Department of Electrical and Computer Engineering

Collaborative Area Monitoring Using Wireless Sensor Networks with Stationary and Mobile Nodes

Theofanis P. Lambrou

A Dissertation
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
at the
University of Cyprus

May 2011

APPROVAL PAGE

Doctor of Philosophy Dissertation

Collaborative Area Monitoring Using Wireless Sensor Networks with Stationary and Mobile Nodes

by

Theofanis P. Lambrou

Research Supervisor ____ ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ____
Christos G Panayiotou

Committee Member (Chairperson) ____ ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ____
Marios Polycarpou

Committee Member ____ ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ____
Christoforos Hadjicostis

Committee Member ____ ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ____
Anthony Tzes

Committee Member ____ ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ____
Savvas Loizou

Date: 19 May 2011

Abstract

This dissertation investigates path planning problems for improving the coverage and detection performance of mixed sensor networks consisting of both static and mobile nodes. With recent advances in distributed robotics and low power embedded systems, such mixed wireless sensor networks (WSNs) are becoming attractive.

Conventionally, WSNs consist of a large number of static sensors randomly deployed in a large region of interest. However, dense random deployments of static sensors can not guarantee complete sensing coverage of large areas and also imply prohibitive cost and excessive radio interference. In this dissertation, a different WSN model that employs stationary and mobile sensor nodes (mixed WSN) is assumed. The main objective of this work was to develop path planning algorithms to enable mobile sensors to collaborate and sample areas not easily or effectively monitored by stationary sensors. The path planning approaches should comply with the specific requirements of sensor networks like distributed-collaborative processing, limited communication and computation and partial knowledge of the environment. In addition, proposed path planning methods, should search the whole sensor field and find as efficient a route as possible in terms of coverage improvement over time and/or event detection performance. However, satisfying all of these criteria, in such complex environments, is not always possible and thus a priority consideration is required.

In the context of this research work, several methods and algorithms to solve this path-finding problem with single and multiple mobile sensors, alongside the several properties and parameters of these algorithms, were extensively investigated. The main contribution of this dissertation is the development of a distributed collaborative path planning framework that enables mobile sensors to find their path autonomously in order to search the sensor field area and minimize event detection delay. The path is estimated on-line using local information which is essential in the context of large, distributed WSNs. Further, the proposed approach was also transferred from simulation to a real-world mixed sensor network test-bed and evaluated experimentally.

This dissertation provides a better understanding and a deeper knowledge of the current problems of mixed sensor networks. The path planning framework developed is easily scalable to large numbers of mobile sensors and different static sensor deployments. Applications of these types of mixed sensor networks include sensor coverage improvement, search and detection of event sources, and dynamic environment monitoring.

Περίληψη

Αυτή η διατριβή εξετάζει τα προβλήματα σχεδιασμού διαδρομών για την βελτίωση της κάλυψης και του εντοπισμού γεγονότων σε μικτά δίκτυα αισθητήρων που αποτελούνται από σταθερούς και κινητούς κόμβους αισθητήρων. Τέτοια μικτά ασύρματα δίκτυα αισθητήρων γίνονται ελκυστικά με τις πρόσφατες προόδους στα κατανεμημένα ρομποτικά συστήματα και στα ενσωματωμένα συστήματα χαμηλής ισχύος.

Συμβατικά, τα δίκτυα αισθητήρων αποτελούνται από έναν μεγάλο αριθμό σταθερών αισθητήρων οι οποίοι εγκαθίστανται σε τυχαίες θέσεις σε μια μεγάλη περιοχή υπό παρακολούθηση. Ωστόσο, οι πυκνές τυχαίες εγκαταστάσεις των σταθερών αισθητήρων δεν μπορούν να εγγυηθούν πλήρη κάλυψη παρακολούθησης μεγάλων περιοχών και προϋποθέτουν απαγορευτικό κόστος και υπερβολικές παρεμβολές επικοινωνίας. Σε αυτήν την διατριβή, υποθέτουμε ένα διαφορετικό μοντέλο δικτύου αισθητήρων το οποίο χρησιμοποιεί σταθερούς και κινητούς κόμβους αισθητήρων (μικτό ΑΔΑ) και ο κύριος στόχος είναι η ανάπτυξη αλγορίθμων σχεδιασμού πορειών που θα επιτρέπουν στους κινητούς αισθητήρες να συνεργάζονται και να παρακολουθούν περιοχές που δεν παρακολουθούνται από τους σταθερούς αισθητήρες. Οι μέθοδοι εύρεσης διαδρομών των κινητών κόμβων πρέπει να είναι συμβατές με τις ειδικές απαιτήσεις των δικτύων αισθητήρων όπως τον κατανεμημένο-συνεργάσιμο υπολογισμό, την περιορισμένη επικοινωνία και υπολογιστική ικανότητα και τη μερική γνώση του περιβάλλοντος. Επιπλέον, οι προτεινόμενες μέθοδοι προγραμματισμού πορειών, πρέπει να επιτρέπουν στους κινητούς κόμβους να παρακολουθούν όλες της ακάλυπτες περιοχές του σταθερού δικτύου αισθητήρων και να βρίσκουν μια όσο το δυνατόν καλύτερη διαδρομή σε σχέση με την βελτίωσης της κάλυψης σε σύντομο χρονικό διάστημα ή της απόδοσης εντοπισμού γεγονότων. Εντούτοις, η ικανοποίηση όλων αυτών των κριτηρίων σε τέτοια σύνθετα περιβάλλοντα δεν είναι πάντα δυνατή.

Στο πλαίσιο της ερευνητικής εργασίας διερευνήσαμε διάφορες μεθόδους και αλγόριθμους για να λύσουμε πρόβλημα εύρεσης πορειών ενός ή πολλαπλών κινητών αισθητήρων και διερευνήθηκαν εκτενώς διάφορες ιδιότητες και παράμετροι αυτών των αλγορίθμων. Η κύρια συνεισφορά αυτής της διατριβής είναι η ανάπτυξη ενός κατανεμημένου συνεργατικού πλαισίου προγραμματισμού πορειών που επιτρέπει στους κινητούς αισθητήρες να υπολογίζουν την διαδρομή που θα ακολουθήσουν αυτόνομα με στόχο να παρακολουθούν το δίκτυο αισθητήρων και να ελαχιστοποιούν την καθυστέρηση εντοπισμού γεγονότων. Η διαδρομή υπολογίζεται καθοδόν χρησιμοποιώντας μόνο τοπικές πληροφορίες, αυτό είναι πολύ σημαντικό στα πλαίσια μεγάλων και κατανεμημένων δικτύων αισθητήρων. Η προτεινόμενη μέθοδος έχει επίσης μεταφερθεί από το μοντέλο προσομοίωσης σε ένα πραγματικό πειραματικό δίκτυο αισθητήρων κινητών και σταθερών κόμβων και αξιολογήθηκε πειραματικά.

Αυτή η διατριβή παρέχει καλύτερη κατανόηση και βαθύτερη γνώση στα τρέχοντα προβλήματα των μικτών δικτύων αισθητήρων και το πλαίσιο σχεδιασμού διαδρομών που αναπτύχθηκε υποστηρίζει μεγάλους αριθμούς κινητών αισθητήρων και διαφορετικές εγκαταστάσεις ανάπτυξης σταθερών αισθητήρων. Οι εφαρμογές αυτών των τύπων μικτών δικτύων αισθητήρων περιλαμβάνουν βελτίωση κάλυψης, αναζήτηση και εντοπισμός πηγής γεγονότος και δυναμικό έλεγχο του περιβάλλοντος.

Acknowledgments

Not everything that can be counted counts, and not everything that counts can be counted.

–Albert Einstein

This work would not have been possible without the support, encouragement, and feedback of a number of people. I would initially like to express my sincerest gratitude to my advisor, Dr. Christos Panayiotou for his outstanding guidance, invaluable encouragement and consistent patience through the course of this research work. Thanks for leading me into the fascinating world of research and also for providing me moral support and guidance, which will benefit my personal and career development. I am also grateful to Professor Marios Polycarpou, for providing perceptive advice and broadening my scope on cooperative control design for multi-agent systems. Without his professional insight, this study would be very difficult.

I am also deeply indebted to my co-authors and collaborators. I thank Dr. Christos Anastasiou from Environmental Engineering for providing useful comments and guidance regarding environmental sensing and Dr. Santiago Felici-Castell of the University of Valencia for providing useful comments and suggestions regarding sensor networks deployment issues. Their advice and good humor on academics and life made my PhD journey more enjoyable. Special thanks should be directed to Professor Anthony Tzes, Dr. Christoforos Hadjicostis, and Dr. Savvas Loizou for serving on the dissertation committee and for their helpful comments and feedback for improving this dissertation, and many thanks my colleagues and friends at the University of Cyprus for their assistance and friendship.

For financial support, I am very grateful to the ODYSSEAS project funded by the Cyprus Research Promotion Foundation under contract ENIΣX/0505/30 and the CRUISE, WATERBEE and C4C projects funded by the European Commission.

Finally, my warmest thanks go to my family for their understanding, love and support over the years, which was indispensable for completing my research work.

To my wife and son.

Contents

List of Publications

Published Journal Articles

1. **T.P. Lambrou**, C.G. Panayiotou, *"Collaborative Area Monitoring Using Wireless Sensor Networks with Stationary and Mobile Nodes,"* in EURASIP Journal on Advances in Signal Processing, Special Issue on Signal Processing Advances in Robots and Autonomy, vol 2009, Article ID 750657, pp. 1-16, doi:10.1155/2009/750657
2. **T.P. Lambrou**, C.G. Panayiotou, S. Felici-Castell and B. Beferull-Lozano, *"Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks,"* in Wireless Personal Communications Journal, vol 2009, pp. 1-15, ISSN: 0929-6212, doi: 10.1007/s11277-009- 9717-0

Published/Accepted Conference Papers

1. **T.P. Lambrou** and C.G. Panayiotou, *"Improving area coverage using mobility in sensor networks,"* in the International Conference on Intelligent Systems And Computing: Theory And Applications, ISYC 2006, 6-7 Jul. 2006.
2. **T.P. Lambrou** and C.G. Panayiotou, *"Collaborative Event Detection Using Mobile and Stationary Nodes in Sensor Networks,"* in the 4th International Conference on Collaborative Computing: Networking, Applications and Work sharing, CollaborateCom 2007, 12-15 Nov. 2007.
3. **T.P. Lambrou**, C.C. Anastasiou and C.G Panayiotou, *"A Nephelometric Turbidity System for Monitoring Residential Drinking Water Quality,"* in the 1st International Conference on Sensor Networks Applications, Experimentation and Logistics, SensAppeal 2009, 24-25 Sept. 2009.
4. **T.P. Lambrou**, S. Felici-Castell, C. Panayiotou and B. Beferull-Lozano , *"Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks,"* in the 10th International Symposium on Wireless Personal Multimedia Communications , WPMC 2007, 3-6 Dec. 2007.

5. **T.P. Lambrou** and C.G. Panayiotou, *"Distributed Collaborative Path Planning in Sensor Networks with Multiple Mobile Sensor Nodes,"* in the 17th Mediterranean Conference on Control and Automation, IEEE MED 2009, 24-26 Jun. 2009
6. **T.P. Lambrou** and C.G. Panayiotou, *"A Survey on Routing Techniques supporting Mobility in Sensor Networks,"* in the 5th International Conference on Mobile Ad-hoc and Sensor Networks, IEEE MSN 2009, 14-16 Dec. 2009
7. **T.P. Lambrou** and C.G. Panayiotou, *"Area Coverage Vs Event Detection in Monitoring Applications using Mixed Sensor Networks,"* in the 18th International Federation of Automatic Control World Congress, IFAC WC 2011, accepted

Submitted Journal and Conference Papers

1. **T.P. Lambrou**, C.G. Panayiotou, *"A Testbed for Coverage Control using Mixed Wireless Sensor Networks,"* in Elsevier Journal of Network and Computer Applications, Special Issue on Advances in Simulation, Testbeds, and Application of Integrated Wireless Mesh and Sensor Networks, pp. 1-13, submitted
2. **T.P. Lambrou** and C.G. Panayiotou, *"On the Optimal Search Neighborhood in Mixed Wireless Sensor Network,"* in the 50th IEEE Conference on Decision and Control, CDC 2011, submitted
3. **T.P. Lambrou**, C.G. Panayiotou,, *"A Local Search Strategy for Dynamic Coverage in Mixed Wireless Sensor Networks,"* in IEEE Transactions on Mobile Computing, pp. 1-12, submitted

List of Figures

List of Tables

Chapter 1

Introduction

Rapid advances in Integrated Circuit (IC) technologies and micro-electrical-mechanical systems (MEMS) have enabled the generation of low-cost, low-power sensor nodes. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of wireless sensor networks (WSN), where large numbers of spatially distributed sensor nodes operating cooperatively in a wireless network to monitor physical or environmental conditions and collaborate to perform an otherwise difficult task. The technology of WSN promises to revolutionize the way we live, work, and interact with the physical environment.

The development of wireless sensor networks was originally motivated by military applications such as battlefield surveillance. However, WSNs are now used in many industrial and civilian application areas, including, among others, industrial process monitoring and control, machine health monitoring, environmental monitoring, infrastructure security, building and structures monitoring, healthcare applications, home automation and traffic control.

Recent progress in two seemingly disparate research areas; namely, distributed robotics and low power embedded systems, has led to the creation of mobile sensor nodes [3]. These mobile sensor nodes can move around by self-propelling means (e.g. wheels), or by attaching themselves to transporters (e.g. robots, vehicles). The ability of a sensor node to self-propel, or to otherwise influence its location, will be critical in sensor networks. The possibility of combining sensing, computation, communication and actuation to not only passively monitor the environment (like static sensor networks) but also to actively search or track, in some cases, mitigate problems considered.

Augmenting static sensor networks with few mobile nodes immensely benefits the functionality of the sensor network and helps solve many of the design problems of static sensor networks. Controlled mobility enables a whole new set of possibilities in sensor networks such as dynamic coverage, optimal deployment, detecting spatially and temporally spread environmental events, or network repair (e.g. positioning mobile nodes at points of disconnection or by having mobile nodes collecting data from stationary nodes).

In the last decade, researchers have begun to investigate the use of mobile robots or unmanned aerial vehicles (UAVs) in surveillance and exploration. Typical applications include air and ground surveillance, target detection, tracking, search and rescue operations. The use of multiple collaborative robots or UAVs is ideally suited for such tasks. However, the coordination of multiple autonomous robots poses significant theoretical and technical challenges. A common problem within this area is to find the paths that a group of robots should follow to reliably and efficiently achieve a common objective under constrains like obstacles, hazards, cooperation, sensing, motion, communication and computation capabilities. Effective collaboration strategies should provide near-optimal, robust and real-time computation performance.

This dissertation investigates the path finding problem for the navigation of single and multiple mobile sensors to improve the coverage and detection performance of static sensor networks. The path planning approaches should comply with the specific requirements of sensor networks like distributed-collaborative processing, limited communication and computation capabilities, partial knowledge of the environment as well as adaptivity to the environment. Additionally, proposed path planning methods, should cover the whole sensor field, while the goal is to find as efficient a route as possible in terms of coverage improvement over time or event detection performance. However, satisfying all of these criteria, in such complex environments is not always possible and thus some priority consideration is required.

1.1 Motivation

Conventionally, WSNs consist of a large number of densely deployed static sensors in a given region. Sensors remain stationary, in their initial deployed position, and waiting for the detection of some event. When the environment of interest is inaccessible, or is located in a hostile area, or when deployment is limited by budgetary constraints, sensors may be air-dropped from an aircraft, or via other means, resulting in random placements.

To achieve reliable monitoring of a large area with a WSN, typically, requires a very large number of stationary nodes (especially when random deployment is considered), implying a prohibitive cost and excessive (radio) interference. To better exemplify these, the monitoring of a lake or a reservoir dam may be considered. In this example, an event occurance may be the exceedance of a given threshold of water turbidity [4]. Assuming a lake area of 500km^2, and reliable sensing range of 1m for each sensor,

10^8 sensor nodes would be necessary (under grid deployment) to effectively monitor the lake, or more than 10^{10} sensors are needed, if random deployment is used. With current technology availability, these scenarios would be accompanied by an immense cost.

Therefore, considering the near-impossibility of reliably covering such an entire large area with stationary nodes, this research work focuses on finding an alternative way of monitoring the area by using a combination of several stationary and some mobile sensor nodes that collaborate in a way that improves the area coverage and/or the rapid detection of an event. Such a system, consisting of both stationary and mobile sensor nodes, will be referred to as a *mixed WSN*. The cost of mobile nodes can be comparable (same order of magnitude) to that of stationary nodes, despite the need for more advanced hardware and motion capabilities. Therefore, such a configuration will result in significant cost savings.

The aim of this work is to develop a path-planning framework where a set of mobile sensors will collaborate with stationary sensors, and with each other, in order to search an area, in an efficient manner, and locate events. A key concept of this collaborative framework is that the mobile sensors should sample the areas that are least covered (monitored) by the stationary sensors. An important component of the proposed framework is the fact that mobile nodes can autonomously decide their path by using local and on-line information (their own beliefs and measurements as well as information collected from nearby stationary sensors). It should be noted that such an approach is appropriate in the context of WSNs since it is not feasible to have an accurate global view of the state of the environment (i.e. some nodes may have failed or been displaced).

Finding searching paths that guarantee complete coverage of an arbitrary sensor field, in an efficient manner, under local information, is not feasible. Even when global information is assumed and simple instances of the problem are considered, finding complete coverage paths that are optimal with respect to the time needed to complete coverage has been proved to be an NP-complete problem. Therefore, it can be implied that an optimal solution of an arbitrary problem instance cannot be found in reasonable time, which can be attributed to a large number of uncovered regions and multiple mobile nodes. Instead, a heuristic path-planning method that provides fast and satisfactory solutions for any arbitrary problem instance, under local information, must be adopted, even though complete coverage and optimal solutions cannot be always guaranteed.

1.2 Objectives

The *main objective* of this research work is the development of a distributed on-line path-planning framework that enables the collaboration of mobile robots in computing their path, using only local information in the context of mixed WSNs. The development of such a path-planning algorithm should satisfy the following requirements:

- Distributed - such that each mobile sensor can make local decisions for enhancing the system's global performance, since the continuous collection of information from a large WSN and the computation in a central controller may be infeasible or too costly.
- Dynamic - to consider the changes taking place in the environment and in the WSN deployment. For instance, the occurrence of events may be spatially or temporally random, or some stationary sensors may fail, or be displaced, through time.
- Collaborative - to support collaboration of multiple mobile sensor nodes to achieve efficiently better system performance (area coverage and event detection) without the need of a central controller. Mobile nodes should be aware of the actions of other mobiles, thus collaborating and avoiding overlaps while accomplishing the common objective.
- Simple and computationally efficient - for implementation on tiny microcontrollers.
- Require limited amount of information - thus extending the lifetime of the wireless sensor network.
- Efficient - to provide near-optimal performance (e.g. minimize average event-detection delay)
- Adaptive - to perform consistently under different static WSN deployment densities, numbers of mobile sensor nodes, sensor field deployment area sizes, and event source localization algorithms.

