitted and critical applications need to be operating [2]. It is, therefore, not difficult to envision scenarios within such demanding applications and settings where mobility would also be required [3],[4].

Supporting mobile users in an industrial environment is something that the existing industrial standards like WirelessHart [5] and ISA100 [6] do not give special attention to. WirelessHART and ISA100.11a use a centralized network management approach for communication scheduling and managing routes. Despite the advantages of such approach when the network topology and application requirements are static and heavily pre-configured, it is not certain how these standards perform in dynamic situations involving node mobility. The inability to properly handle mobility may result in problems, including increased packet loss, delayed data delivery, and increased downtime, all of which increase the overall energy consumption.

Mobility support in this work has been mainly motivated by the need to monitor the health and status of mobile workers in industrial settings. There are many hazardous activities in an industrial plant that need to be monitored for safety. One such activity is the cleaning and condition assessment of storage tanks in an oil refinery. Tanks are very hazardous environments and typically contain a toxic atmosphere and residues of their previous contents. When employees enter such hazardous areas there is a possibility to loose consciousness. By using body orientation/tilt and heart or pressure monitoring sensors attached to employees, their condition can be monitored and alarms can be signalled when an emergency occurs. Surrounding the tank that is being cleaned are usual sensors deployed for other scenarios, e.g. production monitoring. As the mobile worker moves around the tank, body orientation/tilt measurements are sent from the sensor to the sink forwarded by intermediate nodes. Data may be sent via different intermediate nodes (attachment points) based on the location of the mobile worker. In order to continuously receive information from the mobile workers a mobility management technique must be implemented so as to enable the handoff between different access points, while at the same time maintaining some performance guarantees for the critical application. In addition, it is not difficult to envision scenarios where mobile robots are used in hazardous areas to perform several monitoring tasks.

The overall system was implemented and evaluated in the context of the EU-funded GINSENG project [7]. The end user of the project was the company operating the Petrogal oil refinery at Sines, Portugal. The Petrogal refinery is a complex industrial facility, which includes a wide range of processing units that need careful monitoring and control of critical operations. Currently, the refinery is completely automated, but totally wired based. Upgrades to the current wired system are impossible to perform in order to support mobile users. Therefore, a

real WSN has been deployed in the refinery, targeting several specific scenarios including the monitoring of mobile workers (personnel safety scenario).

The contributions of this work are the following:

- an intelligent controller, based on fuzzy logic is proposed. This controller enables sensor MN to decide intelligently whether they have to trigger the handoff procedure and perform the handoff to a new position or not.
- a real industrial setting (oil refinery) is used as the evaluation environment, something that poses new challenges regarding the design of mobility support.

The rest of this work is organized as follows: Section 2 describes related work regarding the mobility management, Section 3 provides experimentation scenario and the system architecture in general, Section 4 describes the handoff procedure using the RSSI metric, where in Section 5 we propose the fuzzy logic-based mobility management solution. Finally, Section 6 contains an evaluation of the proposed solution, and, Section 7 provides the conclusions stemming from this work.

2 Related Work

2.1 Single Metric-Based Mobility Management

In general, the handoff procedure is initiated by a triggering decision. The handoff triggering is usually based on parameters like the Received Signal Strength Indicator (RSSI) or the Link Quality Indicator (LQI). The main issue of using these metrics is the unpredictable behaviour and the rapid fluctuations of the wireless medium. In the literature, several approaches have been proposed that make use of RSSI either as a single decision method or supported by a threshold and hysteresis margin methods [8],[9],[10]. Usually the selection of the RSSI threshold is based on a targeted Packet Reception Rate (PRR).

The most commonly used triggering/handoff criteria are the following:

- *Better Signal Strength*: the MN selects the attachment point with the strongest RSSI. It can be considered as being a simple solution, but it can cause too many unnecessary handoffs. In case of sensor networks, it will increase the energy consumption since the MN must be always on (it is always triggered) for hearing for new attachment points.
- *Threshold:* if the current signal strength is less than the threshold the handoff is triggered. In case that a new attachment point with strongest RSSI is available the MN will handoff. The issue with this metric is the threshold value selection since low threshold may lead to late handoff where high threshold to early handoff.
- *Better Signal Strength with hysteresis:* the MN selects the attachment point with sufficiently stronger (by a hysteresis margin, h) RSSI compared to the one of the serving attachment point. Using this technique, the ping-pong phenomenon can be avoided. However, there may be the case where the

handoff decision that occurs could be unnecessary since the serving attachment point signal may be strong enough to maintain the connectivity. In case of sensor networks, energy consumption is increased.
- *Threshold with hysteresis:* if the current signal strength is less than the threshold and a new attachment point with sufficiently stronger (by a hysteresis margin, h)RSSI is available, then the MN will handoff. Using this technique, the ping-pong phenomenon can be avoided.

The importance of the RSSI metric as a quality indicator was argued in [11] where the authors have shown that generally for RSSI values greater than -87dBm the resulting PRR is at least 85% indicating a very good link. In addition, they have shown that RSSI is a promising indicator when its value is above the sensitivity threshold of the radio communication chips (in their case the CC2420 chip). Finally, they concluded that protocol designers looking for inexpensive and agile link estimators may choose RSSI over the LQI.

In [12] the authors measured the wireless link burstiness and they concluded that if the mean received signal strength (RSS) is above -80dBm then the link is almost always good. An exception to this value occurs when people were actively moving between the nodes, in which case there is a grey region of good, intermediate, and poor links slightly below the identified -80dBm threshold.

In [13] the authors performed a set of experiments to get a better understanding of key parameters, namely, the lower link quality threshold level and the hysteresis margin. They concluded that the network perform best when the lower link quality threshold is equal to -90dBm and the hysteresis margin is equal to 5dBm.

In [14] the authors proposed a handoff scheme in which a mobile node constantly monitors the received power from its cluster-head and it triggers a handoff decision when the RSSI drops below a power threshold of -75dBm. Authors justified this value based on previous studies which found that such threshold can guarantee packet reception ratios above 95% ([15], [16]).

In [17], the authors have the mobile node sending periodic probe messages to its current access point and expecting some acknowledgement messages. They measured the RSSI average based of the acknowledgement messages received and if this value is lower than a predefined threshold the MN initiates the handoff procedure. Using this approach, they managed to reduce unnecessary handoffs.

Based on the aforementioned related work the RSSI threshold value varies from -90dBm to -75dBm depending on the evaluation environment and on the targeted PRR.

2.2 Fuzzy Logic-Based Mobility Management

Recently, some work has come to light that propose the use of heuristic models, like fuzzy logic, to support the handoff triggering decision. In [17] authors provided a fuzzy logic system to support the mobility procedure based on RSSI level, velocity of mobile node, number of hops to sink node, and some other metrics such as traffic load, energy level, and link quality value. Although they

discussed in detail their solution they did not provide any implementation or evaluation of it. Therefore, the applicability of their solution and possible overheads are undetermined. In addition, the high number of metrics that they aim to use will lead to an increased fuzzy logic complexity since a big number of rules must be enabled at any time. Due to the limited capabilities of the sensor nodes a fuzzy logic based system must be as simple as possible.