1.3 Dissertation Organization and Overview

1.3.1 Dissertation Organization

This dissertation is structured as follows: Chapter 2 presents an extensive literature review of various pertinent issues. The subsequent two chapters present the investigation of some tools that enable the development of path-planning algorithms in the context of sensor networks. Specifically, Chapter 3 investigates path-planning methods for finding the path between two arbitrary points in a sensing field to improve coverage when a single mobile sensor is used. Chapter 4 investigates methods of finding possible destinations points for the mobile sensors. Centroids of uncovered regions in a sensing field are considered as such destination points. Chapter 5 builds upon the tools developed in the previous chapters and proposes a complete coverage path-planning method for a single mobile node, based on global knowledge of the sensor field. Chapter 6 proposes a distributed collaborative path-planning method for multiple mobile nodes that succeeds fast coverage improvement rate and given enough time almost complete coverage of the sensor field could be achieved. The remaining chapters adopt the algorithm presented in chapter 6 and investigate, in depth, several of its properties and parameters. Specifically, Chapter 7 investigates several collaboration schemes to reduce information exchange between mobile nodes without significant loss of the coverage performance. Chapter 8 integrates the measurements mobile sensors received in the path-planning algorithm and extents the overall framework further. In addition, several other cost functions as well as target assignment schemes that have been considered are also presented in Chapter 8. Chapter 9 investigates the problem of finding the optimal path that minimizes the expected detection delay for a simple case of the problem and derives the optimal solution. Further extending this simple case proves that such simple problems are NP-complete. A new global heuristic path-planning approach to tackle such simple problems is being proposed, and the insights gained from the analysis help us to further investigate some of the parameters of the proposed distributed path-planning approach. Chapter 9 concludes that a very important parameter of the proposed path planning algorithm is the searching neighborhood radius. Consequently, Chapter 10 proposes a surrogate metric to select the searching neighborhood size that enhances the dynamic coverage improvement-rate based on the statistical properties of stationary nodes deployment as well as other parameters used in the presented path-planning algorithm. Chapter 11 describes the transfer of the proposed path planning algorithm from simulation to a real-world mixed sensor network test-bed to experimentally evaluate the proposed approach. Finally, conclusions and future directions of this work are elaborated in Chapter 12.

1.3.2 Dissertation Overview

The summary of each chapter follows:

Chapter 2 presents an extensive literature review of the state of the art algorithms, protocols, implementations and solutions proposed for the problem of coverage, event detection and localization in the context of wireless sensor networks when mobile sensor nodes are available, and the problem of path planning and cooperative search in the context of mobile robotics and unmanned aerial vehicles. The first section of this chapter is a brief summary of the accomplished work that more directly pertains to this dissertation, while the remaining sections provide the background and cite recent work related to various issues raised in this dissertation; such as the coverage issue in WSN, the path planning algorithms for mobile robotics, the cooperative path planning and control for multi-robot systems and finally the event localization and odor plume source localization.

Chapter 3 presents the development and investigates the parameters of various path-planning algorithms for a mobile node to improve the area coverage of a stationary sensor field. The path planning algorithms developed allow the mobile sensor node to navigate towards a given, predefined, target-point in the WSN while it passes through areas that are not adequately sampled by stationary nodes thus improving area coverage. Two different families of path-planning algorithms, based on artificial potential field and receding-horizon approaches, were developed. The fact that the receding horizon family of algorithms achieves good area coverage improvement and is computationally tractable is demonstrated. Algorithms based on the potential field approach are also computationally efficient, but may require the use of heuristics to allow them to escape from local minima.

Chapter 4 proposes two coverage hole-detection algorithms able to provide automatically the coordinates of coverage holes in a WSN, which can be used as target points for the navigation of mobile sensors. The theoretical analysis of the computation complexity of the developed algorithms, as well as the simulation results of the coverage hole detection efficiency, are presented. A novel algorithm, entitled "Zoom algorithm", which efficiently provides the locations of coverage holes in a sensor network with negligible computation needs, was developed.

Chapter 5 presents a case study for complete area coverage of the uncovered regions in a sparse sensor network using a mobile sensor node. The proposed complete coverage path-planning method requires global information to solve the problem (i.e. positions of all static sensor nodes) and thus can be classified as centralized. The algorithm consists of two major components, the global estimation of all the coverage holes in the sensor field and a path decision method for visiting all the coverage holes while avoiding passing over the region already covered by stationary sensor nodes. The proposed path-planning method succeeds complete area coverage of a sparse WSN.

Chapter 6 presents the development of a distributed collaborative framework for complete area coverage of the uncovered regions in stationary sensor networks using a group of mobile sensor nodes. In the proposed architecture, a small set of mobile nodes collaborate with the stationary sensor nodes, and with each other, in order to cover areas not covered (monitored) by stationary sensors. An important element of the proposed system is the ability of each mobile node to autonomously decide its path, based on local information, which is essential in the context of large WSNs. Mobile nodes autonomously, and collaboratively, navigate through the field in order to improve the area coverage. Evaluation results indicate the ability of the proposed algorithm to significantly improve the area coverage, when compared to other well-known path planning approaches.

Chapter 7 investigates the information-exchange between mobile nodes in the distributed on-line path planning algorithm for complete coverage proposed in chapter 6. The main objective of chapter 7 is to investigate collaboration schemes among sensor nodes in which the amount of exchanged information between nodes is reduced without significant loss of area coverage performance.

Chapter 8 investigates the collaborative area monitoring path-planning problem in stationary sensor networks using multiple cooperative mobile sensor nodes that patrol the region of interest and locate events. The development of an architecture where a set of mobile sensors collaborate with the stationary sensors in order to reliably detect and locate events is presented. Under this setting, when stationary sensors have a "suspicion" that an event may have occurred, they report it to a mobile sensor that can then move closer to the suspected area and verify the event. An important component of the proposed architecture is that the mobile nodes autonomously decide their path based on local information. An extensive set of simulation results, indicating the effectiveness of the proposed algorithm, are presented.

Chapter 9 considers the problem of deciding the optimal searching path for a single mobile node that minimizes the expected event detection time when several large, unconnected, uncovered regions exist in the sensor field. Finding the optimal searching path is an NP-complete problem (it can be reduced to the multi-agent travelling salesman problem), which implies that we cannot find the optimal solution of an arbitrary problem instance in a reasonable time (due to a large number of uncovered regions and multiple mobile nodes), but must adopt a heuristic solution method. Thus a centralized heuristic path-planning strategy is presented in sequel to enable mobile sensors to decide their searching paths when several large unconnected, uncovered regions exist in the sensor field. This heuristic method has been compared to the general distributed path planning method presented in chapter 8 which remains valid for the general case where uncovered regions created by random deployment of static sensor nodes. Both analytical and simulation results indicate that a “local” search is better than a “global” search in the context of mixed sensor networks and that that the selection of the searching neighborhood radius in general distributed path planning method is of critical importance.

Chapter 10 proposes a surrogate metric that can be used in order to select the optimal searching neighborhood in the proposed distributed path-planning algorithm that enhances the dynamic coverage improvement rate. This approximation metric is based on the theory of coverage processes and is given by the variance of vacancy (i.e. area not covered) when a number of stationary sensors with identical sensing radius are distributed independently within a two dimensional region. Evaluation results demonstrated the effectiveness of this approach.

Chapter 11 describes the implementation, on real hardware, of the proposed on-line distributed path-planning algorithm for improving area coverage in a sparse sensor network deployment with stationary nodes when two cooperative mobile sensor nodes are used. The experimental test-bed of the proposed mixed WSN was designed and developed using off-the-shelf components in order to reduce the overall system implementation cost. Specifically, mobile sensor nodes prototypes and an indoor vision-positioning system were designed, developed and calibrated alongside their firmware counterparts.

Chapter 12 presents some concluding remarks and indicates possible future directions.

1.4 Contributions

This dissertation contributes to the field of collaborative path planning of multi-robot systems and collaborative area monitoring in the context of mixed sensor networks. Specifically, the main contributions of this dissertation are the following:

- A novel coverage hole detection algorithm capable of efficiently providing the locations of coverage holes in a sensor network with negligible computation needs was developed.

- A global off-line path-planning algorithm for complete coverage of uncovered regions in sparse stationary WSN was introduced. This algorithm requires global knowledge of the stationary sensor deployment and supports only one single mobile node.
- A distributed on-line path-planning algorithm, which allows a group of mobile nodes to autonomously and collaboratively navigate through the sensor field, sample the areas not covered by the stationary sensor nodes, and adapt their path to the sensor field changes, was developed. The algorithm is computationally efficient and uses a dynamic receding horizon approach where each mobile computes its path is on-line using only local information.
- An efficient event-driven communication scheme, which enables mobile nodes to sufficiently reduce information exchange (communication cost) without significantly affecting the overall coverage performance, was introduced.
- A dynamic target-switching policy that allows mobile sensors to search and locate events sources was introduced. Switch path-planning objectives (between source search and coverage) depends on the state of the sensor measurements obtained by the mobile node and its neighbors. The distributed on-line path-planning algorithm has been further extended by incorporating this switching policy and additional collaboration protocols for when mobiles are searching for event sources.
- The solution of the optimal path problem for a single mobile node that enables the mobile to search several non-connected uncovered locations with the minimum average detection delay or with the maximum dynamic coverage over time was achieved. The resulting optimal strategy confirms that it is better to search areas that are less likely to hide a target but are located closer to the mobile node, rather than heading towards the most likely area. It was also proven that the optimal search path strategy is an NP-complete problem and, consequently, another heuristic global path-planning strategy that provides computationally tractable and near-optimal solutions to this special case problem was proposed.
- A surrogate metric for approximating the optimal search neighborhood radius (where each mobile is searching for a coverage hole centroid) in the distributed on-line path planning algorithm that enhances the dynamic coverage over time was introduced. The proposed approximation associates the optimal neighborhood radius with several other parameters used in the path-planning method.
- The distributed path-planning algorithm was implemented and experimentally evaluated on a real-world mixed WSN test-bed that consisted of stationary sensor and mobile robots that collaborated to monitor uncovered regions.

Chapter 2

Literature Review

2.1 Overview

This chapter presents an extensive literature review of the state of the art algorithms, protocols, implementations and solutions proposed for the problem of coverage, event detection and localization in the context of wireless sensor networks when mobile sensor nodes are available, and the problem of path planning and cooperative search in the context of mobile robotics and unmanned aerial vehicles.

The next section gives a brief summary of the proposed work that is more relevant to this dissertation, for more details and extensive descriptions the reader is prompted to continue with the forthcoming sections.

2.2 State of the art and related work

The work presented in this section is partially related with two research fields, the area coverage in WSNs and path planning in the fields of mobile robotics and unmanned aerial vehicles. Although many researchers in the WSNs area have studied the coverage problem, to the best of our knowledge, this is the first time that a general architecture is proposed that combines the coverage problem with distributed path planning algorithms so that the mobile nodes can navigate towards poorly covered areas. The benefit of this approach is that events that would have remained undetected can now be detected.

Next, we present a brief overview of papers that address the coverage problem in the context of WSNs. For a more thorough survey of the coverage problem the reader is referred to [5] and [6]. Also [7] presents a survey of the holes problem (coverage, routing, jamming, sink holes, etc) in WSNs.

In [8] authors proposed the Grid Scan algorithm to find the maximum blind region in order to deploy additional static sensors. The proposed scheme is a multi-step scheme where each step is a greedy exploration process over all potential redeployment points. Only one potential point with maximum number of neighboring non-covered grids in the sensing range is chosen as the point at which the newly added sensor node will be deployed. As shown in [9], the proposed Zoom algorithm is computationally significantly more efficient.

Next, we present several other approaches that have been proposed in order to determine the coverage holes where mobile nodes can be deployed. All these approaches do not consider the path that the mobile should follow in order to reach to its destination. In [10] authors used Voronoi diagrams to discover the existence of coverage holes. A node needs to know the location of its neighbors to construct its Voronoi diagram. The diagram partitions the whole space into Voronoi polygons. Each polygon has a single node with the property that every point in the polygon is closer to this node than any other node. A sensor node compares its sensing disk with the area of its Voronoi polygon to estimate any local coverage hole. Three distributed self-deployment algorithms have been proposed to calculate new optimal positions to which mobile sensors should move to increase coverage: Vector based (VEC), Voronoi based (VOR) and Minimax algorithm.

The same authors in [11] describe a bidding protocol, for mixed sensor networks that use both static and mobile sensors to achieve a cost balance. Their algorithm considers a random initial deployment, where static sensors detect their local coverage holes based on Voronoi diagrams. The mobile sensors calculate coverage holes formed at their current position if they decide to leave their current position. The static sensors bid mobile sensors based on the size of their detected coverage hole. A mobile sensor compares the bids and decides to move if the highest bid received has a coverage hole size greater than the new hole generated in its original location due to its movement. The bids are broadcasted locally up to two hops and the static sensors are able to direct neighboring eligible mobile sensors to a point close to the farthest vertex of their Voronoi polygon. However, the local broadcast may prevent the bid messages reaching mobile sensors if they are located farther than two hops. In this case the authors propose a mixed architecture for the coverage problem.

In [12] authors address the problem of enhancing coverage in a mixed sensor network. They present a method to deterministically estimate the exact amount of coverage holes under random deployment using Voronoi diagrams and use the static nodes to estimate the number of additional mobile nodes needed to be deployed and relocated to the holes locations to maximize coverage. The static nodes also find out the optimal positions of those mobile nodes based on certain heuristics. In our case we use a small number of mobile nodes that move collaboratively using path planning algorithms in order to enhance the event detection probability of the stationary sensor network.

Sensor relocation has been studied in [13], which focuses on finding the target locations of the mobile sensors based on their current locations and the locations of the sensed events. In [14] a polynomial-time algorithm is presented in terms of the number of sensors to determine whether every point in the service area of sensor networks is covered by at least k sensors, where k is a predefined value. With this algorithm, WSNs work well in situations that require stronger coverage and impose more stringent fault-tolerant capability. In [15], the authors provide a polynomial-time, greedy, iterative algorithm to determine the best placement of one sensor at a time in a grid based scenario, such that each grid is covered with a minimum confidence level. They model the obstacles as static objects and assume that a complete knowledge of the terrain is available. As already mentioned, none of the aforementioned approaches considers the actual path that each mobile should follow.

Next, we present some path planning algorithms that have been proposed and are relevant to our work. A good overview of motion planning in robotics is given in [16]. The path planning algorithms presented in this dissertation have been motivated by the approach in [17] where an approach for cooperative search by a team of distributed agents is presented. In that approach two or more agents move in a geographic environment, cooperatively searching for targets of interest and avoiding obstacles or threats.

Authors in [18] use the concept of Voronoi diagrams and triangulation to provide polynomial-time worst case and best case algorithms for determining maximal breach path and maximal support path, respectively, in a sensing field. On similar lines, in [19], the authors use the concepts of minimal and maximal exposure paths to find out how well an object moving on an arbitrary path can be observed by the sensor network over a period of time. The algorithm in [19] uses certain graph theoretic abstractions and compute minimal exposure path using Dijkstra's single source shortest path algorithm or Floyd-Warshals all pair shortest path algorithm. These approaches are centralized and solve static problem instances in the sense that they do not allow changes in the field once the paths have been computed.

In [20], the authors have focused on the coverage capabilities that result from the continuous random movement of the sensors. However, in this dissertation, we develop distributed path planning algorithms for cooperative movement of the mobile sensors.

Finally, we present some approaches that address the coverage problem in mobile sensor networks (*all* sensors are mobile). In [21] authors have looked at the problem of how mobile sensors move collaboratively in order to search a region and also incorporate communication costs into the coverage control problem.

The coverage concept with regard to the many-robot systems was introduced by Gage [22], who defined three types of coverage: blanket coverage, barrier coverage, and sweep coverage. Potential field techniques for robot motion planning were first described by Khatib [23] and have since been widely used in the mobile robotics community for tasks such as local navigation and obstacle avoidance. A robot moving according to the potential will never hit obstacles, but it may get stuck in local minima.

Assuming that all sensors have motion capabilities, several approaches have been developed to address the coverage problem using the concept of potential fields [24], [25], and virtual forces in [26]. In [24], the authors propose a deployment strategy using mobile autonomous robots that maximize the area coverage with the constraint that each of the nodes has at least k neighbors. The sensing field is modelled using attractive and repulsive forces exerted on each node by all other nodes. The network stabilizes when equilibrium is reached, i.e., the net force on each node becomes zero. This approach is computationally expensive because as the network size grows or new nodes join, all nodes need to reconfigure themselves to satisfy the equilibrium criteria.

In a similar fashion, assuming that all sensors have motion capabilities, the authors of [25] proposed a potential field-based algorithm in which nodes are treated as virtual particles subjected to virtual force. Virtual forces repel the nodes from each other and from obstacles, and ensure that the initial configuration of nodes quickly spreads out to maximize coverage area. In [26], the authors presented another virtual-force-based sensor movement strategy to enhance network coverage after an initial random placement of sensors. A cluster head computes the new locations of all the sensors after the initial deployment that would maximize coverage and then nodes reposition themselves to the designated locations. After the execution of the algorithm and once the effective sensor positions are identified, a one-time movement is carried out to redeploy the sensors.

2.3 Coverage in WSN

Sensing coverage is one of the most fundamental problems in wireless sensor networks. In this section, we present and compare several state-of-the-art algorithms and techniques that aim to address this coverage issue. A typical large-scale WSN consists of thousands of sensor nodes deployed either randomly or according to some predefined statistical distribution over a geographical region of interest. Each sensor node can sense only a small portion of the environment. However, a group of sensors collaborating with each other can accomplish a much bigger task efficiently. The notion of area coverage can be considered as a measure of the quality of service (QoS) in a sensor network, for it means how well each point in the sensing field is covered by the sensors. Historically, three types of coverage have been defined in [22]:

- Blanket Coverage: To achieve a static arrangement of nodes that maximizes the detection rate of targets appearing in the sensing field,
- Barrier Coverage: To achieve a static arrangement of nodes that minimizes the probability of undetected intrusion through the barrier,
- Sweep Coverage: To move a number of nodes across a sensing field, such that it addresses a specified balance between maximizing the detection rate of events and

minimizing the number of missed detections per unit area.

The requirements of coverage may vary across applications. For instance, a military surveillance application possibly requires a high degree of coverage as it would want a region to be monitored by multiple nodes simultaneously, so that in the event of node failures the security of the region is not compromised. On the other hand, environmental monitoring applications possibly require a low degree of coverage. Coverage schemes can be classified in two categories based on certain inherent properties that are common to these schemes. These categories are:

- Exposure-based: Most of these strategies use tools from computational geometry, such as the Voronoi diagram and Delaunay triangulation, and are targeted towards applications that try to detect unauthorized intrusion in the network.
- Mobility-based: Algorithms in this category exploit mobility of nodes in order to achieve a better degree of coverage. These algorithms typically relocate nodes to optimal locations after an initial deployment, and try to spread nodes in a uniform way so that coverage is maximized.

Before describing the algorithms in detail, we first define the sensing models used in these algorithms and establish a common framework to make the presentation clear.

2.3.1 Sensing models and Coverage

A model is a description of the physical system that captures its important behaviors, while abstracting away the gory details that complicate analysis. The fundamental principle behind modeling is that it should be as simple as possible, but no simpler that it becomes unrealistic. According to [19, 27], sensing devices generally have widely different theoretical and physical characteristics. Thus, numerous models of varying complexity can be constructed based on application needs and device features. Interestingly, most sensing device models and empirical observations share one facet in common: sensing ability diminishes as distance increases. Having this in mind, for a sensor s_i, we express the general sensing or measurement model Z at an arbitrary point p as:

$$Z(s_i, p) = \begin{cases} \frac{\lambda}{r(s_i,p)^\alpha}, & r_{\min} \le r(s_i,p) \le r_{\max} \\ 0, & otherwise \end{cases} \tag{2.1}$$

where $r(s_i, p)$ is the Euclidean distance between the sensor s_i and the point p, and parameters $r_{\min}$ and $r_{\max}$ define the range of a sensor's sensing capability and the positive constants λ and α are sensor technology-dependent parameters (λ indicates the energy emitted by events occurring at point p and α is the exponent of the power law indicating how the signal decays in the environment).

A sensor node detects an event if the received signal strength is greater than the threshold value of detection, known as the sensing sensitivity. The detection process depends on the strength of the emitted signal, behavior of the environment and the sensor hardware of the node.

Boolean sensing model

The simplest model for detection is the binary sensing model, according to which each each sensor has a certain sensing range r_s. A sensor can only sense the environment and detect events within its sensing range. This model ignores the dependency of the condition of the environment and the strength of the emitted signal.

Probabilistic sensing model

Despite its simplicity, the above Boolean sensing model does not capture the degradation of a sensor's sensing capability as the distance between the sensor and measuring point increases. The binary sensing model can be extended to a more realistic one, called the probabilistic sensing model [28, 29]. In this model, a quantity r_u is defined in order to capture the uncertainty in sensor detection. According to this model, the probability that a sensor detects an event to a distance r is

$$p(r) = \begin{cases} 1, & r \leq r_u \\ e^{-\lambda(r-r_u)^{\alpha}}, & r_u < r < r_s \\ 0, & r \geq r_s \end{cases} \tag{2.2}$$

where, r_u defines the starting of uncertainty in sensor detection, r_s is the maximum sensing range of the node and the parameters λ and α are adjusted according to the physical properties of the sensor and the environment. This model is more general because it becomes Boolean sensing model when $r_u = r_s$.

Area Coverage

The area coverage is defined as the ratio of covered area by the sensor network to the area of interest. In other words, it defines the probability that an event can be detected by at least one sensor node in the sensor field. The area coverage depends on the sensing model, number of nodes, node placement strategy. Consider a vast two-dimensional area $\mathcal{A}$ where N sensors are deployed uniformly (randomly) and the detection probability for each sensor is given by $p(r)$, thus the probability that the event will be undetected by an arbitrary sensor is equal to $(1-p(r))$ and the probability that the event will be undetected by all N sensor nodes randomly deployed is $Puncover = (1-p(r))^N$. The probability that the event will be detected by at least one of the N nodes is equal to the area coverage and is given by $C = 1 - Puncover = 1-(1-p(r))^N$. Now, for N large and $p(r)$ small the equation above can also be approximated by

$$C = 1 - e^{-Np(r)} \tag{2.3}$$

Proof: This is a result in stochastic geometry [30, 31] as the sensors are uniformly and independently distributed and thus following a two-dimensional Poisson point process with density $\mu = Np(r)$. It can also be derived from the approximation of a binomial random variable with large N and small p(r) using a poisson random variable with

$\mu = Np(r)$. Thus $C = P(X > 1) = 1 - P(X = 0) = 1 - exp(-\mu) = 1 - exp(-Np(r))$

In the case of the *boolean sensing model* assuming r_s and A are the sensing radius and area of the deployment respectively then the probability that an event will be detected by an arbitrary sensor is $p(r) = p{r_s}^2/A$. Thus the probability that the event will be detected by at least one of the N sensor nodes or the area coverage is given by

$$C = 1 - e^{-\frac{N}{A}\pi r_s^2} \tag{2.4}$$

In the case of the *Probabilistic sensing model* the probability that an event will be detected by an arbitrary sensor is $p(r) = \frac{1}{A} \int\limits_{x=0}^{x=r_s} p(x) 2\pi x dx$. Thus the area coverage can be found from 2.3 where $p(r)$ is given above. In the case where $r_u = 0$ and $\alpha = 1$ the area coverage is given by [29]

$$C = 1 - \exp\left(\frac{2\pi N}{A\lambda^2}\left(\left(\lambda r_s + 1\right) e^{-\lambda r_s} - 1\right)\right) \tag{2.5}$$

2.3.2 Coverage based on exposure for WSN

Approaches to solve the coverage problem using the notion of exposure is basically a combinatorial optimization problem formulation. Two kinds of viewpoints exist in formulating the coverage problem: 1) worst-case coverage, and 2) best-case coverage.

In the *worst-case coverage*, the problem is formulated with the goal to find a path through the sensing region such that, an object moving along that path has the least observability by the nodes, and thus, the probability of detecting the moving object is minimum. The two well-known approaches to the worst-case coverage problem are the Minimal Exposure Path [19] and the Maximal Breach Path (worst covered path) [18].

In the *best-case coverage* problem formulation, the goal is to find a path that has the highest observability, and therefore, an object moving along such a path will be most probable to be detected. The two approaches to solve the best-case coverage problem are the Maximal Exposure Path [32] and the Maximal Support Path [18].

Informally exposure path is a measure of how well a sensing field is covered in terms of the expected ability to detect a moving target. Exposure of a stationary sensor network has been studied in [19, 32] and defined as the path integral of a sensing function (that is inversely proportional to the distance of the target from sensors) along a path $p(t)$ as the target moves during the time interval $[t1, t2]$ as shown below

$$E(p(t), t_1, t_2) = \int_{t_1}^{t_2} I(F, p(t)) \left|\frac{dp(t)}{dt}\right| dt \tag{2.6}$$

where the sensing function $I(F, p(t))$ is a measure of sensitivity Z given by Eq. 2.1 at a point p on the path by either the closest sensor $I_c(F, p(t)) = Z(s_{\min}, p(t))$ or by all the sensors $I_a(F, p(t)) = \sum_1^N Z(s_i, p(t))$ in the sensing field where N is the number of sensors contribute a certain value of sensitivity to the point p depending on their distance from it. This definition of exposure as given by Eq. 2.6 is a path-dependent value as given two points in the sensing field, different paths between them are likely to have different exposures.

The minimal exposure path between two arbitrary points in a sensing field is the one which minimizes the value of integral given by Eq. 2.6. In [19] the problem in the case of many sensors is transformed from the continuous domain into a tractable discrete domain by using a grid. The minimal exposure path is then restricted to straight line segments connecting any two consecutive vertices on the grid. This approach transforms the grid into an edge weighted graph, and computes the minimal exposure path using Djikstra's single-source shortest path algorithm or Floyd-Warshal's all-pair shortest path algorithm. In [32], a distributed localized algorithm based on variational calculus, and a grid-based approximation algorithm are used to find expressions for the minimal exposure path for the cases of single sensor and multiple sensors, respectively.

The maximal exposure path between two arbitrary points in a sensing field is defined as the path following which the total exposure, as given by Eq. 2.6, is maximum. It can be interpreted as the path having the best quality of coverage. It is shown that finding a maximal exposure path is NP-hard and can be reduced to the known NP-hard problem of finding the longest path in an undirected weighted graph. Several heuristics are proposed in [32] to achieve near-optimal solutions under certain constraints, such as bounded object speed, path length, exposure value, and time of traversal.

Another very similar concept to the worst-case coverage path is the maximal breach path. In [18], it is defined as the path through a sensing field, such that, the distance from any point on the path to the closest sensor is maximum. The structure of Voronoi diagram is used to find such a maximal breach path. In two dimensions the Voronoi diagram [33] of a set of discrete nodes partitions the plane into a set of convex polygons or cells such that any point in the cell corresponding to a node is closer to that node than to any other node. Since by construction, the maximal breach path lies along the edges of a Voronoi diagram as this path maximizes the distance from the closest sensor nodes. An algorithm is described in [18] to find such a maximal breach path.

Alongside the concept of maximal exposure path, in [18] authors also defined another measure of the best-case coverage, called the maximal support path. A maximal support path through a sensing field between two points is a path for which the distance from any point on it to the closest sensor is minimum. Such a path can be found by replacing the Voronoi diagram by its dual, the Delaunay triangulation. Delaunay triangulation [33] is a triangulation of graph vertices, such that, the circumcircle of each Delaunay triangle does not contain any other vertices in its interior. The Delaunay triangulation can be obtained by connecting the nodes in the Voronoi diagram whose polygons share a common edge. The Delaunay triangulation can be used to find the two closest nodes by considering the shortest edge in the triangulation.

2.3.3 Coverage exploiting mobility for WSN

Initial random deployments result in accumulation of nodes at certain parts of the sensing fields while leaving other parts deprived of nodes. This section describes several coverage schemes that exploit mobility to relocate nodes to optimal locations and improve coverage. The first algorithms are based on the notion of potential field and virtual forces, where the mobile nodes could spread out from an initial configuration in order to improve area coverage. The next three algorithms known as the VEC, VOR, and Minimax are based on the structure of Voronoi diagram in which nodes are relocated to fill up coverage holes. Then, we describe algorithms for improving area coverage in the context of mixed sensor networks where a limited number of mobile sensor nodes exists in the network. Next, we describe two schemes aim to reposition and organize mobile sensors in response to the event distribution in the environment. Lastly, we present the concept of dynamic coverage, which is useful in applications where not every part of the terrain is needed to be covered at all times, instead over a period of time the whole terrain needs to be swept at least once.

Based on Potential Fields and Virtual forces

In [25], a potential field-based deployment technique using mobile robots is proposed, while in [24], the scheme is augmented so that every node has at least k neighbors. The potential field technique using mobile robots is first introduced in [23]. The idea of potential field is that every node is subjected to a force $\vec{F} = -\nabla U$ that is a gradient of a scalar potential field U. Each node is subjected to two kinds of forces: a repulsive force $\vec{F_c}$ that causes the nodes to repel each other to increase the coverage and an attractive force $\vec{F_d}$ that constrains the neighboring degree for nodes by making them attract towards each other when they are on the verge of being disconnected. By using a combination of these forces each node maximizes its coverage while maintaining a degree of at least k neighbors. The forces are modeled as inversely proportional to the square of inter-node distances thus $r(i,j)$ is the Euclidean distance between two nodes s_i and s_j, r_c is the communication range and $\hat{n}_{ij}$ represents the unit vector along the line joining the two nodes. Using this notation then $\vec{F_c}(i,j)$ and $\vec{F_d}(i,j)$ can be expressed as:

$$\begin{aligned}
\vec{F_c}(i,j) &= \tfrac{-k}{r(i,j)^2}\ \hat{n}_{ij} \\
\vec{F_d}(i,j) &= \begin{cases} \tfrac{-k}{[r(i,j)-r_c]^2}\ \hat{n}_{ij}, & for\ critical\ connection \\ 0, & otherwise \end{cases}
\end{aligned} \tag{2.7}$$

In the initial configuration all the nodes are accumulated in one place, possibly at the center of the sensing field, and therefore, assuming the total number of nodes is larger than k, each node has at least k neighbors. Then, they start repelling each other using $\vec{F_c}(i,j)$ until each node has only k neighbors left. Each node continues to repel all its neighbors using $\vec{F_c}(i,j)$, but as the distance between a node and its critical k neighbors increases, $\vec{F_c}(i,j)$ decreases and $\vec{F_d}(i,j)$ increases. Finally, at some distance,

ηr_c , where $0 < \eta < 1$, the total net force $\vec{F_c}(i,j) + \vec{F_d}(i,j)$ becomes zero and the nodes reach an equilibrium.

Similar to the potential field-based approach, a sensor deployment technique based on virtual forces is proposed in [26] and [28] to increase the area coverage after an initial random deployment. In this model, each node s_i is subjected to three kinds of forces: 1) a repulsive force $\vec{F_o}(i)$ exerted by obstacles, 2) an attractive force $\vec{F_a}(i)$ exerted by areas of preferential coverage (i.e. sensitive areas where a high degree of coverage is required), and 3) an attractive or repulsive force $\vec{F_s}(i,j)$, by another node s_j depending on its distance from s_i. A threshold distance d_{th} is defined between two nodes to control how close they can get to each other. Once the nodes are randomly deployed in the sensing field, the algorithm calculates the total coverage as defined by Eq.11.2. Then it calculates the net virtual force exerted on each sensor s_i by all other sensors, obstacles, and preferential coverage areas. Depending on the net force, new locations are calculated by a cluster head and an one-time movement is performed by the nodes to their designated locations. However, for relocating nodes the algorithm does not provide any route plan to avoid collision.

In [34] and [25], an incremental and greedy self-deployment algorithm is presented for mobile sensor networks in which nodes are deployed one at-a-time into an unknown environment. Each node makes use of the information gathered by previously deployed nodes to determine its target location, while satisfying the line-of-sight constraint. The authors in [25] proposed a potential field based algorithm in which nodes are treated as virtual particles subjected to virtual force. Virtual forces repel the nodes from each other and from obstacles, and ensure that the initial configuration of nodes quickly spreads out to maximize coverage area.

Based on Voronoi Diagrams

In [35], three distributed self-deployment algorithms known as VEC, VOR, and MiniMax are proposed for mobile sensor networks that exploit the structure of Voronoi diagrams. As noted before, a Voronoi diagram consists of Voronoi polygons with the property that all points inside a polygon are closest to the node that lies within the polygon. The common strategy in all these three algorithms is that once the Voronoi polygons are constructed, each node within its polygon finds out the existence of possible holes and relocates itself to a new position in order to reduce or eliminate the coverage holes.

The vector-based algorithm, VEC, pushes nodes away from densely covered areas to sparsely covered areas. Two nodes exert a repulsive force when they are too close to each other. If d_{avg} is the average distance between any two nodes when they are evenly distributed in the sensing field, the virtual force between two nodes s_i and s_j will move each of them a $[d_{avg} - r(i,j)]/2$ distance away from each other. However, if one of the node's sensing range completely covers its Voronoi polygon, then only the other node moves away a distance $[d_{avg} - r(i,j)]$. In addition to the mutual repulsive forces between nodes, the boundaries also exert forces to push nodes that are too close to the boundary inside the sensing field. However, each node before moving to the new position, it calculates whether its movement would increase the local coverage within its Voronoi polygon (calculated by the intersection of the Voronoi polygon and the sensing circle). If the local coverage is not increased, the sensor instead of moving to the target location, it moves to the midpoint position between its target location and the current location if the local coverage is increased at the new target location, otherwise, it will stay at its current position. This strategy is named by the authors as the movement adjustment scheme.

VOR is a greedy strategy that pulls nodes towards the locations of their local maximum coverage holes. If a node detects a coverage hole within its Voronoi polygon, it will move towards its farthest Voronoi vertex, such that the distance from its new location to its farthest Voronoi vertex is equal to the sensing radius. However, the maximum moving distance for a node is limited to at most half the communication radius, because the local view of the Voronoi polygon might be incorrect due to limited communication range. VOR also applies the movement adjustment scheme as in VEC, and additionally applies an oscillation control scheme that limits a node's movement to opposite directions in consecutive rounds.

The MiniMax algorithm is very similar to VOR; it moves a node inside its Voronoi polygon, such that, the distance from its farthest Voronoi vertex is minimized. Since moving a node to its farthest Voronoi vertex might lead to a situation that the vertex which was originally close now becomes a new farthest vertex, the algorithm positions each node such that no vertex is too far away from the node. It defines the concept of a Minimax circle, the center of which is the new targeted position. The Minimax point is the center of the smallest circle that enclose all the Voronoi vertices and can be calculated by the algorithms described in [36].

Between Minimax and VOR, Minimax needs more rounds to terminate and has higher message complexity but it usually results in better coverage performance.

Based on Combination of Static and Mobile Sensor Nodes

The algorithms described in the previous sections apply to networks where all the nodes are capable of moving around. However, there is a high cost associated to make each node mobile, a balance can be achieved by using a combination of static and mobile nodes, usually referred to as *hybrid or mixed sensor networks.*

In [11], authors describe a protocol, called the bidding protocol. Their protocol considers a random initial deployment, where static sensors detect their local coverage holes based on Voronoi diagrams. The mobile sensors calculate coverage holes formed at their current position if they decide to leave their current position. The static sensors bid mobile sensors based on the size of their detected coverage hole. A mobile sensor compares the bids and decides to move if the highest bid received has a coverage hole size greater than the new hole generated in its original location due to its movement. The bids are broadcasted locally up to two hops and the static sensors are able to direct neighboring eligible mobile sensors to a point close to the farthest vertex of their

Voronoi polygon.

In [12] authors address the problem of enhancing coverage in a mixed sensor network by dynamically estimate the number of additional mobile nodes required to improve coverage. Initially a fixed number of static nodes are deployed and they deterministically find out the exact amount of coverage holes existing in the entire network using the structure of Voronoi diagrams. Then dynamically estimate the additional number of mobile nodes needed to be deployed and relocated to the optimal locations of the holes to maximize overall coverage. The uncovered area within a Voronoi cell is computed using the triangles formed by the node and its two adjacent Voronoi vertices. For example see the triangle $\Delta(s_1, v_3, v_4)$ in Fig. 2.1. The line $I(v_3, v_4)$ is the perpendicular bisector of the line $I(s_1, s_4)$ and the area of $\Delta(s_1, v_3, v_4)$ can be computed as $1/4.r(v_3, v_4).r(s_1, s_4)$. The area of the Voronoi cell for a node is the sum of the area of all triangles contained within the Voronoi cell. However, the exact area of the uncovered portion of a Voronoi cell might not be equal to the area of this Voronoi cell minus the area of the sensing disk as the sensing disk of a sensor node may protrude its Voronoi cell (e.g see s_1 sensing disk in Fig. 2.1). The protrusion depends on the relations between the sensing range and the distance between two Voronoi neighbors and the lengths of the Voronoi triangle sides. In [12] a complicated calculation for the exact area of uncovered portion within a Voronoi cell is provided. Finally, for each Voronoi vertex, one mobile node should be used to cover the coverage hole around this voronoi vertex, if the size of the hole within the voronoi cell is larger than a threshold. The target location that the mobile node should be placed, lies on the line that bisects the angle formed by the voronoi vertex and the adjacent voronoi edges and it is inside the voronoi cell.

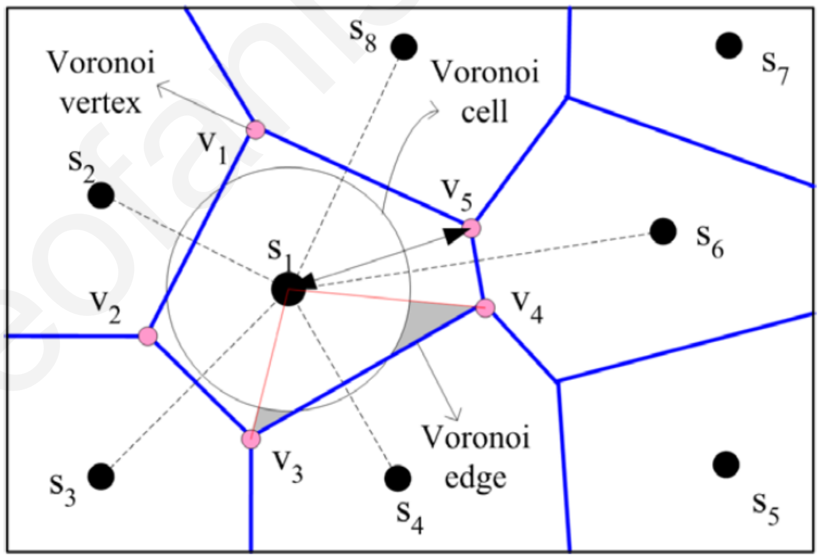


Figure 2.1: Detect and find the area of a coverage hole using Voronoi diagram

In [8] authors proposed the Grid Scan algorithm to find the maximum blind region in order to deploy additional static or mobile sensors. The proposed scheme is a multi-step scheme where each step is a greedy exploration process over all potential redeployment points. Specifically, the sensor field area is divided into uniform-sized squares (Grid). The coverage of each grid square is estimated by checking whether its center point is covered by a predefined number of sensors. Clearly, the true coverage rate of each square can be approximated when the grid size is sufficiently small. Next a greedy re-deployment scheme take place where at each step a new additional sensor node is deployed at the center of the grid square with maximum number of neighboring non-covered grid squares such that the most non-covered grid squares can be covered by the new sensor node. The main drawback of Grid Scan scheme is that the entire sensor field has to be divided into a huge number of small grid squares leading to extremely high computational cost.

Based on Event Distribution

In this section, we explore the mobility strategies in order to reposition and organize mobile sensors in response to events in the environment and thus improving the performance of event coverage.

In [21], the authors propose a distributive coverage control scheme based on a probabilistic sensing range model to maximize the joint detection probability and minimize the communication cost. They consider an event density function $R(x)$ to represent the frequency of random events taking place over the mission space Ω (sensor field area) and develop an optimization problem that aims at maximizing coverage using sensors with limited ranges, while minimizing communication cost. Starting with initial sensor positions, the authors develop a gradient algorithm to converge to a (local) solution to the optimization problem. The sequence of sensor distributions along the solution is seen as a discrete time trajectory of the mobile sensor network until it converges to the local minimum. Specifically, they consider a simple probabilistic sensing model (eq. 2.2 where $r_u = 0$ and $\alpha = 1$) given by $p_i(x) = e^{-\|x-s_i\|}$, if $\|x - s_i\| \leq r_s$ and they define the concept of joint event detection probability as given by equation below.

$$P(x, s) = 1 - \prod_{i=1}^{N} [1 - p_i(x)] \tag{2.8}$$

In other words the joint event detection probability is another definition of the area coverage defined in paragraph 2.3.1) and gives the probability that an event, taking place at x is detected by a mobile sensor.