Several works using fuzzy logic techniques have appeared in the field of mobility management, with the majority targeting the support of vertical handoffs. In [18], a handoff decision for heterogeneous networks is identified as a fuzzy multiple attribute decision-making problem and fuzzy logic is applied to deal with the imprecise information. In [19], a handover algorithm is proposed to support vertical handoffs between heterogeneous networks. This is achieved by incorporating the mobile IP principles in combination with fuzzy logic concepts utilizing different handoff parameters. Furthermore, in [20], the authors deal with a vertical handover decision algorithm based on the fuzzy control theory. The algorithm takes into consider the factors of power level, cost, and bandwidth in order to decide about the vertical handover. In [21] [22], the authors proposed and implemented a Fuzzy-Based Handover System (FBHS), where they showed that the proposed system had a good behaviour for handover enforcement, but in some cases could not avoid the ping-pong effect.

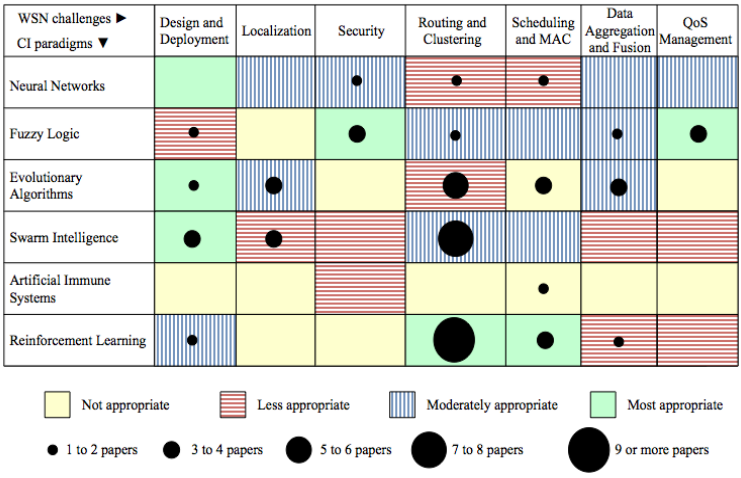


Fig. 1. WSN challenges and CI paradigms [23]

In order to support the complex situations of mobility management such as the triggering procedures, mobility management solutions can use tools from the family of Computational Intelligence (CI). In [24], CI is defined as *the computational models and tools of intelligence capable of inputting raw numerical sensory data directly, processing them by exploiting the representational parallelism and pipelining the problem, generating reliable and timely responses and withstanding high fault tolerance.* Several examples of application of such CI tools were

presented in the literature but whose prime focus is not WSNs. Recently, researchers started thinking of ways to use CI tools in order to solve WSN issues such as design and deployment, localization, security, routing, data aggregation and QoS management. Examples of such work are presented in [25], [26] and [27]. An overview of the CI techniques in WSNs are presented in [23], where authors findings have been summarized in Figure 1.

It is obvious that several parameters can affect the triggering and the handoff procedures especially when the targeted environment is an industrial field. Therefore, techniques that will distinguish triggering and handoff procedures and also combine available information in order to produce a successful necessary handoff are required.

Our approach to provide mobility support for mobile nodes resides on the fact that we have to control the handoff procedure, which means that at a first stage we have to control the handoff triggering procedure. Due to the unpredictability of the environment, we cannot rely on a single specific metric and we target a solution that can combine information using more than one metric. Since our environment is expected to perform dynamically, we decided to use fuzzy logic to control the triggering procedure. The selection of fuzzy logic is supported by the fact that it can handle multiple inputs with minimum overhead. In addition, the selection of fuzzy logic system was based on its simplicity and the fact that since it processes experts-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. Therefore, an intelligent controller is used, based on fuzzy logic, in order to help mobile sensor nodes to decide whether they have to handoff to a new position or not.

Analysis of the state of the art in this area reveals that there are a whole raft of projects and initiatives covering a wide spectrum of related research challenges, technological problems and collaboration activities in Mobile Wireless Sensor Networks. However, the motivation of this work is the fact that there is no protocol designed and evaluated to support the mobility process in critical environments; thus, this work provides effective solution to this missing piece. This issue is considered of the utmost importance for today's real-world industrial applications.

3 System Setup

This section describes the experimentation scenario with its requirements and the system architecture used to evaluate the proposed solution.

3.1 Experimentation Scenario

Mobility support in this work has been mainly related to monitoring mobile workers in support of the refinery Personnel Monitoring scenario. Figure 2 depicts this application scenario. A worker is tasked with cleaning a storage tank, located in one of the production lines. Surrounding the tank that is being cleaned

Fig. 2. Mobility Scenario

are usual sensors deployed for other applications, e.g. production monitoring. As the mobile worker moves around the tank, orientation messages are sent from his/her monitoring sensor to the sink, forwarded by intermediate nodes. Orientation is sampled at a frequency of 0.2Hz.

In order to continuously receive information from the mobile workers a mobility management technique must be implemented so as to enable the handoff between different access points. For example, based on Figure 2, we have three possible receiver nodes (indicated by the numbers 1, 2 and 3). The mobile worker at the beginning of his/her trip is attached to the receiver node 1. When the mobile worker is near to receiver node 2, the communication link with receiver node 1 is still good; therefore, there is no need to handoff. But as the mobile node gets far away from receiver node 1, it has to handoff to a new attachment point. Possible new attachment points are receiver nodes 2 and 3, but based on the communication quality the mobile worker may prefer to connect with receiver node 3.

Table 1 summarizes the mobility requirements for the Personnel Safety scenario. In terms of the plant network, mobile workers are temporary objects that only exist for a short period of time (time it takes to complete a specific job). Information on their state must arrive at the control center within few seconds. Although packet losses should be minimized, this application can be tolerant to a small amount of loss. Based on [7] and [28], the achieved reliability was 99% using fixed nodes with pre-deployed antennas. In our case, we set this requirement to 97% since we expect higher losses due to the mobility of the nodes. The number of mobile nodes shown in Table 1 has a two-fold significance. First it means that each worker is expected to carry only 1 sensor node. Second it means that our testing has been done using 1 person, i.e 1 mobile sensor node. However this does not mean that the solution cannot support multiple nodes at the same time. Since the solution is distributed, the triggering and decision algorithms run independently at each mobile node.

Table 1. Personnel Safety Requirements/Assumptions

Requirements	Definition	Value
Delay	The Time bound of data delivery	Data should arrive at the in-field sink in 1 second.
Reliability	How important is data delivery	>97%
Mobility	Level of Mobility	Mobile Workers
Network Size	Maximum number of Nodes	30
Topology Classification	Type of Topology	Tree
Hop Count	Max Number of Hops nodes can reside from the sink	4
Time Critical Traffic Direction	What direction are the time-critical flows in?	Upstream
Non-Time Critical Traffic Direction	What direction are the non time-critical flows in?	Downstream
Traffic Frequency	How often does each node generate a packet	> 1 seconds
Traffic Delay Bound	Time bound of the time-critical traffic	Upstream 1 seconds
Free positions	The total number of free positions in the tree topology	$> 2\,x\,number\,of\,MNs$
Number of mobile nodes	The total number of MNs in the network	1

3.2 System Architecture

As already mentioned, the work in this chapter makes use of the GINSENG framework [29] and knowledge gained from its deployment. The aim of the GINSENG project was to assure controlled performance and to achieve high reliability in a wireless sensor network operating in critical environments. The environment used as the development and testing ground was an oil refinery in Portugal.

The main characteristics of the architecture are:

1. Use of a TDMA-based MAC [30] protocol. Time is divided into epochs where each epoch has a predefined number of slots. Every node is assigned specific slots to transmit and receive packets. A number of slots is also assigned to each node (at the beginning of each epoch) for processing purposes.
2. The network uses multi-hop communication through a tree-based topology (Figure 3). The tree consists of H layers, where H is equal to the number of hops from the sink. A reasonable small number of nodes (N < 30) is used where N is directly proportional to the required communications delay bound; the smaller the required delay, the smaller the N.