The optimal coverage problem is formulated as a maximization of the expected event detection frequency by the sensor nodes over the mission space Ω and expressed by $\max_{s} (F(s))$ where

$$F(s) = \int_{\Omega} R(x)P(x, s)dx \tag{2.9}$$

In this optimization problem, the controllable variables are the locations of mobile sensors in the vector s. The authors develop a distributed method to solve this optimization problem locally via the partial derivatives of the cost function $F(s)$. The partial derivatives with respect to s_i are given by

$$\frac{\partial F}{\partial s_i} = \int\limits_{\Omega} R(x) \frac{\partial P(x,s)}{\partial s_i} dx \tag{2.10}$$

This partial derivative can be evaluated locally by each mobile node s_i and then a gradient method can be applied to direct nodes towards locations that maximize $F(s)$. The next waypoint of the mobile sensor s_i motion trajectory is given by

$$s_i(k+1) = s_i(k) + \alpha \frac{\partial F}{\partial s_i(k)} \tag{2.11}$$

where k is an iteration index and α is the step size. Finally the authors also introduced a communication cost function into the coverage control problem, viewing the sensor network as a multi-source, single-base station data collection network to control the mobile nodes such that the communication cost is minimized.

In [13] the objective is to dispatch more mobile senors to event locations, while still maintaining complete coverage of the sensor field. In order to approximate event distributions by mobile nodes, two moving strategies are proposed, namely, the history-free strategy and the history-based strategy. In the history-free strategy, each sensor reacts to an event by moving according to a function of the form $s_i(k+1) = s_i(k) + f(d)$ where $s_i(k)$ is the position of sensor i at step k and d is the distance between a sensor and an event. An example of the function f is $f(d) = \alpha d^{\beta} e^{-\gamma d}$ where the values of parameters α, β, γ must satisfy the inequality $\alpha e^{-\gamma d}(\beta d^{\beta-1} - \gamma d^{\beta}) > 1,\ \forall d$. In the history-based strategy, each sensor needs to maintain event history to improve the sensor's approximation of the event distribution. Each sensor maintains a discrete version of the cumulative distribution function (CDF) of the events, which is updated after each event and used to compute its next position. The above two strategies move sensors closer to event locations in order to maintain complete coverage of the sensor field, the authors also use a Voronoi diagram to determine whether the movement of a sensor will cause a coverage hole by checking whether if any sensor's Voronoi vertex is farther away than the sensing range. In this case, the sensor should stop moving to maintain its original coverage.

The deployment strategies discussed so far strive to relocate, spread or add additional sensors to an initial static configuration in order to maximize coverage. The main difference among these algorithms is how exactly the new positions are computed. However, one drawback is that the final network configuration is again static, and therefore, parts of the sensing field that are still uncovered even after the relocations will remain so. As a consequence, an intruder moving along those uncovered regions will never get detected. In addition, static sensor networks are also not able to cope with dynamic environments where new obstructions may appear after the initial deployment. To overcome these drawbacks, the concept of dynamic coverage is introduced as described below.

Dynamic Coverage based on Continuous Movement of Mobile Sensors

The concept of dynamic coverage is relatively new and provides a new perspective on looking at coverage that uses node mobility to patrol the environment in order to provide better quality of coverage and detection capability. In practice, most of the applications would not require 100% coverage of a region at all times, rather it is more likely that they would require the region to be covered at least once during a time interval. Thus, instead of coming up with the optimal deployment strategy of nodes that guarantees complete coverage of a region at all times, it is more useful to devise a strategy that could provide complete coverage over a period of time. Coverage provided by mobile sensors depends on the velocity, mobility patterns, number of sensors deployed, and the dynamics of the phenomenon being sensed [37]. It must be noted that although a mobile sensor is able to cover more area than a stationary sensor over a period of time, the instantaneous area covered by both are the same. Hence, proper motion planning is required to exploit the full advantage of mobile sensors.

In [20], the authors investigate the area coverage resulting from the continuous random movement of mobile sensors during a time interval $[0, t)$ and they also investigate the detection time of an intruder in such sensor networks. Assuming that the static distribution of nodes in the two-dimensional plane is a Poisson point process with density λ and the sensing model is a binary disk of radius r_s, the fraction of the region covered by at least one sensor at time t as mention in paragraph 2.3.1 is given by $C_s = 1 - e^{-\lambda \pi r_s^2}$. However, if the nodes move around in the sensing field following a random mobility model, then the fractional area coverage during a time interval $[0, t)$ is now given by $C_m = 1 - e^{-\lambda(\pi r_s^2 + 2 r_s E(v) t)}$ where $E[v]$ represents the expected sensor speed (see Fig.2.2). The detection time x of a randomly located stationary target, is

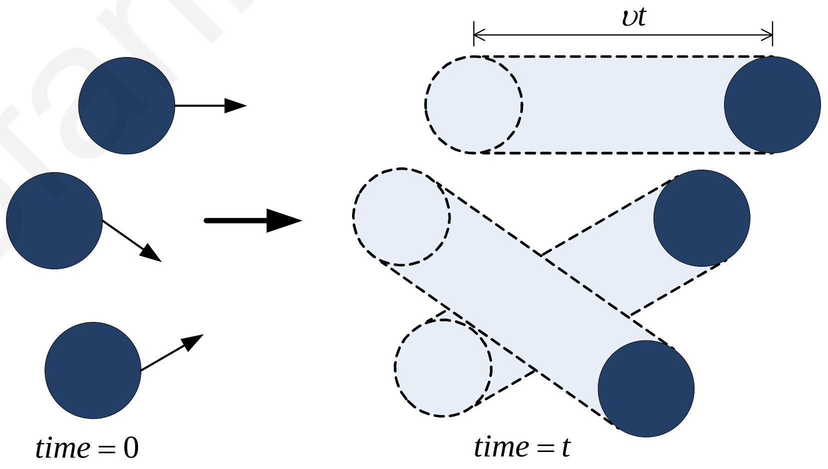


Figure 2.2: Coverage due to random movement of mobile sensors

defined to be the time at which the target first enters the sensing area of a sensor (i.e. time for first detection). The authors prove that when mobile sensors move according to a random mobility model with fixed speed v, the detection time for a stationary target follows an exponential distribution $x \sim exp(2\lambda r_s v)$. In the case of a moving target with speed v_t and direction θ_t, authors prove that the detection time also follows an exponential distribution with mean $1/2\lambda r_s \bar{v}$, where $\bar{v}$ is the effective sensor speed relative to the target speed. We observe that the detection times of both stationary and moving targets follow exponential distributions with parameters of the same form. Thus, minimizing the detection time corresponds to maximizing the effective sensor speed. For example the movement of the target in any direction will result in a higher effective speed with respect to the sensors, since the sensors move in all directions with equal probability and therefore the target will be detected faster, on the other hand if the target remains stationary the first hit time will become larger.

2.4 Path Planning Algorithms for Mobile Robotics

Path planning is one of the most fundamental problems in mobile robotic agents. The problem of planning a motion path for mobile robots can be defined as finding a path between a starting point and the destination in an environment with obstacles such that no collisions with obstacles occur and the path is optimal with respect to some particular measure such as shortest path length, least energy consumption or minimum traveling time, etc. Several deterministic path planning approaches - Road Map, Cell Decomposition and Potential Fields methods, are broadly surveyed in [38, 16, 39]. Due to the NP-Hardness of the deterministic approaches, heuristic methods have outperformed deterministic approaches and have gained wide popularity [40, 41]. Path planning methods can also categorized as either being global or local based on the robot's knowledge about the environment. A global method assumes that the environment is completely known before the mobile agent begins its traverse while a local one assumes partial knowledge of the environment.

2.4.1 Deterministic Path Planning Methods

In mobile robotics the most common approach is to assume, for path-planning purposes, that the robot is holonomic or even that the robot is simply a point. Thus, the configuration space for a mobile robot is reduced to 2D space. Using this simplified configuration space, we introduce below the common deterministic techniques for mobile robot path planning. The first step of any path-planning algorithm is to transform the continuous environmental model into a discrete map. Deterministic path planning approaches differ as to how they effect this discrete decomposition. Given a way of describing the free space, the path planning problem reduces to a graph-searching problem for finding a connected sequence of feasible configurations between the start and goal from the representation. Several graph-searching methods [39] developed in artificial intelligence [42] can be used as Dijkstra's shortest-path algorithm, depth-first, breadth-first, best-first, A* search, random-search and others.

Road Map Path Planning

Road map approaches capture the connectivity of the robot's free space in a network of 1D curves or lines. Once a road map is constructed, it is used as a network of road (path) segments for robot motion planning. Path planning is thus reduced to connecting the initial and goal positions of the robot to the road network, then searching for a series of roads from the initial robot position to its goal position. The road map is a decomposition of the robot's configuration space based specifically on obstacle geometry. The challenge is to construct a set of roads that together enable the robot to go anywhere in its free space, while minimizing the number of total roads. Two well-known road map approaches are the Visibility graph and Voronoi diagram. In the visibility graph, graph edges joining all pairs of obstacle vertices that can see each other including both the initial and goal positions as vertices, thus roads come as close as possible to obstacles and resulting solutions have minimum-length paths (see Fig. 2.3) In the case of the Voronoi diagram, roads stay as far away as possible from obstacles.

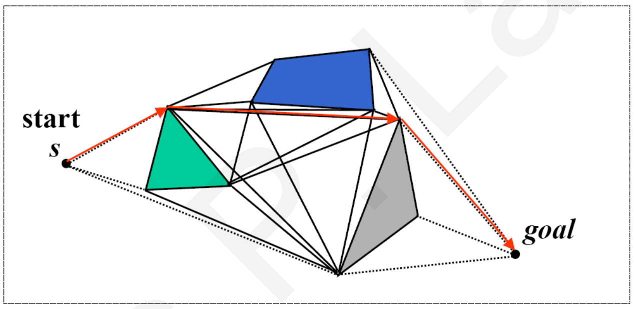


Figure 2.3: An example of Visibility graph Path Planning.

Cell Decomposition Path Planning

The idea behind cell decomposition is to discriminate between geometric areas, or cells, that are free and areas that are occupied by objects. The free space is decomposed into a set of simple connected cells, then the adjacency relationships among the cells are computed and a "connectivity graph" is constructed. A collision-free path between the start and the goal configuration is found by first identifying the two cells containing the start and the goal points and then connecting them with a sequence of connected cells (e.g. passing through the midpoints of the cell boundaries). An important aspect of cell decomposition methods is the placement of the boundaries between cells. If the boundaries are placed as a function of the structure of the environment, such that the decomposition is lossless, then the method is termed exact cell decomposition (see Fig. 2.4). If the decomposition results in an approximation of the actual map, the system is termed approximate cell decomposition (e.g. grid-based decompositions with fixed cell size).

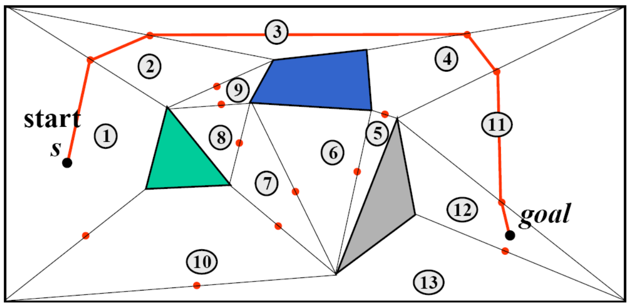


Figure 2.4: An example of Exact Cell Decomposition Path Planning.

Potential Field Path Planning

The potential field concept was first introduced by Oussama Khatib [23]. The potential field method treats the robot as a particle moving under the influence of an artificial potential field U. The robot moves by following the downhill gradient of the artificial potential field, just as a ball would roll downhill. The goal acts as an attractive force on the robot and the obstacles act as repulsive forces. The superposition of all forces is applied to the robot and smoothly guides the robot toward the goal while simultaneously avoiding known obstacles. It is important to note, though, that this is more than just path planning. The resulting force is also a control law for the robot. The artificial force acting at the position of robot is given by $\vec{F} = -\nabla U$. The potential field method can be very efficient and under ideal conditions, the robot is smoothly attracted toward the goal while being repulsed away from the obstacles (see Fig. 2.5). However there is the risk of getting stuck in local minima or oscillate between the two closest points, which obviously sacrifice completeness.

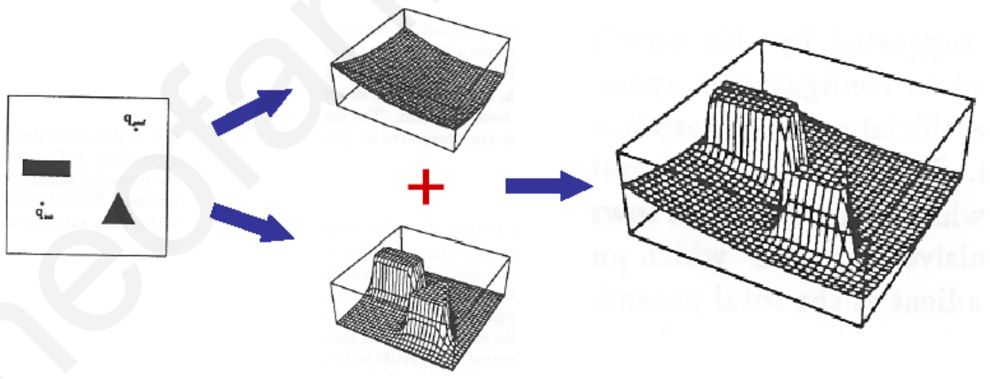


Figure 2.5: An example of Potential Field Path Planning.

To overcome the problem of local minima several techniques have been proposed when planning with potential functions. One approach is the wave-front planner [43] which affords the simplest solution to the local minima problem, but can only be implemented in spaces that are represented as grids. For example, consider a two-dimensional space. Initially, the planner starts with the standard binary grid of zeros corresponding to free space and ones to obstacles. The planner also knows the grid cell locations of the start and goal. The goal grid cell is labeled with a two. In the first step, all zero-valued cells neighboring the goal are labeled with a three. Next, all zero-valued cells adjacent to threes are labeled with four, etc. This procedure terminates when the wave front reaches the cell that contains the robot start location. The planner then determines a path via gradient descent on the grid starting from the start. The wave-front planner essentially forms a potential function on the grid which has one local minimum and thus is resolution complete. The major drawback of this method is that the planner has to search the entire space for a path and thus becomes computationally intractable for large configuration spaces.

Another technique to solve the local minima problem is a special function which has the only minimum at the goal position. This function is called *navigation function* and formally defined in [44, 45]. The construction of navigation function resides on algebraic topology and is based on the assumption that the obstacles are disks with finite radii placed in such a fashion that none intersect. It is also assumed that the configuration space is bounded by a sphere (or a star) space. Given the above assumptions an “almost” global navigation function could be constructed using a steepening parameter. Increasing the steepening parameter causes the other critical points gravitating toward the goal and local minima turning into saddles. Unfortunately, this steepening effect has an adverse consequence as sometimes the navigation function becomes flat near the goal and far away from the goal and thus has sharp transitions in between, which makes the implementation of gradient descent approach quite difficult because of numerical errors. It is worth mentioning that the navigation function methodology has been also extended to the case of multiple robots systems [46].

2.4.2 Sampling-based Path Planning Methods

The aforementioned Deterministic methods suffer from many drawbacks, such as high computation complexity and trapping in local minima, which makes them impractical. In order to improve the efficiency of classic methods, sampling-based algorithms [47] have been developed, including Probabilistic Road Maps [48] and Rapidly-exploring Random Trees [49], with major advantage the high-speed implementation. However, these algorithms depend on having a complete and accurate knowledge of the environment. In sampling-based or randomized path planning algorithms information on the configuration space is acquired by generating samples and edges between them and store them in a suitable data structure. Sampling-based path planning algorithms can be separated between probabilistic roadmap-based planners and tree-based planners. Below we introduce the basic ideas in the aforementioned two categories of those algorithms.

Probabilistic Road Maps

The Probabilistic Road Map (PRM) algorithm works in two steps, the first called learning stage and the second called query stage. In the learning stage the algorithm samples the configuration space and builds an undirected graph $G = (V, E)$ which captures the information gathered. The graph is called probabilistic road map. In the query stage, the Probabilistic Road Map is used to solve specific motion planning instances which are then reduced to a graph search. In learning stage samples randomly generated over the configuration space are accepted if they belong to the free space and discarded otherwise. When a sample is accepted, it becomes a vertex of the G graph. After a vertex is added, the algorithm checks if it is possible to add edges between the inserted vertex and vertices already in the graph. For this aim a subset of neighboring vertices are selected and the new vertex is checked whether it can be connected to them. Two vertices are neighbors if their distance is less than a fixed threshold. The algorithm simply connects the two points with a straight segment and verifies if it lies in free space or not. This is done by selecting a set of intermediate points along the segment and check whether all the intermediate points lie in free space. If the resulting graph contains more than one connected component (the graph is not fully connected), an improving stage is performed in order to merging them. In the query stage a start and goal points are given and the algorithm is required to produce a path between them using the graph G. The algorithm tries to connect the two points to vertices in the graph G by using the same technique used to insert edges in the road map and then the graph is searched to find the path between the start and goal points. Figure 2.6 shows an example of Probabilistic Road Map.

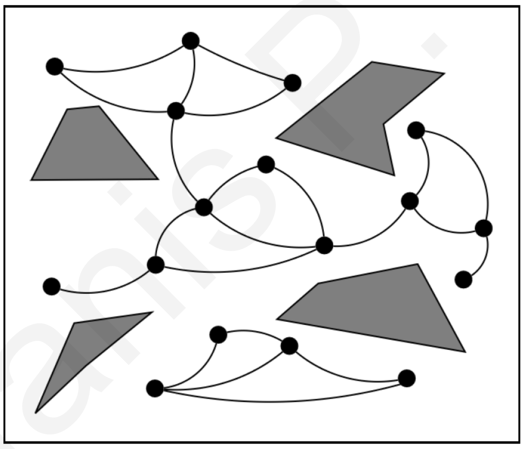


Figure 2.6: An example of Probabilistic Road Map.

Rapidly-exploring Random Trees

The Rapidly-exploring Random Trees approach has become the most popular single-query motion planner in the last years especially for problems involving nonholonomic planning. The Rapidly-exploring Random Tree algorithm works by growing a tree starting from a given root (starting point). The growth is performed one vertex at a time, by alternating the two steps, selection and propagation, that are common to most tree-based planners. In the selection step, a new sample is chosen uniformly at random and the tree is searched to find the nearest neighbor among the samples already existing in the tree. In the propagation step an edge is then extended from the selected nearest neighbor toward the new sample, not necessarily reaching it. The ending vertex from the edge extended from nearest neighbor is then the new sample added to the tree. If a start and goal points are given a significant speedup is obtained by growing two trees, one from start and the other from goal. Figure 2.7 shows an example of a Rapidly-exploring Random Tree.

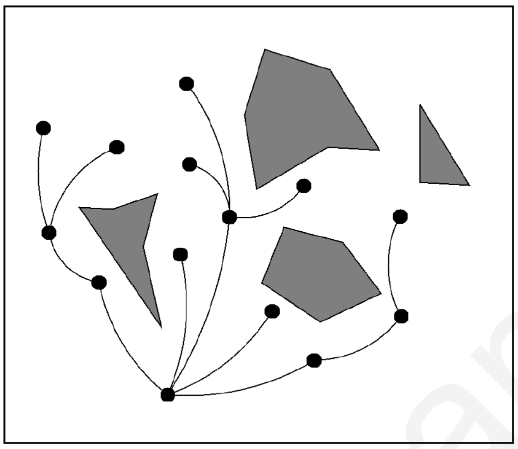


Figure 2.7: An example of Rapidly-exploring Random Tree.