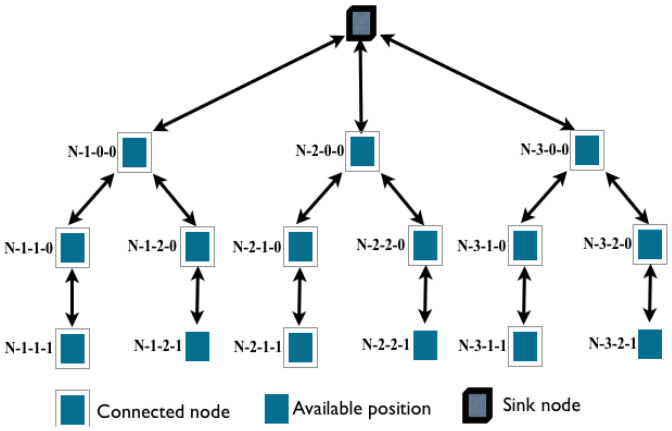


Fig. 3. 3-2-1 Tree Topology

3. Use of Dynamic Topology Control (DTC) [31] techniques. Using DTC each node is attached to the best available tree position during the construction of the network topology. DTC is responsible for Neighbour Discovery, joining and leaving the tree, re-attachment (as in the case of mobility) to the tree, and maintenance of the topology in case of faults.
4. The network is made up of resource constrained embedded systems where the majority of the nodes are deployed in fixed and predetermined positions.
5. The majority of nodes are static with no mobility; however, after transient faults or periodic tree maintenance, even static nodes can assume a different logical tree position, thus appearing as if moving inside the tree.
6. Mobile nodes cannot communicate directly with sink nodes except in the case when they are directly connected (logically) to the sink. Thus, the data communication of mobile nodes with the sink is accomplished via the other sensor nodes.
7. Nodes report data frequently with relatively high rate (up to once per second) and data must reach the sink within a given time bound.

For the work in this chapter we adopt a tree topology with 16 positions (Figure 3), an epoch of 92 time slots and a time slot duration of 10 ms. These parameters ensure a maximum delay, for any packet communicated between the sending node and any other node in this network, to be 920 ms (i.e. sub-second delays). At the same time the protocol is provisioned with two retransmission slots per normal upstream slot, i.e. the network will tolerate burst link losses of up to two packets. Packets that do not get through in an epoch are dropped.

4 Handoff Control in Industrial WSNs Using RSSI

In our earlier work [32], [9], two separate handoff algorithms have been proposed and implemented using RSSI Threshold as the triggering metric. A hard-handoff

solution was proposed in [32] and the S-GINMOB soft-handoff solution was proposed in [9]. Their only difference is that in the first case the connection is released before even searching for a new attachment point, whereas in the second case the connection with the old attachment point is maintained until the handoff process is completed. The latter provides a seamless handoff with no packet losses during or due to the handoff procedure. The operation of the soft-handoff solution is shown in the following Algorithm 1:

Algorithm 1. S-GINMOB Algorithm

```
if attached then
  set the handoff_trigger
  case 1: RSSI is below threshold
  if handoff_trigger == TRUE then
    scan(idle_slots);
    search for better attachment point based on the RSSI value
    sendPkt(Join_Ctrl_Pkt);
    receivedPkt(join_ack)
    switch to new address
  end if
end if
```

An important parameter of the above solution is the selection of the proper RSSI threshold value. The evaluation of the threshold was performed in [9] where we found that this values is equal to -78dBm.

We performed a number of experiments using the refinery testbed in order to evaluate the S-GINMOB solution compared to the scenario where no mobility management is supported. The results are shown in Table 2. The RSSI Threshold approach can maintain the continuous connectivity of the MN, meaning that there is no downtime during the handoff procedure. It also proves that, in some cases, RSSI can be used as an indicator of the need for handoff, but in the majority of the tests the solution suffered from rapid signal fluctuations.

Using S-GINMOB we observe a large number of triggers, which in the majority of the cases are unnecessary, since no handoff occurs. However, compared with

Table 2. Mobility Solutions Comparison

Metric	No Handoff	S-GINMOB
Packet Loss [%]	9.05	8.11
Total Power Consumption [mW]	0.117	0.57
Reception Power Consumption [mW]	0.096	0.51
Transmission Power Consumption [mW]	0.016	0.057
Number of Triggers	0	108.5
Number of Handoffs	0	4.5

the scenario where no mobility is supported, the total packet loss in S-GINMOB was slightly reduced.

The S-GINMOB solution shows an increased overall consumption due to the operation of the handoff triggering solution. Specifically there is a significant increase in the reception power consumption by a factor of 5, due to the increased scanning duration.

These items point to the fact that a different definition of the triggering and/or decision parameters may provide better results.

The suitability of the RSSI in an industrial environment was argued and used in [11], using a moving average calculation of the RSSI (average of at least 100 measurements) in order to minimize the RSSI drop effect and to make the behaviour of the RSSI smoother. However, in the experiments performed in this work a moving average solution could not be directly applied, since the target user was mobile and there was no time for taking the average of sufficient number of measurements before initiating a handoff. Another drawback of the RSSI threshold solution is that it has an End-to-End packet loss rate that is very high by any standard. According to [9] and [33] acceptable values for End-to-End packet loss are between 1% and 3%.

The main drawback of the RSSI-based triggering solution is the fact that, the decision depends on a single metric value to decide whether it will initiate the handoff or not to a new position without considering any other system metric. Therefore, we need an improved solution that will use system capabilities and that will manage to fulfil all the targeted performance requirements. As a consequence, we decided to consider both, the average link loss metric along with RSSI in order to support the handoff procedure. In other words, to combine the two single metrics. The reason of selecting the link loss metric and the RSSI was the fact that both metrics are available at each MN. The distributed nature of the approach allows the system to adapt quickly to disturbances or changes within the network in real-time.

As a first step, we run some experiments in SINES testbed area to extract information regarding the relationship of the End-to-End losses, RSSI and Link losses. The reason of not using directly End-to-End packet loss was that this information is not available at each node but only in the end system (sink node). Therefore, we need to somehow "predict" the value of the End-to-End loss using some other metrics that are available to each node. The results of those experiments are shown in Figure 4. Based on Figure 4, we can conclude that indeed we can use a combination of RSSI and link loss metrics in order to "predict" the End-to-End losses and support the handoff triggering procedure.

Figure 4 shows that when the link loss is above 15% and the RSSI is less than -78dB, the End-to-End packet loss is increased. Another conclusion is that when the RSSI is good enough (greater than -60dB) and the link loss in up to 40%, the End-to-End packet loss is acceptable. This behaviour is due to the ability of the mobile node to retransmit the packets in case the communication link between MN and parent node is good. Therefore, low End-to-End packet loss can be achieved by minimizing the link loss and maximizing the RSSI. In order

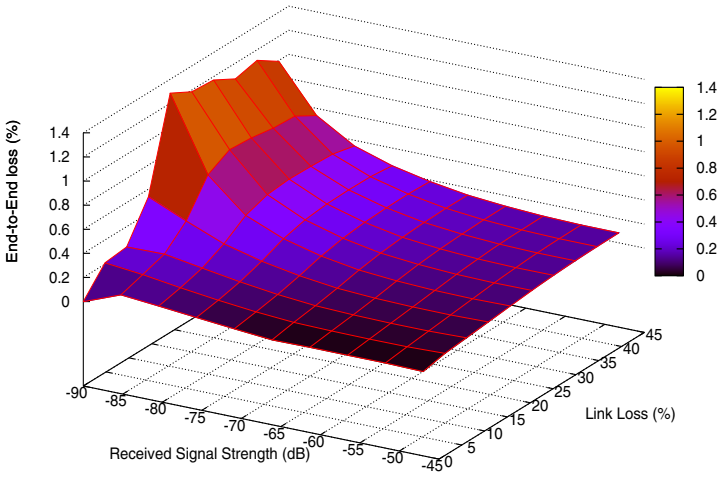


Fig. 4. End-to-End loss, Link Loss and RSSI relation

to exploit the above conclusions, we decided to use Fuzzy Logic techniques. We expect that Fuzzy logic will allow us to minimize the End-to-End packet loss, as it will combine RSSI and Link loss metric information, to minimize the total number of triggers, maximize the success ratio of triggers, handoffs and on-time triggering.

5 Proposed Intelligent Fuzzy Logic-Based Mobility Controller (FLMC)

Usually, a fuzzy controller design is based on empirical methods, a methodical approach to trial-and-error. This method is considered to be simple in terms of design and implementation. Thus, concerning the limitations that arise from sensor networks, this method seems to be a suitable approach for our system.

The general procedure that was followed consists of the following steps:

- Step 1: identify input and output variables.
- Step 2: determine fuzzy sets for the input and output linguistic variables.
- Step 3: choose the membership functions for the input and output fuzzy variables and derive the fuzzy control rules.
- Step 4: define inference engine.
- Step 5: choose the right defuzzification method.
- Step 6: If needed, trial and error approach is followed in order to tune the system.

Fuzzy Logic Control (FLC) has been applied successfully for controlling numerous systems in which analytical models are not easily obtainable or the model

itself, if available, is too complex and possibly highly non-linear (e.g. in communication networks). Therefore, FLC concentrates on attaining an intuitive understanding of the way to control the process, incorporating human reasoning in the control algorithm. It is independent of mathematical models of the system to be controlled. It achieves inherent robustness and reduces design complexity. This is in contrast with conventional control approaches that concentrate on constructing a controller with the aid of an analytical system model that in many cases is overly complex, uncertain, and sensitive to noise. The reason of selecting Fuzzy logic for our mobility scenario is twofold:

1. It can control non-linear systems (such is our system) that would be difficult or impossible to model mathematically.
2. Since FLC processes experts-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance.

A novel, intelligent controller, based on fuzzy logic, is proposed to be applied, in order to support the mobile workers scenario and to help mobile sensor nodes to decide whether they have to initiate the handoff to a new position or not. We use FLC principles to design a simple, effective and efficient non-linear control law, in order to offer inherent robustness with effective control of the system. Due to the mobility of the node and the resulting highly dynamic network environment, the proposed control mechanism needs to operate in a decentralized and self-organized way, i.e. locally at each mobile sensor node.

The system model of the proposed fuzzy logic based mobility controller (FLMC) is shown in Figure 5, where all quantities are considered at the discrete instant kT:

1. T is the sampling period.
2. RSSI(kT) is the signal strength indication, taken every sampling period.
3. LL(kT) is the link loss rate measured at each sampling period.
4. Pd(kT) is the calculated decision probability that triggers the handoff procedure
5. $SGi_{1,2}\,(kT)$ are the input scaling gains.
6. $P_{Threshold}$ is a predefined value that it is compared with the Pd.

A simple fuzzy inference engine (FIE) is designed to operate locally at each mobile sensor node, and control the handoff decision procedure, using linguistic rules that describe the behaviour of the environment in differing widely operating conditions. As shown in Figure 5, the FIE dynamically calculates the decision probability (to trigger the decision whether a mobile sensor node has to handoff to a new position or not), based on two network state inputs: the instantaneous value of the signal strength indication (RSSI), and the Link Loss rate, both taken at the end of each sampling period kT.

In fuzzy control theory, the range of values of inputs or outputs for a given controller is usually called the "universe of discourse". Often, for greater flexibility in fuzzy controller implementation, the universe of discourse for each process input is normalized by means of constant scaling factors [34]. For the fuzzy controller design developed here, the input scaling gains, $SGi_{1,2}\,(kT)$, are inherently

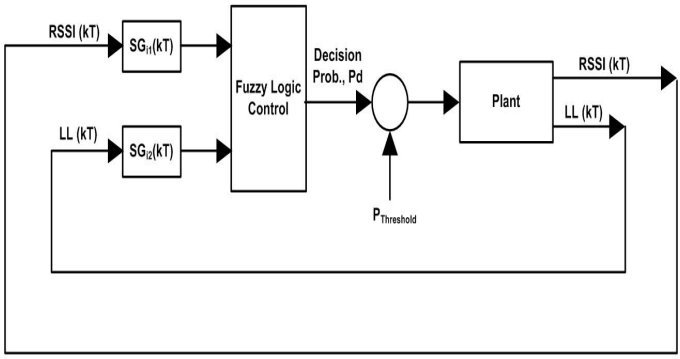


Fig. 5. Fuzzy Logic-based Mobility controller (FLMC)

chosen so that the range of values SG_{i1}(kT)RSSI(kT) and SGi_2(kT)LL(kT) lie in the real interval [0, 1] (see Eq. (1),(2)).

$$SGi_1(kT) = \frac{1 - \frac{RSSI_{min}}{RSSI(kT)}}{RSSI_{max} - RSSI_{min}} \tag{1}$$

$$SGi_2(kT) = \frac{1}{100} \tag{2}$$

where $RSSI_{min}$ and $RSSI_{max}$ were obtained during the experiments conducted in the setup phase of the oil refinery testbed.

The decision probability is calculated dynamically based on a non-linear control law derived by the construction of the FIE. Due to the high variability and dynamics of the system, a non-linear control law is more efficient to cope with these uncertainties and dynamics, in contrast with a linear control method.

The multi-input FIE uses linguistic rules that form the control knowledge rule base of the controller and describe how to best control the system, under differing operating conditions. Hence, linguistic expressions are needed for the inputs and the output, and the characteristics of the inputs and the output. Linguistic variables (that is, symbolic descriptions) are used to describe the fuzzy system inputs and output. The linguistic variables take on linguistic values that change dynamically over time and are used to describe specific characteristics of the variables; such values are generally descriptive terms such as “low”, “medium” and “high”.

The philosophy behind the knowledge base of the proposed scheme is that of being aggressive when the RSSI is low and the Link Loss is high, but on the other hand being able to smoothly respond in the case of adequate conditions in the environment. This point can be illustrated by observing the visualization of

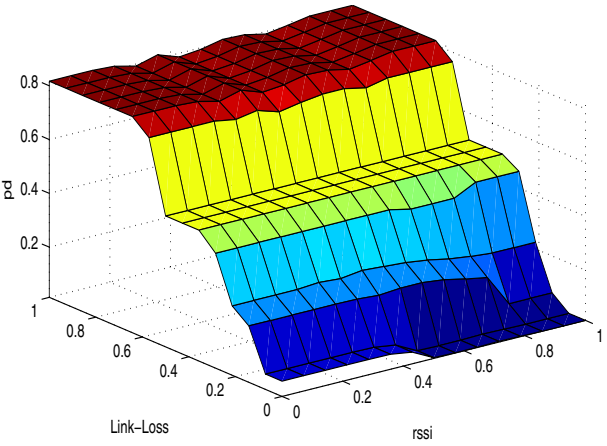


Fig. 6. FLMC Decision Surface

Table 3. FLMC Linguistic Rules - Rule Base

Decision Probability		Link Loss Rate			
		L[1]	***M***	***H***	***VH***
RSSI	***L***	LM	M	H	VH
	M	LM	M	H	VH
	H	L	M	H	VH
	VH	L	LM	H	VH

the control-decision surface of the FIE used in the proposed scheme (see Figure 6). It is shaped by the constructed rule base and the linguistic values of the input and output variables. A list of all possible "IF-THEN" control rules is shown in Table 3.