2.4.3 Heuristic Path Planning Methods

Also other approaches for robot path planning include Artificial Neural Networks, Genetic Algorithms, Particle Swarm Optimization, Tabu Search, Simulated Annealing, Stigmergy, Wavelet Theory etc. Heuristic algorithms do not guarantee to find a solution, but if they do, are likely to do so much faster than deterministic methods. Below we outline some heuristic approaches proposed for mobile robot path planning.

A novel biologically-inspired general neural network approach for real-time collision-free path planning in a dynamic environment is presented in [50]. The state space of the neural network is the configuration space of the robot, and the dynamic environment is represented by the dynamic activity landscape of the neural network. The target globally attracts the robot in whole state space, while the obstacles locally push the robot away to avoid collisions. In [51] another neural network approach to path planning for two dimensional robot motion is developed and in [52] a neural network approach for dynamic task assignment of multiple robots is presented.

The idea of using a genetic algorithm for robot path planning was used in [53] for generation of collision-free paths. In [54] a multiple path planning method for a group of mobile robots in a 2D environment using genetic algorithms is presented.

Particle swarm optimization is inspired by the ability of flocks of birds, schools of fish, and herds of animals to adapt to their environment, find rich sources of food, and avoid predators by implementing an information sharing approach. An algorithm for mobile robot path planning using particle swarm optimization for obstacle avoidance in dynamic environment is presented in [55]. Also, an algorithm is developed in [56] for robot path planning using particle swarm optimization of Ferguson Splines. Finally, obstacle avoidance path planning for soccer robots using particle swarm optimization has been extended in [57].

Tabu search approach can be used in robot path planning. This algorithm does not suffer from the local minimum problem. In [58] an online motion planner is developed to govern the movements of mobile robots during their explorations. By using the tabu search, a set of tabu (i.e. forbidden) moves are defined at each iteration of the search to confine the robot's navigable locations, and guide it toward the goal. Based on range-sensor readings and the cost function value defined for each ray, the robot is attracted to certain obstacle vertices, and moves along a path consisted of lines connecting the vertices of different obstacles. The planner also takes advantage of random moves when trapped in dead-ends.

2.4.4 Coverage Path Planning

The Coverage path planning problem can be defined as finding a motion path for mobile robots that guarantees the complete coverage of an environment. Unlike conventional path planning, coverage path planning enables applications such floor cleaning, lawn mowing, harvesting, mine hunting, painting, searching, contamination cleanup etc [59]. Recent commercial implementations in consumer products include automatic vacuum cleaners [60] and automatic lawn mowers [61].

An early paper in 1988 by Cao et al [62] defines the criteria for the region filling operation or coverage by a mobile robot as follows:

1. Robot must cover (move through) the entire area
2. Robot must cover the region without overlapping path
3. Continuous and sequential operation without any repetition of paths is required
4. Robot must avoid all obstacles
5. Simple motion trajectories (e.g. straight lines or circles) should be used (for simplicity in control)
6. An "optimal" path is desired under available conditions.

However as mention in [62] it is not always possible to satisfy all these criteria for complex environments and thus sometimes a priority consideration is required.

In the past years the coverage problem has received much attention and several solutions have been proposed in the literature including single and multi-robot solutions, a survey on recent results in coverage for robotics is provided in [59]. Many algorithms either implicitly or explicitly use cellular decomposition of the free space to achieve coverage. In cellular decomposition, the free space is broken into simple regions, which should guarantee the coverage. The cellular decomposition algorithms can be classified into three classes: approximate, semi-approximate and exact [59]. Another other way to classify the algorithms is off-line and on-line. Off-line algorithms rely only on stationary information and the environment is assumed to be known. Usually on-line algorithms are employed if some kind of adaptivity to the environment is needed. On-line algorithms use real-time sensor measurements for robots guidance and they can also be called sensor-based coverage algorithms.

In approximate cellular decomposition, the region is discretized using a grid, which approximately covers the region, and the algorithm is applied to the grid. Authors in [63] present an algorithm that solves the coverage path planning problem in a grid utilizing a distance transform algorithm named wavefront algorithm. The wavefront algorithm initially assigns a 0 value to the goal and then a 1 to all surrounding cells. Then, all unmarked cells neighboring the marked 1 are then labeled with a 2. This process repeats until the wavefront crosses the start. Once this occurs, the robot can use gradient descent on this numeric potential field to find a path. In [63] after the wavefront algorithm has spread throughout the entire free space, the robot finds a path from start to goal by moving from its current cell towards a not visited neighboring cell with the highest value. In an environment with no obstacles, this approach reduces to following the equipotential curves from top to bottom. Authors also define a wavefront potential function to encode path safety by using an obstacle distance transform to compute distance of each cell to the nearest obstacle. The new path planning algorithm uses a weighted sum of both potentials to compute the coverage path, resulting in a path with fewer turns which is beneficial for mobile robots. The main problem in this algorithm is that it does not count kinematic constraints. Another approximate cellular decomposition coverage path planning algorithm is proposed in [64]. The algorithm, called Spanning Tree Covering, subdivides the work-area into disjoint cells and then follows a spanning tree of the graph induced by the cells, while covering every point precisely once.

In exact cellular decomposition, the workspace is decomposed into a collection of non overlapping cells, and then the robot searches the connectivity graph that represents the adjacency relation among cells. Thus the complete coverage can be achieved by back and forth robot motions. In [65] a novel boustrophedon cellular decomposition approach is proposed. The algorithm allows the robot to cover each cell like the way the ox drags a plough. This solution combines the advantages of cell decomposition and template based approaches, and minimizes the number of cells used in cell decomposition. It requires the prior knowledge of the obstacle locations and the critical points.

The coverage path planning problem has been also approached by many researchers using random strategies as these approaches were suitable for inexpensive robots because of their low sensor (localization is not needed) and computational power requirements. Random coverage strategies do not guarantee completeness, however Gage [66] showed that the percentage of the covered area can be increased by either using multiple robots or sweep the area for long enough time. In [67], it has been shown that the efficiency of robots with random algorithms is approximately 20% of those that use more advanced methods. Therefore if a robot with a random algorithm can be constructed at 1/5 of the price of a robot with localization and advanced path planning, it may be effective to use a randomized search. Recently, a lawn mowing robot [61] that uses random zigzag motions for coverage was reported in Consumer Reports 2008.

In [68] authors focus on the coverage problem of efficiently coordinate a mobile robot with uncertain heading to cover a specified area. Five control strategies based on event-triggered position measurements when the vehicle intersects the boundary of the area to be covered are compared. They are denoted Receding Horizon, Robust Receding Horizon, Opportunistic Receding Horizon, boustrophedon and randomized. The proposed path-planning strategies are evaluated by comparing the number of turns for various degrees of heading uncertainty. It is shown that for large uncertainties, a randomized strategy is the best one, which is intuitive since the system state does not reveal much information in that case. For small uncertainties, a boustrophedon-path strategy sweeping the area by a simple back-and-forth motion is sufficient.

There are also several studies on coverage path planning of multi robot systems. The authors in [69] presented a novel neural network approach to solve coverage path planning problem for non stationary environments. Complete coverage paths are generated from a dynamic activity landscape of the neural network and the previous robot location. The dynamics of each neuron in the topologically organized neural network is characterized by a shunting equation derived from Hodgkin and Huxley's membrane equation for a biological neural system. In [70] an off-line multi-robot coverage strategy is proposed which is suitable for covering unstructured environments using a Voronoi diagram and a cost function to optimize the collective coverage task. Finally, in [71] authors proposed an efficient boustrophedon multi-robot coverage algorithm. The algorithm use the same planar cell based decomposition as the Boustrophedon single robot coverage algorithm, but provide extensions to handle how robots cover a single cell, and how robots are allocated among cells based on the type of communication that exists between the robots.

2.5 Cooperative Path Planning and Control for Multi-Robot Systems

In the previous section, different kinds of path planning algorithms for mobile robots have been presented, however the majority of research effort has focused on centralized path planning approaches for single autonomous robots and nothing has been said about how to control a group of cooperating autonomous mobile robots to achieve different missions such as search, mapping surveillance, exploration, target detection and tracking. Moreover, most of algorithms presented previously, do not consider the maneuverability constraints of robotic vehicles and the dynamic nature of the environment.

In the recent years (early 2000s), several research efforts have been directed toward the cooperative path planning and control of a team of autonomous mobile robots performing cooperative tasks. Cooperative or collaborative indicates the interaction among multiple robots, that means, the robots have to communicate, exchange information or interact in some way to achieve an overall common mission.

This section review various methods and control approaches dealing with multi-robot systems. A multi-robot system or multi-vehicle system can be composed of unmanned aerial, ground or underwater vehicles (UAVs,UGVs or UUVs), however a significant amount of research in this area is performed under the framework of UAVs. A survey of recent research in cooperative control of multi-vehicle systems regarding current applications and technical approaches for formation control, cooperative tasking, spatio-temporal planning and consensus algorithms is provided in [72].

2.5.1 Cooperative Search and Surveillance

An important problem in unmanned air vehicle (UAV) and UAV-mounted sensor control is the target search problem in order to locate target(s) in minimum time. This task is also referred as the cooperative surveillance problem using a collection of autonomous vehicles moving in a way that maximizes the probability of finding the target(s). Theoretical work on optimal searching for targets in unknown location was initiated by B. Koopman during the World War II to find enemy marine vessels for the U. S. Navy. Classical search theory as we know it today is based on work by Koopman [73] and later work by L. Stone [74] who especially study the moving target problem.

Polycarpou et al. [17, 75] developed a general framework for directing a group of UAVs to cooperatively search a dynamic and uncertain environment. The search path generation problem is separated into two parts: the on-line environment modeling process and a real-time path decision process. The coordination among vehicles is achieved by considering the group benefit in each vehicle's decisions. Each UAV maintains a knowledge base about the environment (termed a search map) where assigns an uncertainty measure to each point of the search region. As each UAV moves about, it continually refines its understanding of the environment by collecting information from its own sensors and periodically exchanging search maps with other UAVs. Each UAV uses its search map to dynamically calculate and update its search path to meet certain criteria, e.g., minimizing the uncertainty in the environment. Each UAV maneuvers autonomously based on its understanding of the environment (which in turn is formed in part based on information provided by other UAVs). Path planning is accomplished in a receding-horizon framework where a multi-objective cost function J, weighing the different competing subgoals of the agents, is optimized at each time step, over some planning horizon. In general, the cost function J can be written as $J = \omega_1 J_1 + \omega_2 J_2 + \ldots + \omega_s J_s$ where J_i represents the cost criterion associated with the i-th subgoal and ω_i is the corresponding weight. The weights are normalized such that $0 \leq \omega_i \leq 1$ and the sum of all weights is equal to one, $\sum_{i=1}^{s} \omega_i = 1$. Priorities to specific subgoals is achieved by adjusting the values of weights associated with each subgoal. The agents must periodically exchange their knowledge base (as well as other data, such as course and speed in order to prevent several UAVs from duplicating efforts by choosing the same optimal path. Based on this framework, a

Bayesian map-building method to probabilistically model the environment together with an opportunistic learning method for cooperative path planning is proposed in [76] to cooperatively search for targets by a group of networked UAVs in an uncertain environment. More recently Liao et al. [77] considered a cooperative search path planning approach using a team of UAVs with limited communication and focused on information sharing and information fusing policies. They observed that under the aforementioned framework it is possible to use significantly less than global communication range and to communicate less frequently than once every time step without a serious loss of performance.

Bourgault et al. [78] presented a Bayesian approach to model the search for a stationary or drifting target at sea, principally with the objective of maximizing the probability of detection within a given a time. The search environment is discretized into a large grid of cells, over which a target probability density function is defined. This function is defined a priori with available information, and updated with a process model that accounts for wind, current and other factors. Similarly, a distance-based observation model maps the position of vehicle to the likelihood of detecting the target in each of the cells. Updating the probability distribution with this model then provides a posterior accounting for the effects of search. This cooperative search task with multiple UAVs developed by bringing together a decentralized Bayesian data fusion technique and a decentralized coordinate control scheme. Due to the large number of cells involved, the trajectories were calculated using one-step look-ahead.

Another approach to the cooperative surveillance problem proposed by Cortes et al. [79, 80]. They proposed an algorithm for deploying a group of robots over a region of interest to provide sensor coverage of the environment. The algorithm divide up the region into a set of polytopes among the robots and each robot is responsible of sensing its polytope. The sensing performance of a vehicle depends on its distance from a sense point. Then, they form the coverage control problem by choosing the locations of each vehicle in order to minimize an objective function. The proposed cost function depends on the number and current positions of available robots as well as the event distribution density function. Using a gradient decent control law, it is shown that each robot trajectory converges to the centroid of a cell in a Voronoi partition of the search domain and hence provides (locally) optimal coverage. A key element of this approach is that the only communication required is with the nearest neighbors of each vehicle, however it assumes that the collection of available vehicles is sufficient to cover the entire region of interest.

2.5.2 Formation Control and Consensus Algorithms

One of the simplest cooperative control problems is that of formation control where a set of mobile robots (vehicles or agents) move in a formation, specified by the relative locations of nearby robots. Formation control has receive a considerable attention under the framework of so-called "*swarms of vehicles*". Roughly speaking, a swarm is a large collection of vehicles that perform in a collective fashion, such as flying together in a given direction. One early work in swarm-like behavior was that of Reynolds [81], who developed a set of rules that he used to generate realistic motion of vehicles for the animation industry. An innovative approach to understanding swarm behavior was taken by Jadbabaie et al. [82] who described how to achieve coordination for groups of mobile autonomous agents using nearest neighbor rules. The control law was quite simple, making use of a simple heading model in which each agent updated its heading according to the rule

$$\theta_i(t+1) = \frac{1}{1+n_i(t)} \left(\theta_i(t) + \sum_{j \in N_i(t)} \theta_j(t) \right) \tag{2.12}$$

where $\theta_i(t)$ denotes the heading of agent i at time t, $n_i(t)$ is the number of neighbors of agent i at time t. Equation 2.12 is the average of the headings of agent i and agent i's neighbors at time t and hence this control law essentially tells each agent to steer in the same direction as its neighbors. Jadbabaie et al. are able to demonstrate that with this control law, all vehicles will converge to a common heading.

Control laws for swarms often involve using attractive and repulsive functions between nearby vehicles. In addition to rivaling force between agents proposed by Polycarpou et al. [17], another representative work is that of Olfati-Saber [83], who proposed a control input u_i consisting of three terms $u_i = {f_i}^g + {f_i}^d + {f_i}^\gamma$. The first term ${f_i}^g$ is a gradient-based term, the second term ${f_i}^d$ is based on the relative velocities of neighboring agents and acts as a damping force and third term ${f_i}^\gamma$ is a navigational feedback term that takes into account the group objective.

Finally we briefly describe the problem of "consensus" in cooperative control. The consensus problem is to have a group of vehicles or agents reach a common assessment or decision regarding a certain quality of interest that depends on the state of all agents. A Consensus algorithm is an interaction rule that specifies the information exchange between an agent and all of its neighbors on the network. More recently, there has been a growing interest among researchers in problems related to multi-agent networked systems with close ties to consensus problems. This includes subjects such as collective behavior of flocks and swarms, sensor fusion, asynchronous distributed algorithms, formation control for multi-robot systems, consensus-based belief propagation in Bayesian networks and others (see [84, 85] and references therein). Article [85] provides a tutorial overview of information consensus in multi-vehicle cooperative control where theoretical results regarding consensus-seeking under both time invariant and dynamically changing communication topologies are summarized.

Consider a network of agents interested in reaching a consensus via local communication with their neighbors. The interaction communication topology between agents can be represented using a directed graph $G = (N, E)$ where $N = \{1, 2, \ldots, n\}$ is the set of nodes and $E \subseteq N \times N$ are the edges. The neighbors of agent i are denoted by $\mathcal{N}_i = \{j \in N : a_{i,j} \neq 0\}$ where $a_{i,j} \neq 0$ denotes that agent i communicates with agent j. $A = [a_{i,j}]$ is the adjacency matrix of graph G (representing which agents communicate each other, $a_{i,j} = 1$). A dynamic graph $G(t) = (V, E(t))$ is a graph in which the set of edges $E(t)$ and the adjacency matrix $A(t)$ are time-varying. Clearly, the set of neighbors $\mathcal{N}_i(t)$ of every agent in a dynamic graph is a time-varying set as well.

The most common consensus algorithm is proposed in [84] where the behavior of each agent is governed by

$$\dot{x}_i(t) = -\sum\nolimits_{j\in N_i} a_{ij}(t)\,(x_i(t) - x_j(t)) \tag{2.13}$$

where $x_i(t)$ represents the internal or information state of the i-th agent and $a_{ij}(t)$ is the (i,j) entry of the adjacency matrix of the associated communication graph $G(t)$ at time t.

A consequence of equation 2.13 is that the information state $x_i(t)$ of vehicle i is driven toward the information states of its neighbors. The critical convergence question is, when do the information states of all of the vehicles converge to a common value $(x_1 = x_2 = \ldots = x_n)$. For the system representing by eq. 2.13, it has been shown [84] that if the information flow is bidirectional and time-invariant which means that G is a connected undirected ($a_{ij} = a_{ji}$ for all i, j) graph, the states of the individual vehicles asymptotically converge to the average of the initial state values, $\alpha = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$, α is the collective decision. Thus the consensus algorithm guarantees convergence to a collective decision via local agent interactions. If G is not bidirectional (so that there are asymmetries in the information available to each agent), then the interaction above does not necessarily lead to average consensus. Furthermore, even if the connections are changing as a function of time, it can be shown that the average consensus can be reached under certain conditions.

The consensus algorithm described above by eq. 2.13 can be written in matrix form as $\dot{\mathbf{x}}(\mathbf{t}) = -\mathbf{L}(\mathbf{t})\mathbf{x}(\mathbf{t})$ where $\mathbf{x} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]^{\mathbf{T}}$ is the information state and $\mathbf{L}(\mathbf{t}) = [\ell_{\mathbf{ij}}(\mathbf{t})] \in \mathbb{R}_{\mathbf{n}\times\mathbf{n}}$ is the graph Laplacian of the underlying communication graph G. The graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} = \mathbf{diag}(\mathbf{d_1}, \ldots, \mathbf{d_n})$ is the degree matrix of G with diagonal elements $d_i = \sum_{j\neq i} a_{ij}$ and zero off-diagonal elements and $\mathbf{A}$ is the adjacency matrix of graph G.

2.5.3 Experimental Testbeds

Experimental work with teams of cooperative mobile robots in the context of WSNs has been limited, primarily due to the cost and the implementation overhead required for the experimental validation of the algorithms developed for WSNs. Some experimental testbeds involving only cooperative autonomous robots have been proposed in the literature.

In [86], the authors develop a testbed with multiple mobile khepera robots [87] that form a team which cooperates to visit multiple target points to collect the rewards associated with them. The objective is to maximize the total reward accumulated over a given time interval. This testbed has also been used to address dynamic network deployment in the context of coverage control using only mobile nodes. Khepera robots, though ideal for such testbeds due to their small size and functionality, they are quite expensive (each wireless Khepera robot costs approximately 3000 euros). Therefore, several attempts have been made to develop small, inexpensive, modular and open-source platforms for developing large-scale mobile robot applications [88, 89, 3].

In [90] the authors develop an experimental testbed to study several issues associated with the convergence of control, communication and computation. The testbed consists of remotely controlled cars and each car is controlled by its own dedicated laptop. An overhead camera provides positioning information and communication between laptops is based on IEEE 802.11. The authors have tested many scenarios on this testbed ranging from traffic control to collision avoidance.

Another mobile robot platform that has been incorporated in a testbed is presented in [91]. In this case study authors develop a mobile sensor node based on the MICA2 board developed by Crossbow Technology. Their main objective is to create a network of mobile nodes that trace the boundaries of a diffusion contamination process.

In [92], an experimental testbed for studying multi-vehicle, networked control has been developed. The authors use ducted fans to control the vehicles and develop real-time feedback control algorithms to stabilize the system while performing cooperative tasks. Several other researchers have also demonstrated experimentally leader following cooperation using two Unmanned Aerial Vehicles (UAVs) [93, 94].