These rules reflect the particular view and experiences of the designer, and are easy to relate to human reasoning processes and gathered experiences. Usually, to define the linguistic values of a fuzzy variable, Gaussian, triangular, or trapezoidal shaped functions are used. Due to computational simplicity, trapezoidal and triangular shaped membership functions were selected in the proposed control scheme (see Figure 7, Figure 8 and Figure 9). In order to achieve the desired performance, the membership functions were defined based on the real data obtained from long-term testbed evaluation and based on the characteristics of the underlying system.

The amount of overlapping between the membership functions' areas is significant. The left and right half of the triangular membership functions for each linguistic value is chosen to provide membership overlap with adjacent membership functions. The chosen method is simple in that the sum of the grade

[1] low (L), low-medium (LM), medium (M), high (H), very high (VH)

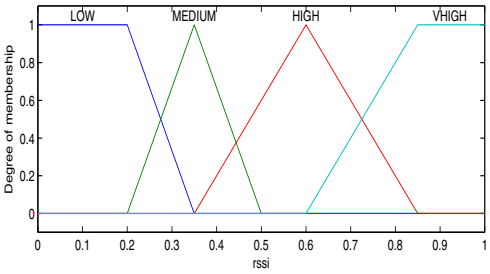


Fig. 7. RSSI Linguistic Input

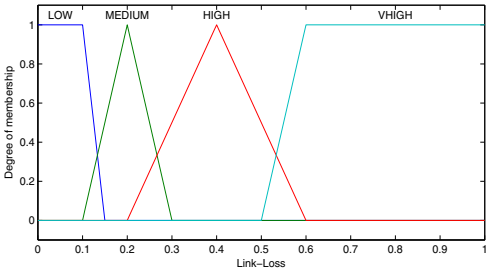


Fig. 8. Link Loss Linguistic Input

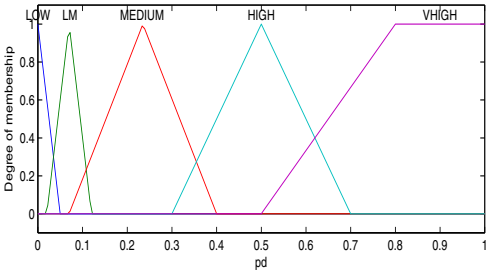


Fig. 9. Decision Probability Linguistic Output

of membership of an input value, concerning the linguistic values of the input variable, is always one (see Eq. (3)).

$$\sum_{k=1}^{m} \mu_k(x) = 1 \tag{3}$$

where $\mu_k(x)$ is the membership value of the input value x taken from the membership function of the linguistic value k, ($1 < k < m$, where m is the number of linguistic values of the linguistic variable), of the input variable of concern.

This results in having at most two membership functions overlapping, thus no more than four rules will be activated at any given time. This offers computational simplicity on the implementation of the proposed scheme, a design objective. Furthermore, there is no need for a fuzzy inference engine to be built in each mobile sensor node, thus saving on memory requirements. After the linguistic rules have been found and the linguistic values are tuned using a "trial and error" approach, the control surface is known and can be stored as a lookup table (size of $n \times n$) for selected sampling points requiring only a few kilobytes of memory in a fuzzy-capable mobile sensor node. In the system examined n is equal to 25, therefore the lookup table has 625 possible combinations of values. In that way, the memory and computation limitations of sensor networks are taken into account.

6 Performance Evaluation

This section presents the performance evaluation of the proposed FLMC system. At a first stage, we evaluate the selection of the $P_{Threshold}$ value and then we proceed with the evaluation of the proposed solution using the refinery testbed.

6.1 Evaluation of Threshold ($P_{Threshold}$)

The output of the fuzzy controller is a decision probability that, compared to a predefined threshold, indicates if the MN will initiate the handoff procedure or not. The fuzzy controller, as explained in Section 5, produces a decision probability value for different input parameters. These probabilities are stored in a lookup table. The mean decision probability value based on this table is equal to $P_{Threshold} = 0.23$. In this section, we will use different values for the threshold in order to identify the most appropriate threshold to use. We start using $P_{Threshold} = 0.23$ and then we increase/reduce it accordingly. We used the COOJA simulator [35] with the parameters that are shown in Table 4.

Table 4. Simulation Parameters

Simulation Time	2000 seconds
Testbed Size	45 x 35 meters
Transmission Range	25 meters
Number of simulations	100
Number of fixed/mobile nodes	13/1
Mobility model/Waypoint paths	Random Waypoint /10
Packet Rate	1 packet / 3 seconds
Mobile Node Speed	3 m/s with random stops

Figure 10 shows the End-to-End packet loss for different thresholds. We observe that for low thresholds the packet loss is minimized. This is due to the fact that when the threshold is low the MN will spend more time in scanning mode therefore there is a bigger probability to find a better attachment point.

Regarding the number of triggers, we see that the number of triggers is exponentially increased as the threshold is decreased. In addition, in Figure 10, we observe the average number of handoffs occurred during the evaluation of the different thresholds using 100 different simulation scenarios.

The small average number of handoffs is due to the fact that even if the handoff is triggered it does not mean that this will lead to a handoff, which means that the MN did not manage to find a better attachment point. Another explanation of the small number of handoffs is the fact that there are scenarios where the handoff was not triggered and therefore no handoff happened.

We conclude that the handoffs are proportional to the number of triggers, as was expected. Finally, concerning power consumption we observe that for the two low threshold values ($P_{Threshold} = 0.06$ and $P_{Threshold} = 0.16$) the power consumption is relatively higher compared to the other threshold values. Again, as in all other evaluations, the power consumption is proportional to the number of triggers and hence to the scanning period duration.

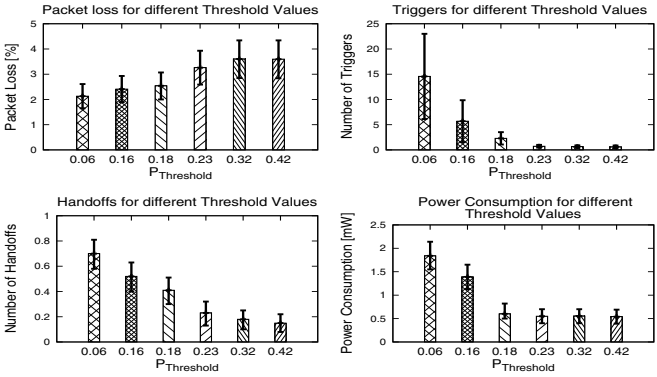


Fig. 10. Evaluation using different Threshold Values

Figure 11 shows the most important metric for the triggering options, the on-time triggering (percentage of successful triggers when End-to-End losses occur). As we see, the $P_{Threshold} = 0.06$, $P_{Threshold} = 0.16$ and $P_{Threshold} = 0.18$ accomplish on-time triggers compared to the other thresholds.

Based on the $P_{Threshold}$ evaluation, we conclude that the best performance is achieved using a $P_{Threshold} = 0.16$, which provides higher on-time triggering and low packet losses compared to any other threshold value. The issue of increased power consumption using this threshold could be solved by increasing the free available positions or by minimizing the scanning mode duration.

Thus, we proceeded with the evaluation of the FLMC, in the COOJA simulator, using a threshold value equal to 0.16. The results have shown that one more free position was enough to decrease the power consumption by 13%. The issue of dynamically adjusting the Threshold can be considered as part of our future work.

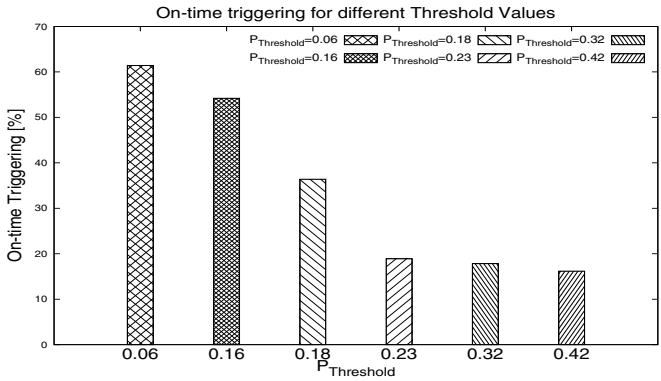


Fig. 11. On-time triggering using different Threshold Values

6.2 Experimental Evaluation of the Proposed FLMC

To evaluate the proposed FLMC algorithm, a number of on-site experiments were performed. The MN was introduced in the refinery testbed area and followed different random walks. The duration of the random walks were around 20 minutes. We use $P_{Threshold} = 0.16$ as the threshold value. The results shown are the average of ten different walks in the testbed area.

Figure 12 shows the operation of the FLMC in a representative experiment. The behaviour of the RSSI, Link Loss, and End-to-End Loss was captured so that to conclude if FLMC managed to decrease the packet losses after the triggering was initiated.

Based on Figure 12, two handoff events happened during the experiment. The first handoff, named Handoff 1, happened when the RSSI value was equal to -80dB and the link loss was equal to 18%. As it is observed, after the trigger and the Handoff 1 event the End-to-End packet loss kept decreasing. Despite that, after a short period of time the End-to-End loss increased again, something that led to a new handoff event, named Handoff 2. The RSSI value during the second handoff was equal to -82dB, where the link loss value was equal to 12%. It is important to note that even though the link loss percentage is lower in this case, the RSSI value is also lower and their combination creates sufficient conditions for a handoff. After the second handoff event, the packet loss had a decreasing trend again.

The proposed FLMC solution was developed based on decentralized information without having any global knowledge about the network condition. Therefore, the decision to handoff or not is based on locally available information that the MN has at the specific time and it cannot predict future losses or disconnections. Thus, the performance of the proposed solution can only be determined regarding the packet loss metric, based on the ability to decrease the packet loss after a handoff event and to decrease the total average packet loss comparing to other solutions, like RSSI threshold based ones. Furthermore, a handoff triggering may not result

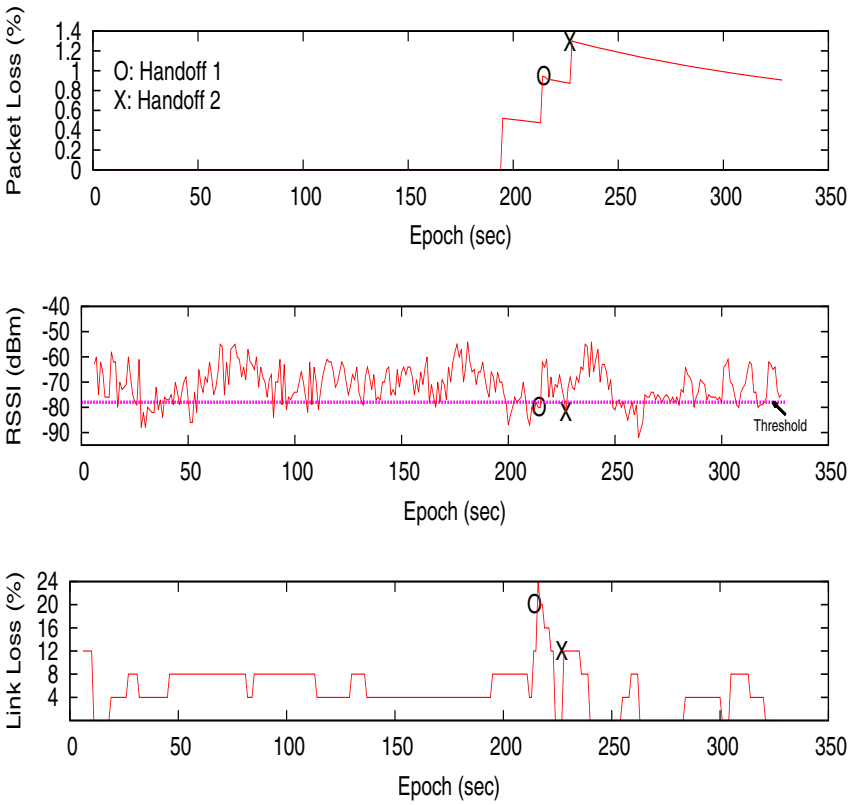


Fig. 12. Fuzzy Mobility Controller Operation

in a re-attachment either because no attachment point exists in the node's vicinity, or because the possible new attachment points do not have performance qualities that satisfy the controller's requirements. In addition, in a number of experiments, it was observed that there are cases at the beginning of the tests where an unnecessary handoff may occur. This is due to the fact that a loss, while not many packets have been sent on the link, indicates a high Link Loss percentage and wrongfully leads to a handoff. This observation was used to add a delay margin for trigger and handoff at the beginning of each test.

The main advantage of the fuzzy based mobility solution, compared with the RSSI threshold-based solution, is that it manages to decrease the average End-to-End packet loss to 2,45%. Figure 13 shows the comparison of the mobility solutions.

It is obvious that in the case of no mobility management (No Handoff) and in the case of the RSSI threshold-based solution the packet loss is high. This is due to the unpredictability of the environment and the RSSI behaviour. On the other hand, using the fuzzy mobility solution, those effects were reduced and a packet loss value within the 3% limit was achieved. Figure 13 shows also the breakdown

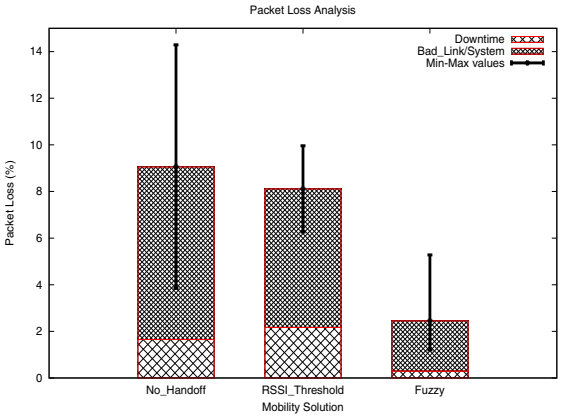


Fig. 13. Average End-to-End Packet loss Comparison

of the causes of packet loss. The losses are distinguished into two categories: the first category is when the MN has the ability to communicate with the parent node but some communication (bad link) or system losses occurred, and the second category is when the node is located in an area where it is not covered by the communication range of any other node (downtime). The majority of the losses in all the cases are due to system or bad links. On the other hand, the existence of packet losses that occurred due to uncovered areas provides a hint that a better placement of the fixed nodes in the network or the addition of more fixed nodes could help minimizing the packet loss. This issue can be considered as part of our future work.

Further to the End-to-End packet loss, Figure 14 shows a power consumption comparison of the mobility solutions.