Recently, MIT's Distributed Robotics Laboratory developed an energy efficient four rotor flying robot for indoor and outdoor navigation [95]. This flying platform offers a 1 kHz control frequency and motor update rate and thus minimizes uncertainty in position control and instable behavior during maneuvers. Also researches at Utah State University have develop another low cost UAV platform [96] for aerial surveillance missions.

Another well-know experimental testbed of autonomous aerial vehicles and robots has been developed at GRASP Laboratory of the University of Pennsylvania. In [97], an experimental study is presented where a team of quadrotor robots cooperatively grasping, stabilizing, and transporting payloads of different configurations to desired positions and along three-dimensional trajectories. In addition, experimental work with Unmanned quadrotor Helicopters for attitude control under the influence of wind gusts has been presented in [98]. Also in [99], a system using Unmanned quadrotor Helicopters (UqHs) for forest fire surveillance is presented where UqHs are coordinated in a decentralized manner to patrol the perimeter of the fire.

Finally, another multi-robot testbed is presented in [100], where a scalable architecture for decentralized traffic management of multi robot systems has been proposed, implemented and evaluated experimentally. In the proposed scenario, a group of vehicles move autonomously in a shared environment and each vehicle is given a specific task to accomplish, on its own or in collaboration. A cooperative conflict avoidance policy is employed to prevent collisions and guarantee that each vehicle eventually accomplishes its individual task. Typical agents are inexpensive robots equipped with sensors, limited onboard processing and short-range ZigBee wireless communication. The overall firmware development is based on the Contiki [101] operating system, which runs on all the mote platforms used in the experiments. Contiki is a lightweight and flexible operating system for tiny networked sensor motes and it is built around a simple event driven kernel on top of which application programs are written with stack-less threads.

2.6 Event Localization and Odor Plume Source Localization in WSN

2.6.1 Event Localization

Localization of the event position is an essential capability of sensor networks in many practical applications. In many event monitoring applications sensor nodes are used only to collect data. That data is routed to a base station equipped with more computational resources where the event location is computed. The position estimation may be accomplished by a triangulation or a least-squares computation over a set of sensor measurements.

For example, consider the problem of localizing a stationary signal source using a set of sensor measurements. In the simplest setting, three or more amplitude measurements, say from microphones, may be used to determine the location of a signal source (see Figure 2.8). If the signal attenuation model is known (i.e such as the

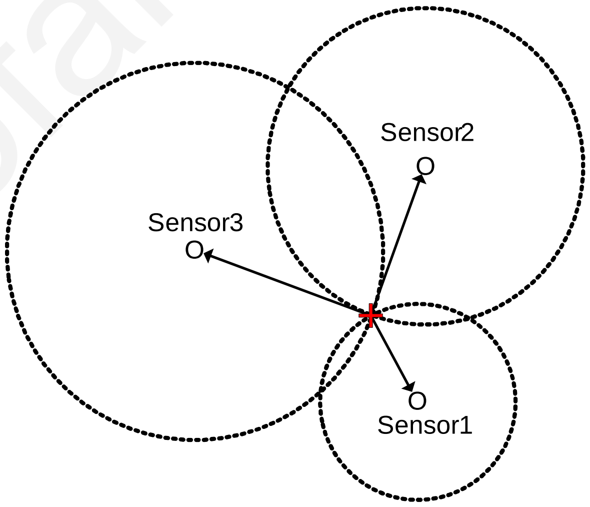


Figure 2.8: An example of a signal source localization using three sensor measurements.

acoustic model), one can recover one range constraint per amplitude measurement. To uniquely determine the location on a two-dimensional plane, one needs at least three independent distance measurements (the third is needed to resolve ambiguities).

Alternately, one may use the angle of Arrival (AoA) or the time difference of arrival (TDoA). In AoA techniques [102], source location can be estimated by exploiting the phase or angle difference measured at receiving sensor nodes using radio, sound or optical sensor arrays, which allow a listening node to determine the direction of a transmitting coherent, narrow band source. Therefore, by using several spatially separated sensors (sensor array) and analyzing the phase or angle difference between the signal"s arrival at different sensor array elements, it is possible to discover the angle of arrival of the signal. Unfortunately, AoA hardware tends to be bulkier and expensive since each node must have an array of sensors with specific spatial separation between sensor elements. Time of arrival (ToA) techniques can not be used since they require the exact time of signal emission by the unknown signal source as well as precise time measurement. Therefore, one may use time difference of arrival (TDoA) of signals at the sensors to estimate the range. TDoA method [103, 104] is suitable for a broadband source, however, this method requires accurate estimation of time delays between sensors (precise time synchronization among sensor nodes). TDoA methods could be also used in an alternative way (as used in the context of node position localization [105] or as one estimates the distance to a lightning cloud by measuring the TDoA between the light and the thunder) where the source emits two types of signals with different speed propagation times (e.g. acoustic and radio signals) and each node estimates distances by determining the TDoA of these signals using different type of sensors.

In contrast to the aforementioned methods, received signal strength (signal amplitude measurements given that signal attenuation model is known) is comparatively much easier and less costly to obtain from the time series recordings from each sensor and does not require any synchronization between the sensor nodes. Next, we focus on localization using signal amplitude measurements.

Assuming that the signal propagation model or the sensor measurement model is given by

$$z_i = \frac{V_i}{\left((x_e - x_i)^2 + (y_e - y_i)^2\right)^{\frac{\alpha}{2}}} + w_i \tag{2.14}$$

where $\mathbf{x}= [x_e, y_e]^T$ is the unknown position of the signal source, $\mathbf{s_i}= [x_i, y_i]^T$ is the position of sensor i, z_i is the amplitude measurement of sensor i, w_i is additive zero mean Gaussian noise with known covariance, V_i is a given variable representing the amplitude of the signal at the target and α is a known attenuation coefficient. Assuming $\alpha = 2$ which is equivalent to the inverse distance squared model for power attenuation and omitting the noise term, we can rewrite the signal model as $(x_e - x_i)^2+(y_e - y_i)^2 = \frac{V_i}{z_i}, \quad i = 1, 2, 3, \ldots$. For each sensor i, this equation gives a quadratic constraint on the unknown $\mathbf{x}$.

To solve this set of equations, it is more convenient to write it as a set of linear equations of $\mathbf{x}$. To do so, the quadratic terms ${x_e}^2$ and ${y_e}^2$ have to be removed. This can be achieved by subtracting the $i = 1$ equation from the rest $i \neq 1$ and obtain

$$2(x_i - x_1)x_e + 2(y_i - y_1)y_e = V_i(\frac{1}{z_1} - \frac{1}{z_i}) - (x_1^2 - x_i^2) - (y_1^2 - y_i^2) \tag{2.15}$$

Letting $c_i = 2\left[(x_i - x_1) \quad (y_i - y_1)\right]$ and $d_i = V_i\left(\frac{1}{z_1} - \frac{1}{z_i}\right) - (x_1^2 - x_i^2) - (y_1^2 - y_i^2)$ we can simplify the above as

$$c_i\mathbf{x} = d_i \tag{2.16}$$

Given k sensors, we can obtain $k-1$ linear equations, expressed in the matrix form $\mathbf{C}_{k-1}\mathbf{x} = \mathbf{D}_{k-1}$ and for $k = 3$, the above uniquely determines the location of the signal source $\mathbf{x}$. For $k > 3$, we can solve for it using the least-squares method [106]:

$$\mathbf{x} = \left[\left(\mathbf{C}_{k-1}^T\mathbf{C}_{k-1}\right)^{-1}\mathbf{C}_{k-1}^T\right]\mathbf{D}_{k-1} \tag{2.17}$$

The above localization method depends on the geometry of the sensor placement as well as the distance to the signal source which determine the significance of the contribution of each sensor.

2.6.2 Odor Plume Source Localization in WSN

Another application of mobile robots and mobile sensor networks is that of odor plume source localization. Robotic odor localization has become a prominent research area in recent years. It promises many valuable practical applications such as finding the source of dangerous substances like airborne toxic gases or hazardous chemicals in industrial plants, detecting fire in its initial stages, locating drugs or explosives, searching for survivors or human bodies in earthquake damaged buildings and landslides.

Odour plume propagation-dispersion includes two processes: advection and diffusion. In advection odor particles are purely shifting from one place to another by fluid currents. In diffusion odor particles are diffusing from locations with high concentrations to locations with low concentrations.

Dispersion is predicted by the Reynolds number of the flow. At low values, diffusion dominates producing smooth variations in chemical concentration. The peak occurs at the source, and decreases according to a Gaussian distribution. At medium to high Reynolds values, turbulence dominates and odor propagation mainly occurs through carriage of particles by the fluid currents (advection), causing an odor plume to form downflow of the source.

Time-varying flow currents and turbulence can cause the plume to meander, become patchy and to a far lesser extent spread out with diffusion. This creates on average a decreasing concentration gradient from the centerline of the plume (which may be meandering) out towards its edges across the flow. The patchy nature of the plume results in peak concentration values much higher than the average, and instantaneous concentrations fluctuate in magnitude and direction (see figure2.9). More details regarding the characteristics of odor propagation-dispersion can be found in environmental and fluid mechanics books [107, 108].

The odor plume localization using mobile robots requires three components; fast and accurate sensors, a suitable robotic platform and odor source localization algorithms

Figure 2.9: An example of a real smoke plume propagation-dispersion.

due to the complex propagation of odor molecules in the environment.

Recently many electrochemical sensors have been developed for detecting volatile chemicals such as alcohol, harmful gases, air contaminants and explosives. However such sensors, in order to be suitable for robotic plume tracking applications must offer high sensitivity, fast response and recovery, low power consumption, miniature size, low cost and long life. A number of different designs and technologies [109] exist including among others metal oxide semiconductors, conductive polymers, quartz crystal microbalances and optical sensors. Thick film metal oxide semiconductor gas sensors manufactured by Figaro Engineering Inc [110] have been used in the majority of robotic experiments [111, 112] with chemical plumes in the air. In aquatic environments optical or conductivity sensors are more appropriate. Optical backscatter sensors or nephelometric sensors are widely used for detecting hydrothermal plumes [113] or chemical substances released in water [114]. The particle concentration in the water can be estimated by measuring the amount of light scattered due to turbidity or cloudiness caused by diffusion.

However, beyond an appropriate sensor and a suitable robotic platform, the complex structures of plumes in the environment require the development of efficient odor source localization algorithms. Odor plume source localization algorithms can be classified into three broad categories: 1) bio-mimetic strategies that seek to emulate the remarkable feats of plume tracing in biological organisms, 2) Probabilistic approaches that focus on information theory and probabilistic techniques such as bayesian filtering for plume mapping 3) Multi-agent cooperative approaches. A comprehensive survey for robotic odor localization approaches that follows a different taxonomy is provided by Kowadlo and Russell [115]

The development of efficient odor source localization algorithms requires a realistic odor plume propagation model to test the plume-tracing algorithms through simulation before being realized in real robots. It's worth to say that realistic gas/odour plumes models are difficult to found in simulation environments which results in erroneous behavior of robots during real experiment for odor source localization. Two well known methods used for plume propagation modeling are the numerical solution to the advection-dispersion equation and stochastic simulation. Farrell et al. [116], use stochastic simulation to generate plume propagation model. This plume model was named the filament-based atmospheric dispersion model. It included a continuous wind field which covered the region of interests and the wind vectors varied with location and time in this region. The odor plume was represented by a sequence of puffs and each puff was composed of a number of filaments. The shape of the filament was predefined and the size and location of the filament were determined by the wind field. In such a plume model, the state of each cell of the plume map can be given by a Markov chain. At any time k, a cell i will either have a probability p_b originating a new plume release if it contains no plume at time $k-1$ or will have a probability p_c releasing the same amount of plume at time k if it contains a plume source at $k-1$ and with probability $p_d(j)$ will have a certain amount of plume at time k coming from the cell j at time $k-1$ depending on source intensity. This plume model is simplified, though computationally efficient but still be quite different from real plume propagation process.

Below we present a brief description of plume source localization algorithms, that have been classified into three broad categories:

Biomimetic Approaches

Numerous authors have attempted to implement on robots bio-mimetic approaches to the problem of odor plume localization. While many of these attempts proved successful to locate the odor source, the results have generally failed to match the performance of the biological entities. Two basic concepts generally used is chemotaxis and anemotaxis.

Chemotaxis is the most widely applied bio-mimetic odor plume localization approach. It follows a local gradient of the chemical concentration within a plume. In purely chemotactic search, a robot is equipped with a pair of gas sensors and turn toward the side with the higher concentration. However, the major problem is that there is no smooth concentration gradient in the patchy meandering plumes. A more sophisticated chemotactic approach is pursued by taking the nature of turbulent plumes into account [112]. Another approach adopted by Lytridis et al. [117] combine chemotaxis with a biased random walk, and discover that this hybrid outperforms its components. Chemotaxis is simple to understand and implement, but it may lead to locations in the plume that are far away from the true source.

Another popular biomimetic approach is anemotaxis. In anemotaxis agent observes the direction of fluid flow, and navigates upstream inside the plume. In anemotactic search, a robot is equipped with airflow sensors in addition to the gas sensors. However, the performances of the robots are severely limited by the capabilities of airflow sensors as their sensitivities are insufficient in general indoor environments. Although anemotaxis can be a very effective strategy for problems where the flow has no large-scale turbulence. In the case turbulent flows upstream movement causes anemotaxis to fail or can lead to a wind source that is not the chemical emitter.

Insects such as moths exhibit a plume-tracking behavior [118]; the flight of a male moth following a pheromone plume is known to consist of upwind surges when in contact with the plume and casting (cross-wind lateral excursions) when the contact is lost. The result is a zigzagging pattern that gradually progresses upwind toward the source. Basic features of the odor anemotactic flight of moths were implemented into mobile robots and reported in literature [119, 120]. Farrell et al. [121] report a successful plume tracing algorithm inspired by moths and implemented on a AUV. Their algorithm consisted of six behaviors switched by chemical detection events and timeouts. In [122] Hayes et al. developed an algorithm called the spiral surge algorithm where the robot moves along an outward spiral path until it perceives a certain concentration and then moves straight upwind. As soon as it looses the scent, it starts spiraling again. The size of the spiral can be decreased when moving closer to the source.

Probabilistic Approaches

Another category of algorithms used for the plume source localization problem is based on probabilistic approaches such as bayesian methods and information theory.

In [123] authors consider the chemical intermittency (i.e., the "puff" frequency), rather than chemical density gradients to find the source location base on the fact that the frequency of chemical puffs typically increases in the vicinity of the source and also demonstrate the advantages of chemical intermittency over other approaches.

Another interesting algorithm, entitled "infotaxis", proposed by Vergassola et al. [124]. Authors use information theory to decide the direction of the robot in order to maximize the expected rate of information gain. Their approach is useful in environments where areas of strong chemical signal are separated by regions of limited or no signal, so it is not possible to track the gradient to the source. The robot continuously samples the environment for odor packets and keeps a sequence of discrete detection events where the signal is above a predetermined threshold. The probability of a detection is higher nearer to the source, so the agent can use detections to build a probability distribution over the source location, $P(r_0)$. This distribution is built using a model of the source, which specifies the rate of emission of tracers, the lifetime of tracers, a diffusion coefficient and a mean current. The movement of the robot is determined by which action is likely to reduces the entropy (or uncertainty) of the source location by the most, where the entropy of a distribution $P(x)$ is given by $S = -\int P(x) \ln P(x) dx$. A lattice of potential new poses neighboring the current location r is found, and the entropy change on moving from r to each potential new pose r_j is evaluated by $\Delta S(r \to r_j) = P_t(r_j)[-S] + [1 - P_t(r_j)][\rho_0(r_j)\Delta S_0 + \rho_1(r_j)\Delta S_1 + ...]$. The first term of the above equation handles the case where the source is located at r_j , and the entropy drops to zero as the source has been found. The second terms

$\rho_k(r_j)$ denote the probability that k hits are detected at location r_j during a small time interval. This probability is found by assuming a Poisson hit distribution and ΔS_k terms represent the change in entropy after detecting k hits at r_j. The authors note that the first term is exploitative, encouraging the robot to go to the most likely source location, but the second term is exploration-biased, as it allows information gain even if the agent does not move. Therefore infotaxis provides a natural balance between exploration and exploitation in search problems. Simulation results show that, in the absence of hits, the algorithm produces interesting search patterns: without wind, the agent moves in increasing radius spirals, and with wind, it moves in a mixture of cross-wind zigzagging and up-wind casting very similar to the behavior of moths. This algorithm has been tested successfully in a number of numerical simulations, but not yet applied to even a simple "real world" plume-tracking situation.

Farrell et al. [114, 125], developed plume mapping and source localization approaches based on hidden Markov methods and bayesian inference. In [114] authors have used a hidden Markov model to model the plume and locate an odor source in simulation. The utilize the idea that moving in upwind direction, when global wind field is known, it is possible to locate the odor source. However, a robot cannot possess perfect sensors and the global wind field is not known. Therefore, a stochastic method (hidden Markov model) has been used. The robot uses the flow velocity and odor concentration measurements at a specific position to estimate the likely previous trajectory of the chemical plume. The predicted trajectories are accumulated over many detection events in order to construct an "Online Source Likelihood Map" (OSLIM). Specifically, they used a grid representation of the environment where the state of each cell represents the presence of detectable odor. A matrix A represents the transition probabilities, π represents the initial probabilities of each state, and b represents the detection probabilities, π is initially unknown and set to be equally distributed over the entire grid. b is known as it is the probability of detecting an odor if odor is present and is a constant over all states. A is known: it is calculated using the measured flow velocity. Using the standard hidden Markov methods, it is then possible to estimate a source probability vector and determine the path that is most likely to encounter odor.

In [125] the same authors present an improved method that comprises a computationally efficient algorithm that uses stochastic process theory and Bayesian inference methods to estimate the vector π. Dispersion of chemical filaments is modeled as a random walk superimposed on down flow advection (due to mean velocity). The random walk is modeled by a Gaussian random noise component. Both methods have been shown to be effective both simulation and practical implementation. The experiments show that bayesian inference method can be more effective to predict the likely location of an odor source over a large area, with increasing accuracy, as more data are gathered.

Multi-Agent Approaches

Several authors have developed multi-agent plume source localization algorithms. The basic idea is that search times can be reduced by sharing information across a distributed group of robots. In [126] each robot is executing its own plume search algorithm (spiral surge algorithm) and communication between robots is used to direct the swarm toward the robot having the greatest success. Alternate methods use robots as nodes in a distributed sensing network to estimate the parameters of a plume model including source location [127] or to instantaneously compute spatial gradients toward source location [128]. Zarzhitsky et al. [128] have approached the problem of tracking chemical plumes from an engineering viewpoint. They have devised an algorithm that uses flow properties to determine how a swarm of robots should move to locate and identify the source of a chemical plume. This algorithm, called "fluxotaxis", has the robot swarm calculating the local divergence of the mass flux. The swarm then moves up the gradient of this variable. This method has been implemented in a simulation environment and tested using a complex plume model where a group of robots has to maintain a geometrical pattern around a wide enough area to obtain meaningful data about the area they surround.

2.7 Summary

Obviously there has been much work in many areas related to mixed sensor networks and autonomous mobile multi-robot systems. In particular, the research work dealing with the aforemention formulated problems is vast, but most studies share common principles. In only a short period of time, the field of autonomous mobile multi-robots systems has developed considerably, from simple applications with single mobile robots, to complex applications with multi-robots systems that exploit collaboration. This chapter summarizes recent contributions to the problems of coverage in WSNs with static and mobile nodes, cooperative path planning and search for mobile robots and autonomous vehicles and finally the problem of event source localization using WSN or autonomous vehicles.

Chapter 3

Finding Paths between Two Arbitrary Points in Sensor Networks that Improve Coverage

3.1 Summary

This chapter investigates the path planning problem for improving the area coverage of a stationary sensor network using a single mobile sensor node. The mobile node is autonomously plan its path in order to navigate to an area of interest and at the same time, improve the area coverage by coordinating its movement to pass by areas that have not been adequately sampled by the static sensors. We present extensive simulation results of the algorithms investigated.