It is clear that both solutions consume more power compared with the scenario where the MN is moving in the testbed without mobility management. This is due to the fact that in order to find a better position more scanning slots are required. Comparing the two mobility solutions, one can observe that the fuzzy solution performs better than the RSSI threshold solution with a total power consumption decrease of 10,78%. The reason of that, is the fact that the fuzzy solution performs fewer triggers and therefore has less scanning slots. In addition, it is worth noting that the transmission power consumption of the RSSI threshold based solution is increased compared with the fuzzy based solution. This is due to the fact that the increased packet loss leads to more retransmissions of data packets.

Moreover, based on Figure 15 the fuzzy mobility solution has increased the effective triggers (ratio of successful handoff triggers) from 4,15% to 8,1% by decreasing at the same time the average total number of triggers from 108,5 to 18,5. The reduction of the unnecessary triggers leads to the reduction of the energy consumption.

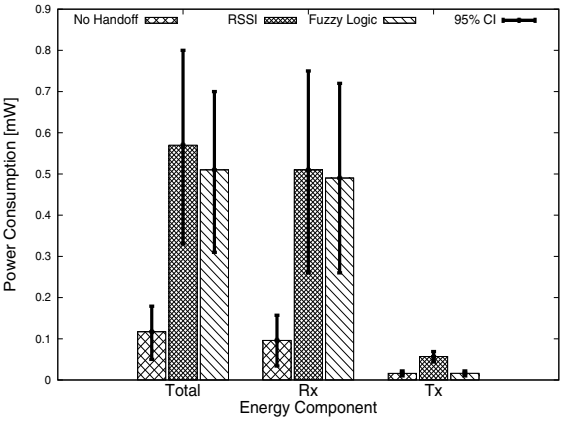


Fig. 14. Power Consumption Comparison

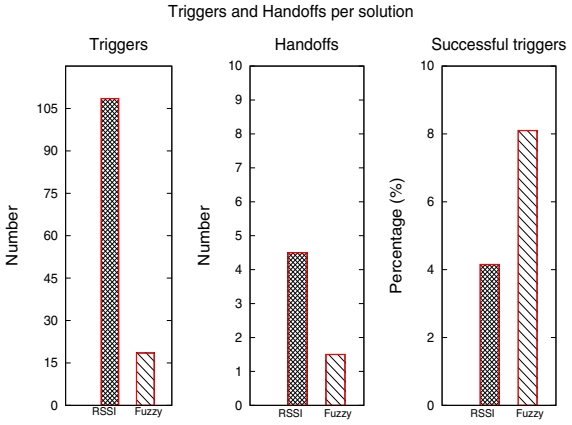


Fig. 15. Triggers and Handoffs for Mobility solutions

Furthermore, the packet delivery delay of both mobility solutions (Figure 16) is inside the limit of 1 second, whereas in the case there is no mobility management this delay is over 1 second. The reason is that the packets are kept in the queue for longer time due to the fact that the MN could be outside the transmission range of its parent node.

We have also used the COOJA simulator to evaluate the proposed solution using higher packet rates. To do so, we have increased the packet rate from 1 packet per 3 seconds to 1 packet per second. Based on the initial results, it was clear that the higher data rate does not affect the operation of the mobility solutions since in all the metrics the results are close to the lower packet rates results. The only remarkable point is the fact that in case of higher packet data

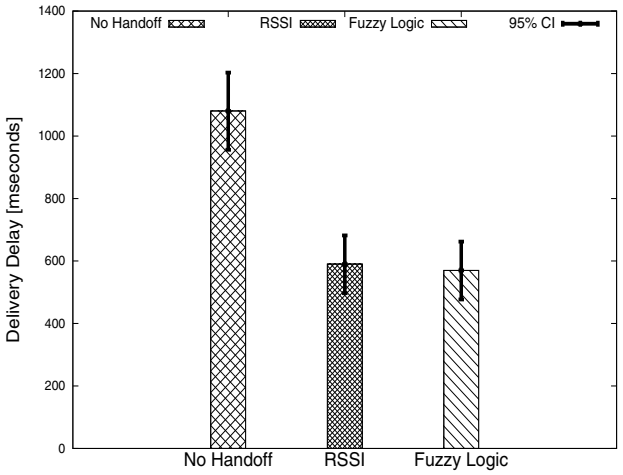


Fig. 16. Packet Delivery Delay

rates (for example, 1 packet per second) the total power consumption is increased by 4.1%. This is due to the increased number of packet transmissions.

In addition, Table 5 shows the percentage of the RSSI density in specific value ranges. As we can observe the use of the FLMC increased the number of high RSSI values compared to the case of not implementing mobility support.

Table 5. RSSI Density

Solution/Ranges in dBm	-50 to -60	-60 to -70	-70 to -80	-80 to -90	-90 to -95
Without Mobility support (%)	24.6	31.5	26	11.1	6.8
FLMC (%)	47.9	34.2	13.8	2.8	1.3

Furthermore, we present the overhead of Fuzzy Logic-based solution in terms of code size and execution time compared to the S-GINMOB solution. Comparing the mobility code size, we observed that the FLMC solution requires 808 bytes where RSSI-based solution (S-GINMOB) requires 50 bytes. In addition, we observed that the FLMC solution results the highest execution time, which is around 5 times more than the RSSI-based solution (S-GINMOB). Despite all these, the Fuzzy Logic-based solution results in a better way of controlling the handoff procedure, therefore any execution overhead is minimized due to the positive effects of the FLMC solution (fewer packet losses, less power consumption and higher on-time triggering).

Concluding, it is obvious that the fuzzy logic based mobility solution performs better in comparison with the RSSI-based mobility solution, and it fulfils some basic performance requirements that were set for the specific application environment (e.g. End-to-End packet loss less than 3% and an End-to-End delivery delay of no more than 1 second).

7 Conclusions

In this work a holistic approach in designing and implementing a mobility management solution in WSN, to support mobile workers inside an industrial environment was taken. The proposed mobility solution efficiently maintains the connectivity of the mobile node by controlling the handoff procedure. In the design of this solution network state variables are used, which are available by all mobile sensor nodes. Thus, the proposed mechanism is generically applicable to any WSN industrial environment or testbed setting with mobility requirements. This work moves beyond simple RSSI-based mobility solutions by proposing an intelligent controller, based on fuzzy logic, in order to help mobile sensor nodes to control handoffs with a need for performance guarantees. The applicability and operability of the proposed mobility solution was validated in a real testbed setting inside an industrial environment, as is an oil refinery. The results clearly show that the proposed mobility solution outperforms the RSSI-based mobility solution, in terms of packet loss, packet delivery delay, energy consumption, and ratio of successful handoff triggers. As future work, it is expected that better node placement solutions can be provided to solve the open issues of the uncovered areas and further minimize the packet losses. In addition, scenarios with adaptive power control, dynamic adjustment of the decision probability Threshold, higher packet rate and multiple MNs will be considered.

References

1. Vassiliou, V., Antoniou, J., Pitsillides, A., Hadjipollas, G.: Simulating Soft Handover and Power Control for Enhanced UMTS. In: 2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1646–1651 (September 2005)
2. O'Donovan, T., Brown, J., Roedig, U., Sreenan, C.J., do O, J., Dunkels, A., Klein, A., Sa-Silva, J., Vassiliou, V., Wolf, L.C.: GINSENG: Performance Control in Wireless Sensor Networks. In: REALWSN 2010, pp. 1–3 (June 2010)
3. Silva, R., Zinonos, Z., Sa Silva, J., Vassiliou, V.: Mobility in WSNs for Critical Applications. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 451–456 (July 2011)
4. Zinonos, Z., Chrysostomou, C., Vassiliou, V.: Controlling the Handoff Procedure in an Oil Refinery Environment Using Fuzzy Logic. In: 2012 10th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC), pp. 477–483 (December 2012)
5. Song, J., Han, S., Mok, A., Chen, D., Lucas, M., Nixon, M., Pratt, W.: WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control. In: Proceedings of the 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2008, pp. 377–386 (2008)
6. ISA-100.11a-2009: Wireless Systems for Industrial Automation: Process Control and Related Applications (2009), `http://www.isa.org/ISA100/`
7. (2012) GINSENG - Performance Control in Wireless Sensor Networks, `http://www.ict-ginseng.eu`

8. Zinonos, Z., Vassiliou, V., Chrysostomou, C.: Handoff Triggering for Wireless Sensor Networks with Performance Needs. In: 18th IEEE Symposium on Computers and Communications (IEEE ISCC 2013), Split, Croatia (July 2013)
9. Zinonos, Z., Vassiliou, V.: S-GinMob: Soft-handoff Solution for Mobile Users in Industrial Environments. In: 2011 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 1–6 (June 2011)
10. Ramamurthy, H., Prabhu, B., Gadh, R., Madni, A.: Wireless Industrial Monitoring and Control Using a SMart Sensor Platform. IEEE Sensors Journal 7(5), 611–618 (2007)
11. Srinivasan, K., Levis, P.: RSSI is Under Appreciated. In: 2006 Third Workshop on Embedded Networked Sensors, EmNets (May 2006)
12. Srinivasan, K., Kazandjieva, M.A., Agarwal, S., Levis, P.: The beta-factor: Measuring Wireless Link Burstiness. In: 2008 Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys), pp. 29–42 (2008)
13. Fotouhi, H., Zuniga, M., Alves, M., Koubaa, A., Marrón, P.: Smart-HOP: a Reliable Handoff Mechanism for Mobile Wireless Sensor Networks. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 131–146. Springer, Heidelberg (2012)
14. Gonga, A., Landsiedel, O., Johansson, M.: MobiSense: Power-Efficient Micro-Mobility in Wireless Sensor Networks. In: 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), pp. 1–8 (June 2011)
15. Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P.: Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks. In: 2006 4th International Conference on Embedded Networked Sensor Systems (SenSys), pp. 419–420 (2006)
16. Wu, Y., Stankovic, J., He, T., Lin, S.: Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks. In: The 27th Conference on Computer Communications, INFOCOM 2008, pp. 1193–1201 (April 2008)
17. Fotouhi, H., Alves, M., Koubaa, A., Baccour, N.: On a Reliable Handoff Procedure for Supporting Mobility in Wireless Sensor Networks. In: The 9th International Workshop on Real-Time Networks RTN 2010 in Conjunction with the 22nd Euromicro International Conference on Real-Time Systems (ECRTS 2010), Brussels, Belgium (2010)
18. Zhang, W.: Handover Decision Using Fuzzy MADM in Heterogeneous Networks. In: 2004 IEEE Wireless Communications and Networking Conference, WCNC 2004, vol. 2, pp. 653–658 (March 2004)
19. Chan, P.M.L., Sheriff, R., Hu, Y., Conforto, P., Tocci, C.: Mobility Management Incorporating Fuzzy Logic for Heterogeneous a IP Environment. IEEE Communications Magazine 39(12), 42–51 (2001)
20. Liao, H., Tie, L., Du, Z.: A Vertical Handover Decision Algorithm Based on Fuzzy Control Theory. In: First International Multi-Symposiums on Computer and Computational Sciences, IMSCCS 2006, vol. 2, pp. 309–313 (2006)
21. Barolli, L., Durresi, A., Xhafa, F., Koyama, A.: A Fuzzy-Based Handover System for Wireless Cellular Networks: A Case Study for Handover Enforcement. In: Takizawa, M., Barolli, L., Enokido, T. (eds.) NBiS 2008. LNCS, vol. 5186, pp. 212–222. Springer, Heidelberg (2008)
22. Barolli, L., Anno, J., Xhafa, F., Durresi, A., Koyama, A.: A Context-Aware Fuzzy-based Handover System for Wireless Cellular Networks and its Performance Evaluation. J. Mob. Multimed. 4(3), 241–258 (2008)

23. Kulkarni, R.V., Forster, A., Venayagamoorthy, G.K.: Computational Intelligence in Wireless Sensor Networks: A Survey. IEEE Communications Surveys and Tutorials 13(1), 68–96 (2011)
24. Konar, A.: Computational Intelligence: Principles, Techniques and Applications. Springer (2005), http://books.google.com.cy/books?id=NuvAERUGUAAC
25. Mamdani, E.: Application of Fuzzy Algorithms for Control of Simple Dynamic Plant. In: Proceedings of the Institution of Electrical Engineers, vol. 121(12), pp. 1585–1588 (December 1974)
26. Lazzerini, B., Marcelloni, F., Vecchio, M., Croce, S., Monaldi, E.: A Fuzzy Approach to Data Aggregation to Reduce Power Consumption in Wireless Sensor Networks. In: Annual meeting of the North American Fuzzy Information Processing Society, NAFIPS 2006, pp. 436–441 (June 2006)
27. Munir, S., Bin, Y.W., Biao, R., Man, M.: Fuzzy Logic Based Congestion Estimation for QoS in Wireless Sensor Network. In: Wireless Communications and Networking Conference, WCNC 2007, pp. 4336–4341. IEEE (March 2007)
28. Liang, C.-J.M., Liu, J., Luo, L., Terzis, A., Zhao, F.: RACNet: a High-Fidelity Data Center Sensing Network. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys 2009, pp. 15–28 (2009)
29. O'Donovan, T., Brown, J., Buesching, F., Cardoso, A., Cecilio, J., D'o, J., Furtado, P., Gil, P., Kugel, A., Poettner, W.-B., Roedig, U., Sa-Silva, J., Silva, R., Sreenan, C., Vassiliou, V., Voigt, T., Wolf, L., Zinonos, Z.: The GINSENG System for Wireless Monitoring and Control: Design and Deployment Experiences. ACM Transactions on Sensor Networks 10(1) (February 2013), www.ict-ginseng.eu/documents.php?idfolder=10
30. Suriyachai, P., Brown, J., Roedig, U.: Time-Critical Data Delivery in Wireless Sensor Networks. In: Rajaraman, R., Moscibroda, T., Dunkels, A., Scaglione, A. (eds.) DCOSS 2010. LNCS, vol. 6131, pp. 216–229. Springer, Heidelberg (2010)
31. Zinonos, Z., Vassiliou, V., Ioannou, C., Koutroullos, M.: Dynamic Topology Control for WSNs in Critical Environments. In: 2011 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5 (February 2011)
32. Zinonos, Z., Silva, R., Vassiliou, V., Sa Silva, J.: Mobility Solutions for Wireless Sensor and Actuator Networks with Performance Guarantees. In: 2011 18th International Conference on Telecommunications (ICT), pp. 406–411 (May 2011)
33. Zinonos, Z., Vassiliou, V., Christofides, T.: Radio Propagation in Industrial Wireless Sensor Network Environments: From Testbed to Simulation Evaluation. In: 2012 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N), pp. 125–132 (2012)
34. Passino, K.M., Yurkovich, S.: Fuzzy Control, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1997)
35. Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., Marrón, P.J.: COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools 2009, pp. 27:1–27:7 (2009)

Author Index