3.2 Introduction

This chapter considers a sensor network of randomly deployed stationary sensor nodes and the objective is to deploy a mobile sensor node which will navigate towards a predefined area of interest and during its motion it must pass through areas that are not adequately sampled by the stationary nodes and improve area coverage. For example, if there is a suspicion of an event source at a specific point, or if the user would like to send a mobile node on a mission to sample a target area, we would like to have the mobile sensor navigate autonomously to the area of interest and during its trip improving the area coverage by coordinating its movements to sample uncovered areas.

In this chapter, we develop and study the parameters of different path planning algorithms for a mobile node to improve the area coverage. The *area coverage* is defined as the fraction of the geographical area covered by at least one sensor during a time interval [0, t) and represents the quality of surveillance that the sensor network can provide. The area coverage of a stationary sensor network is determined by the initial network deployment and the area coverage of a mobile sensor network depends not only on the initial network deployment, but also on the mobility behavior of the sensors.

The main contribution of this chapter is the investigation of the parameters of various path planning algorithms for a mobile node using local information (information available at the current position by neighboring sensor nodes via communication). We investigate several path planning algorithms which are based on sensor node distances and some heuristics. The path planning algorithms in this chapter differ from robotic motion planning algorithms because in our case, static sensor nodes are simulated as "soft" obstacles and mobile nodes may run over them depending on optimization and time constrains. Moreover in our case coverage is improved by moving only a small number of mobile nodes and not assuming that all nodes have motion capabilities in order to reposition themselves in positions that enhance area coverage.

The chapter is organized as follows: Section 3.3 presents the model we have adopted and the underlying assumptions for the path planning algorithms we investigate. In Section 3.4 presents and explains the investigated path planning algorithms and Section 3.5, presents several simulation results using a number of sensor fields with randomly placed sensor nodes.

3.3 Simulation Model & Objectives

Our objective is to improve area coverage in a sensor network with stationary sensor nodes using a mobile sensor node and the motivation is to monitor a possibly huge area for events in the environment. For the definition of the problem we make the following modeling assumptions:

A1. A set of N stationary sensor nodes are randomly placed in a rectangular field $\mathcal{A}$ at positions $\mathbf{x}_i^s = (x_i^s, y_i^s)$, $i = 1, \cdots, N$.

A2. A mobile sensor node is placed in the rectangular field $\mathcal{A}$ at the position $\mathbf{x}(0) = (x(0), y(0))$. Also, $\mathbf{x}(k)$ denotes the position of the mobile node after the kth step, $k = 0, 1, 2, \cdots$.

A3. All sensors (stationary and mobile) know their locations through the use of a combination of GPS and localization algorithms.

A4. All sensors have a common sensing range r_s. Events within this sensing range are detected reliably and events outside this range are not detected at all (Boolean sensing model).

A5. All sensors have a common communication range $r_c > r_s$.

A6. An event occurs at a random point in region $\mathcal{R}$. Once the event occurs, it emits a constant signal/substance that propagates in a circular pattern.

A7. We assume that the target coordinates $\mathbf{x}_t = (x_t, y_t)$ of a suspected source are given either by the static nodes around the event or by a sensor network user who wants to sample this region.

Also, we define the *neighborhood* of a sensor node at position $\mathbf{x}$ as the set of all sensors that are located at a distance less than or equal to r_c. In other words,

$$\mathcal{N}(\mathbf{x}) = \{i \ : \ \|\mathbf{x} - \mathbf{x}_i^s\| \leq r_c, i = 1, \cdots, N\} \tag{3.1}$$

Given the above assumptions we develop a simulation environment (see Fig 3.1) and test the parameters of some path planning algorithms that can be used to improve the area coverage by coordinating the motion of mobile sensor node to cover areas that have not been adequately sampled (covered) by static sensors during its navigation from the initial position $\mathbf{x}(0)$ to the target position $\mathbf{x}_t$. The objective is to maximize the area coverage improvement over a time interval.

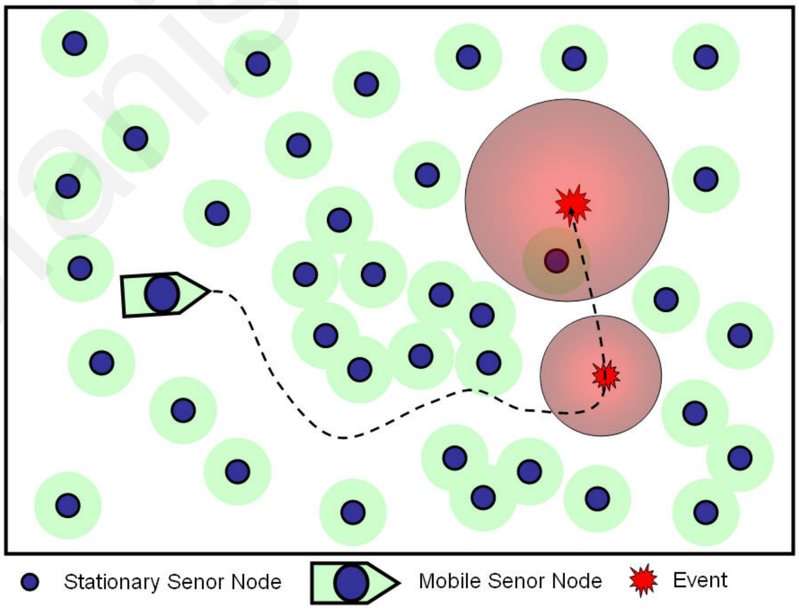


Figure 3.1: Simulation Environment.

3.4 Path Planning Algorithms

The investigated path planning algorithms will guide each mobile sensor node in the sensor field from its initial $\mathbf{x}(0)$ to the goal position $\mathbf{x}_t$ in order to improve the area coverage. Clearly, given more time, a mobile node can achieve better coverage, thus it is reasonable to apply a time constraint, such that the mobile will have to arrive at the target by the deadline. Such a constraint is used in the last simulation study of this chapter.

3.4.1 Algorithm Based on Artificial Potential Field

The first algorithm is based on artificial potential field method, an approach initially proposed for real-time collision avoidance in robotics form Khatib in 1986 [23]. In this algorithm the target (area of interest) generates an attracting potential force on the mobile node while the nearby stationary sensor nodes generate repulsive potential forces on mobile node. The mobile node can easily implement this algorithm. It is given the target position $\mathbf{x}_t$ (e.g., by a user) and as already mentioned we assume that the mobile node will know its current position either by GPS or a localization algorithm. Now at each step k the mobile will communicate with its neighboring nodes, i.e., nodes in $\mathcal{N}(\mathbf{x}(k))$ and will request their coordinates (or it might use RSSI (Received Signal Strength Indication) measurements to estimate the distance of the neighboring nodes though RSSI measurements can heavily oscillate [129]). The following algorithm will execute on the mobile node microcontroller in order to determine the next position of the mobile node. The force function equations are given below. The target attracting force is given by

$$\mathbf{F}_t(k) = -k_t \cdot (\mathbf{x}(k) - \mathbf{x}_t). \tag{3.2}$$

The repulsive force of each sensor is given by

$$\mathbf{F}_i^s(k) = \begin{cases} k_s \left(\frac{1}{r_i(k)} - \frac{1}{r_c}\right) \frac{1}{r_i^2(k)} \left(\mathbf{x}(k) - \mathbf{x}_i^s\right) & \text{if } r_i(k) \le r_c \\ 0 & \text{if } r_i(k) > r_c \end{cases} \tag{3.3}$$

where k_t, k_s, are the force coefficients and $r_i(k) = \sqrt{(x(k) - x_i^s)^2 + (y(k) - y_i^s)^2}$, is the distance from the stationary node i.

In this algorithm, the next position of mobile node is given by

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \sigma \frac{\mathbf{F}(k)}{\|\mathbf{F}(k)\|} \tag{3.4}$$

where

$$\mathbf{F}(k) = \mathbf{F}_t(k) + \sum_{i \in \mathcal{N}(\mathbf{x}(k))} \mathbf{F}_i^s(k)$$

is the resultant force and σ is the constant step size.

3.4.2 Algorithms Based on Receding-Horizon Approach

The second family of algorithms we investigate, is based on Receding-Horizon Approach. In this family of algorithms, the controller evaluates the one step cost and approximates the cost to go for several more steps. Then, it selects the position p^* that optimizes the overall cost, moves to p^* and repeats. Suppose, that during then kth step, the mobile node is at position $\mathbf{x}(k)$ and it is heading to a direction θ. The next possible points are the ν points $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ that are uniformly distributed on the arc that is ρ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$ as shown in Figure 3.2. Note that the parameters ρ and ϕ can be used to also model the maneuverability constraints of the mobile platform.

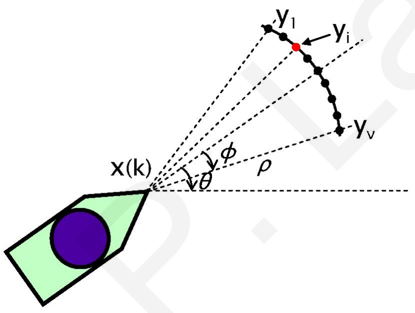


Figure 3.2: Selection of the next position in the mobile sensor node path.

Given the mobile node location/direction, the target point coordinates and neighboring static sensor positions the mobile node will use a cost function J to select the next step in order to improve (maximize) the area coverage during its motion to the target region. To achieve this we choose a cost function J that comprises of several sub-goals as shown in the equation below

$$J(k) = w_t \cdot J_t(k) + w_s \cdot J_s(k) + w_p \cdot J_p(k) + w_r \cdot J_r(k) \tag{3.5}$$

where $J_t(k)$ represents the cost associated with the movement towards the area of interest (target). $J_t(k) = f_1(\|\mathbf{x}(k) - \mathbf{x}_t)\|$ is a *decreasing* function of the distance between mobile node and target position. $J_s(k)$ represents the cost when the possible position $\mathbf{x}_i(k+1) = \mathbf{y}_i$, $i = 1, \cdots, \nu$ is near to stationary sensor nodes. $J_s(k) = f_2(\sum_{i \in \mathcal{N}(\mathbf{x}(k))} (1/r_i(k)))$ is a function of the sum of the inverse distances $1/r_i(k)$ between mobile node and neighboring stationary sensor positions and minimizes when the mobile node goes away from all stationary nodes in the neighborhood. $J_p(k)$ represents an extra cost (penalty) associated with the possible position $\mathbf{x}_i(k+1) = \mathbf{y}_i$, $i = 1, \cdots, \nu$ which takes a non-zero value when there is an overlap between the coverage area of the mobile node with the coverage area of the stationary sensor node. $J_r(k)$ represents an estimate of the cost to go several more steps in the same direction. This cost gives the algorithm a more global view of the problem but it also requires some global (not just local) information. $J_r(k) = f_4(n)$ is a function of the number

n of the sensors that are included in a triangular region associated with the heading direction of the mobile sensor when going from $\mathbf{x}(k)$ to $\mathbf{x}_i(k+1) = \mathbf{y}_i$, $i = 1, \cdots, \nu$ (see Fig 3.3). This triangle has two important parameters, the height μ and the angle δ. Note that as mentioned earlier, the evaluation of $J_r(k)$ requires some global information (i.e., information on the position of sensor nodes that may not be in the communication range of the mobile node). To overcome this problem, we assume that the mobile node, just before starting its trip, requests the positions of all life sensors and loads a map with all points that are covered by stationary nodes. We point out that during the trip this map may change because some nodes run out of energy and some other nodes may be isolated, thus the information contained in the map may not be very accurate, but for the purposes of this chapter, we assume that the loaded map is accurate. As the mobile node moves around, it updates the grid map at every step in order to include the regions that have been covered by the mobile node. As a result, the mobile node will avoid passing through regions that have been covered during previous steps. Finally, the cost functions $J_t(k)$, $J_s(k)$, $J_p(k)$, and $J_r(k)$ are normalized such that $0 \leq J_t(k), J_s(k), J_p(k), J_r(k) \leq 1$ and weights w_t, w_s, w_p, and w_r are selected for better performance.

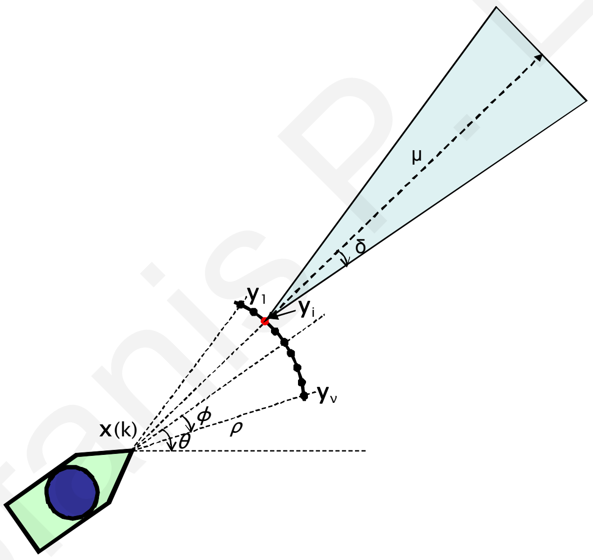


Figure 3.3: Triangular region used for the $J_r(k)$ cost function.

The algorithm will execute on mobile node microcontroller given as inputs the coordinates of the neighboring stationary nodes, its current location/direction and the coordinates of the target point. In case that we choose to use $J_r(k)$, as already mentioned, we have to load on the mobile node controller all stationary nodes positions. The output of the algorithm will be the next position of the mobile node where the cost function $J(k)$ gets the minimum value. In this way the mobile sensor will navigate autonomously to the area of interest and at the same time it will improve the area coverage because it will pass through regions that are not monitored by the stationary nodes.

According to the cost functions that we are going to use and their corresponding weights we will get different paths, computation delays, execution time and field coverage as shown in the next section.

3.5 Simulation Results

The first simulation study compares the coverage accomplished by the mobile node from five different paths (algorithms) (see Figures 3.4 and 3.5). The sensor field is a 200 by 200 area and we assume 250 randomly distributed stationary sensors to monitor the field. The coverage area of each sensor is a disc with radius $r_s = 5$ while the communication range of each sensor is also a circular disc but with radius $r_c = 5r_s$. Furthermore, for the receding horizon family of algorithms we assume that the arc radius is $\rho = 1$ and the angle $\phi = 45^\circ$.

Path 1: In this case the mobile node is following a straight line from its initial position to the target point. We show this case in order to get a clear view of the coverage improvements that we can accomplish using the other algorithms. Note that stationary sensor field coverage is 23.3% and after the mobile node coverage mission sensors field coverage is 26.3%.

Path 2: This path is obtained using the Receding-Horizon family of algorithms with the following weights $w_t = 0.01$, $w_s = 30$, $w_p = 0$, and $w_r = 0$. In other words, we only utilize the target cost function and the cost due to the neighboring static sensors function. This algorithm accomplishes 29.3% field coverage.

Path 3: This path is obtained again using the Receding-Horizon family but with the following weights $w_t = 0.01$, $w_s = 30$, $w_p = 0.3$, $w_r = 0$. In other words, we utilize the target cost function, the cost due to the neighboring static sensors and the penalty due to overlap between the covered area by the mobile sensor and that of stationary sensors in order to force the mobile node to pass as far away from static nodes as possible. This algorithm accomplishes 29.8% field coverage.

Path 4: This path is obtained again by using the Receding-Horizon family of algorithms but with the following weights $w_t = 0.01$, $w_s = 30$, $w_p = 0.3$, and $w_r = 0.5$ (the triangle has parameters $\delta = 20^\circ$, and $\mu = 15$). We utilize the target cost function, the neighboring static sensors cost function, the overlap penalty cost function and the triangular covered area cost. In this case the mobile node has a coverage map and global information and achieves the best sensor field coverage 30.9%. Although this algorithm has the best coverage performance it also has the largest computational cost.

Path 5: This path is obtained using the artificial potential field method with the following weights $k_t = 0.005$, and $k_s = 10$. This approach accomplishes 28% field coverage but it has the least computation cost. The problem is that some times the mobile node gets stuck in local minima and some heuristics in order to avoid this problem.

As is shown in Figures 3.4 and 3.5, Path 4 achieves the best area coverage. As

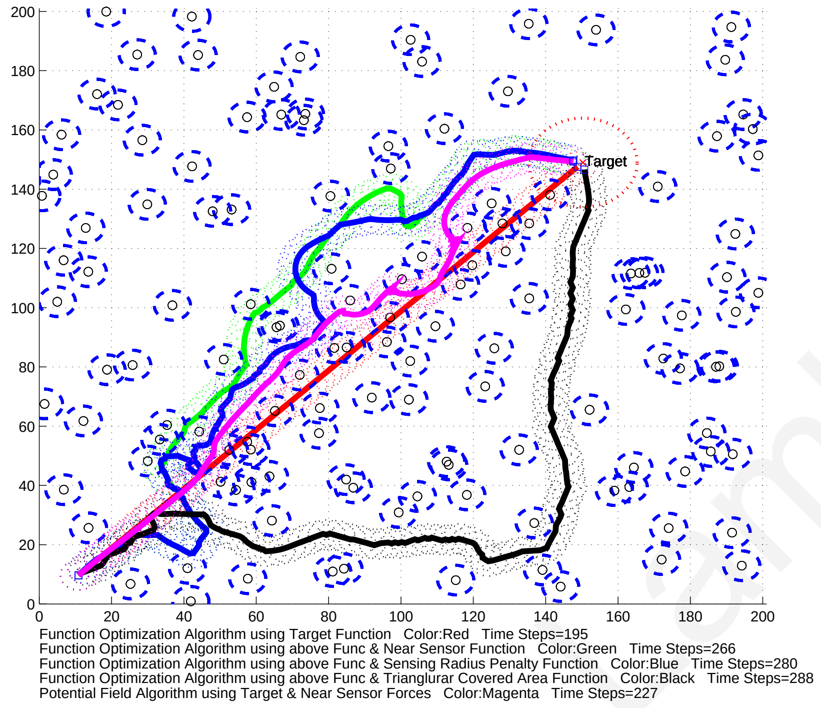


Figure 3.4: Paths followed by the mobile node when using different path planning algorithms.

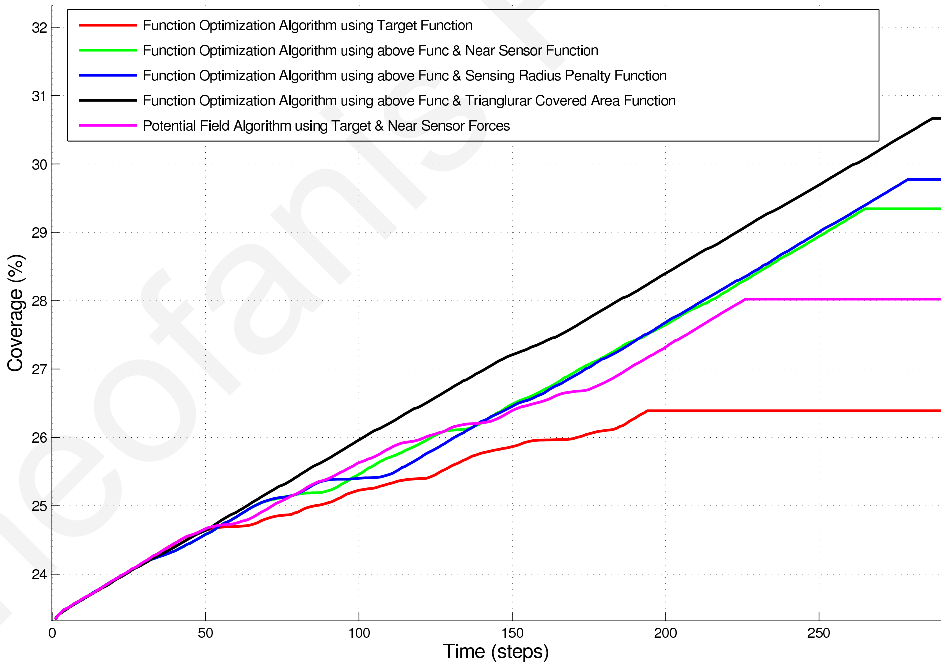


Figure 3.5: Coverage accomplished from 5 different path planning algorithms.

you can see from Fig 3.5 all others algorithms terminated (find target point) before algorithm 4, this is shown by a horizontal coverage line at the final time steps. An algorithm with better performance has a greater gradient over others because it achieves better coverage over time.

The second simulation study considers the coverage accomplished by mobile node using an algorithm from the Receding-Horizon family with the following weights $w_t = 0.008$, $w_s = 30$, $w_p = 0.5$, $w_r = 0.5$, and keeping all other values as in the first simulation experiment (i.e., this algorithm is similar to the general one that produced Path 4 earlier). During this experiment, we investigate the triangle parameters μ, and δ (see Fig 3.3) which maximize sensor field coverage. During this experiment, we measure the average coverage from 10 different randomly distributed sensor fields and vary the height of the triangle $0 < \mu < 60$ and $0 < \delta < 60°$ as shown in the Figures 3.6 and 3.7.

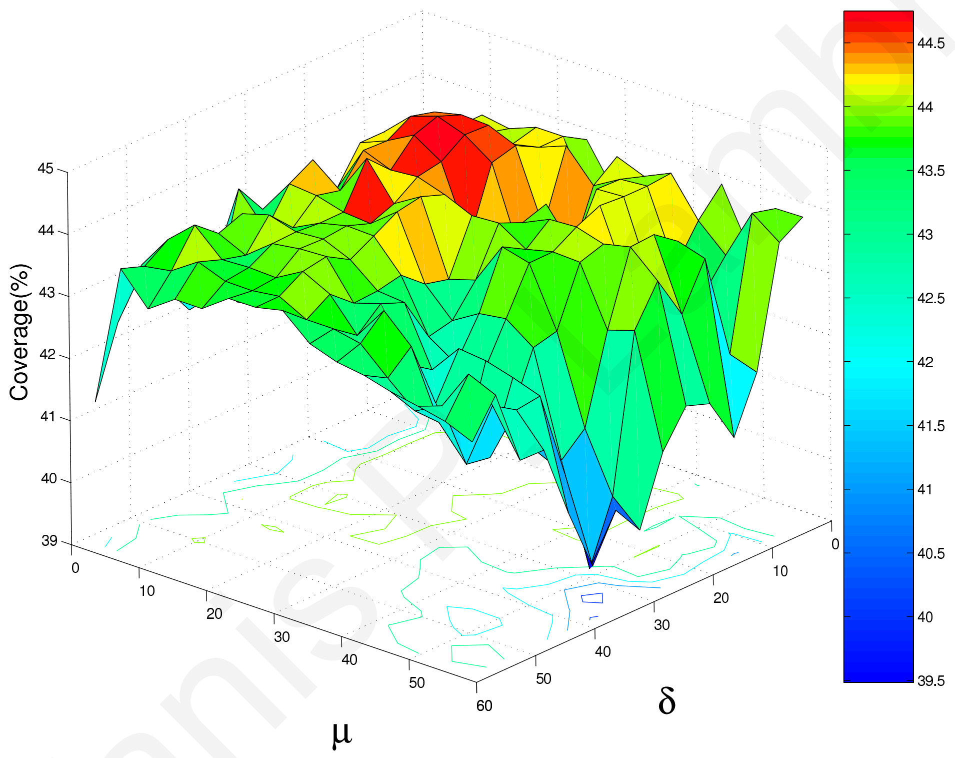


Figure 3.6: Average coverage with respect to μ and δ.

Simulation results show that for a 200 by 200 sensor field area, the maximum coverage is achieved at $14 < \mu^* < 22$ and $14° < \delta^* < 26°$. Therefore, best coverage is achieved if we have a fairly small triangle with height μ around 8-10% of the area length. Note that larger triangles, though more computationally demanding (they require more information) achieve less coverage. One might expect that more information would generate better results so this is a counter intuitive result. The explanation lies in the distribution of the stationary sensor nodes in the sensor field. Recall that sensors are placed *uniformly* in the field. As a result, the average number of sensors in a large enough triangle converges to the sensor density of the field. Therefore, all triangles used for all future positions ($\mathbf{y}_i$, $k = 1, \cdots, \nu$) will yield the same $J_r(\cdot)$ cost thus this term is simply a constant that does not play a role in the optimization.

In the third simulation experiment, we study how the weights in the Receding-Horizon family of algorithms influence the coverage accomplished by the mobile node. We keep the same values for all other parameters except w_q, where $q \in \{t, s, p, r\}$.

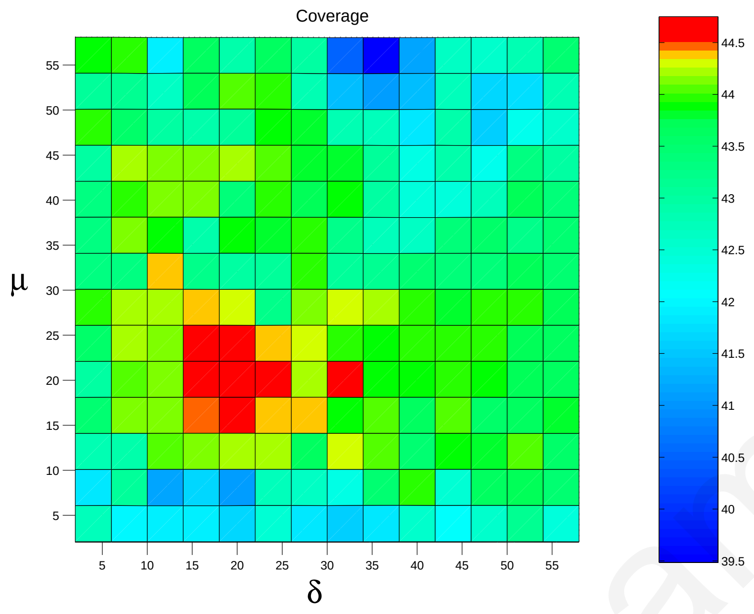
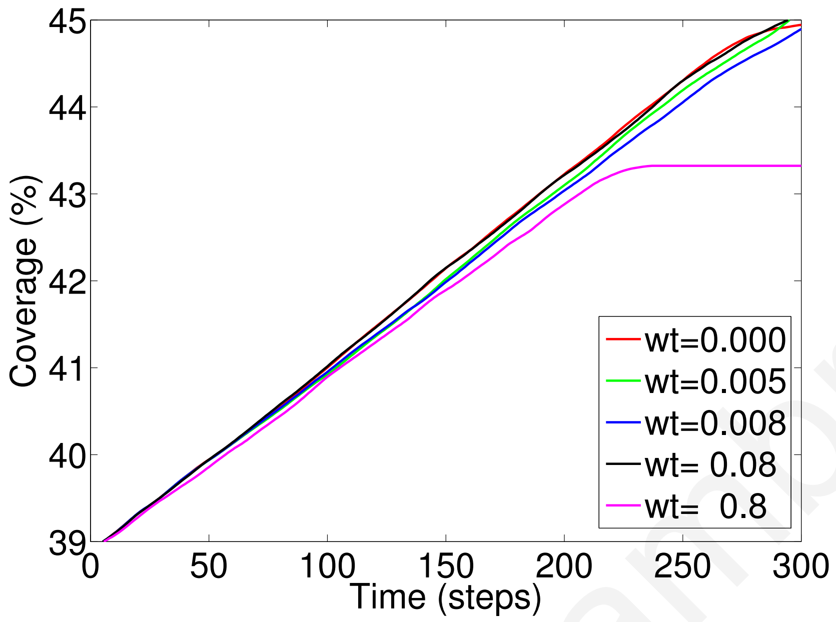


Figure 3.7: Average coverage with respect to μ and δ.

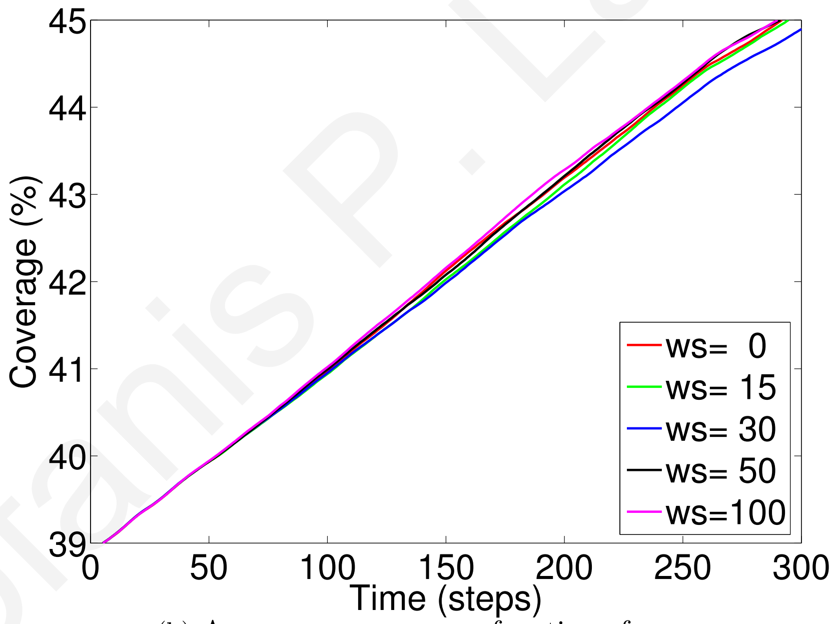
The results for these experiments are shown in Fig 3.8. The upper plot (Fig 3.8(a)) shows that big values for w_t move mobile node straight to the target and coverage is reduced. For values in the range $0.005 < w_t < 0.01$ we achieve good coverage and the mobile node finds the area for interest without significant delay. For small values i.e., $0 < w_t < 0.005$ the mobile node may achieve the best coverage but it will never go to the area of interest. In this case, we have to force the mobile node to go to the target point by increasing w_t after some time. The second plot (Fig 3.8(b)) shows that big values for w_s i.e $w_s > 50$ maximize coverage because mobile nodes will always go away from static sensors. But the mobile node will avoid the target area if some sensors are near the target area. The third plot (Fig 3.8(c)) shows that for $0.1 < w_p < 0.5$ the mobile node will avoid passing over the sensing radius of a static sensor. Large values for w_p i.e $w_p > 0.5$ might trap the mobile node in a static sensor region. The last plot (Fig 3.8(d)) shows that small values for w_r i.e., $0 < w_r < 0.05$ minimize coverage because the sensing triangle has diminished influence on mobile node planning. On the other hand, large values have better performance in coverage but coverage may decrease if w_r is made so large that mobile node decisions depends only on the J_r function.

The last simulation study considers the average coverage accomplished over 30 sensor fields by a mobile node from a set of 6 path planning algorithms with different sets of parameter values (see Fig 3.9). Each sensor field is again 200 by 200 area with 250 randomly distributed stationary sensors. For these simulations we are using the following values: $\rho = 1$, $\phi = 45°$, $r_s = 5$, and $r_c = 5 \cdot r_s$.

In this simulation, the mobile node has executed a coverage mission for 1000 time steps and then we forced the mobile node to find the target point by terminating its coverage mission and following a straight line to the target point. Results clearly show



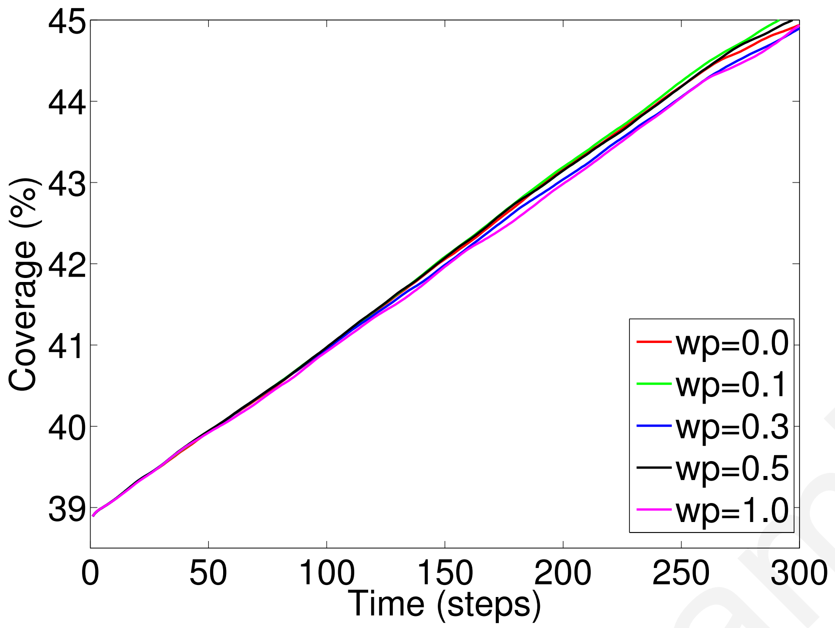
(a) Average coverage as a function of w_t



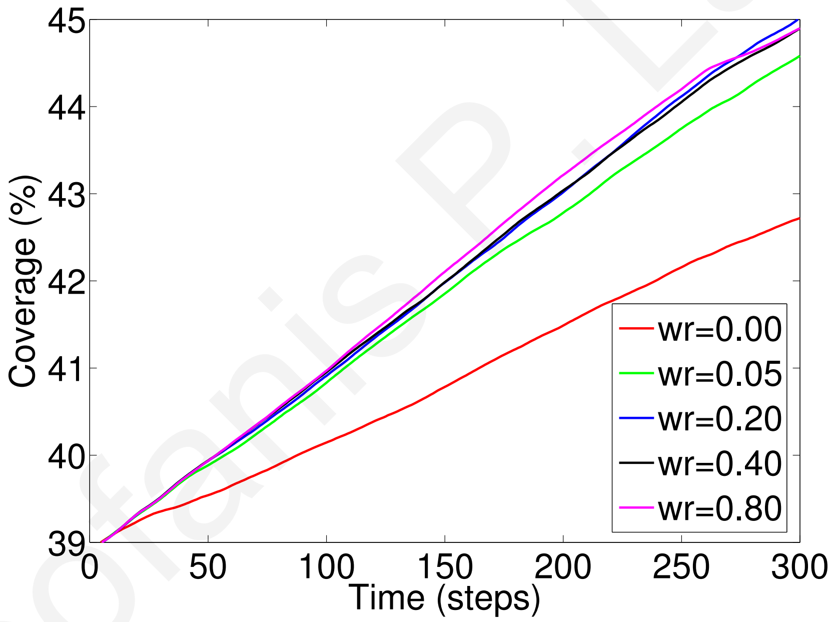
(b) Average coverage as a function of w_s

Figure 3.8: Average coverage as function of w_t, w_s, w_p, w_r

(see Fig 3.9) that Receding-Horizon algorithm 4 has the best average coverage over all other algorithms. A comparison between Receding-Horizon 4 and Receding-Horizon 5 algorithms shows that when we use a small triangular covered area, the average coverage is maximized as mentioned before. Receding-Horizon algorithms 2 and 3 achieve good coverage in short time because mobile node always moves towards the target point (terminates quickly) and mobile node avoids stationary sensors during its motion.



(c) Average coverage as a function of w_p



(d) Average coverage as a function of w_r

Figure 3.8: Average coverage as function of w_t, w_s, w_p, w_r (cont.)

3.6 Conclusion

In this chapter we develop and investigate the parameters of various path planning algorithms for a mobile node to improve the area coverage of a stationary sensor field. The path planning algorithms developed allow the mobile sensor node to navigate to a target point in the WSN while passing from areas that are not adequately sampled by the stationary sensor nodes. The simulation results show that a receding horizon family of algorithms achieves good area coverage improvement and is computationally tractable. Algorithms based on the potential field approach are also computationally

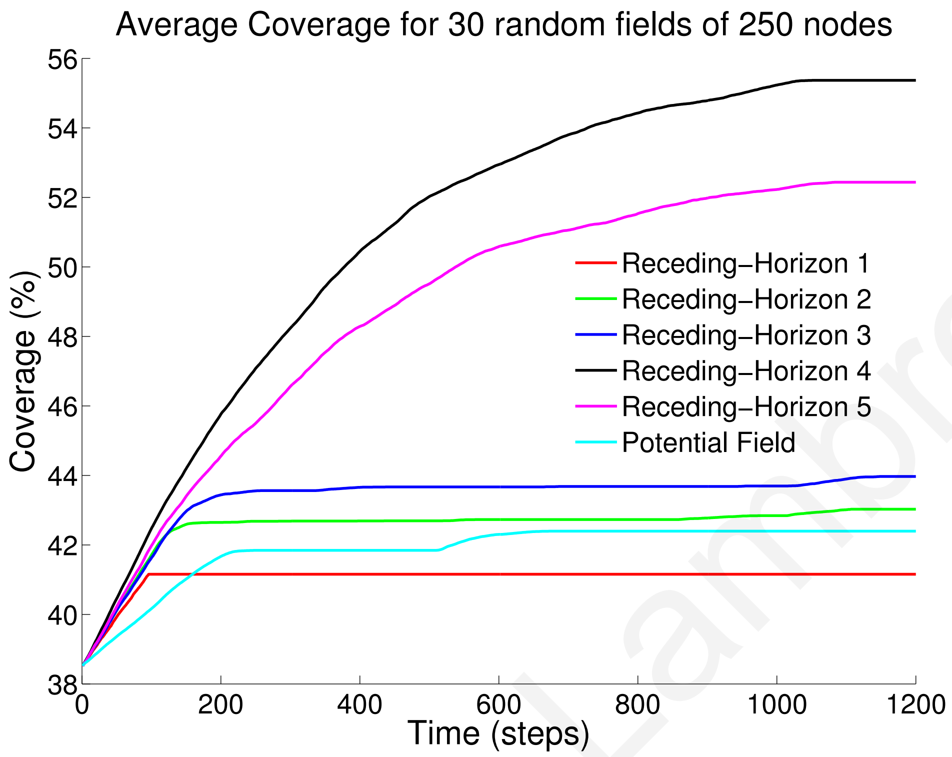


Figure 3.9: Average Coverage accomplished from 6 path planning algorithms over 30 sensor fields. The parameters used are, 1. Receding-Horizon parameter set: $w_t = 10$, $w_s = 0$, $w_p = 0$, $w_r = 0$. 2. Receding-Horizon with parameter set: $w_t = 10$, $w_s = 0$, 002, $w_p = 0$, and $w_r = 0$. 3. Receding-Horizon parameter set: $w_t = 10$, $w_s = 0.002$, $w_p = 0.3$, $w_r = 0$. 4. Receding-Horizon parameter set: $w_t = 10$, $w_s = 0,008$, $w_p = 0.4$, $w_r = 0.5$, $\delta = 20$, $\mu = 16$. 5. Receding-Horizon parameter set: $w_t = 10$, $w_s = 0.008$, $w_p = 0.4$, $w_r = 0.5$, $\delta = 40$, $\mu = 32$. 6. Potential field parameter set: $k_t = 0.005$, $k_s = 10$

efficient but may require some heuristics to allow them to escape from local minima. Finally, the receding horizon algorithm using an additional triangular region cost function achieves better area coverage if the triangular region is fairly small compared with the sensor field area. Larger triangular regions, though require more information, achieve less area coverage as the average number of sensors existing in a large enough triangle converges to the sensor density of the field and thus resulting the triangular region cost function to be a simply constant that does not play a role in the optimization.

Chapter 4

Coverage Hole Detection Algorithms

4.1 Summary

This chapter investigates the coverage hole detection problem for improving the area coverage of a stationary sensor network using mobile sensor nodes. We present the development of two coverage hole detection algorithms that are able to provide the coordinates of coverage holes in WSN which can be used as target points for the navigation of mobile sensors. We present theoretical analysis regarding the computation complexity of the algorithms developed and simulation results regarding the coverage hole detection efficiency.

4.2 Introduction

In the previous chapter we develop and analyze the performance of various types of path planning algorithms for navigating a mobile sensor node towards a predefined target point of interest. In this chapter we relax the assumption of the given target point (i.e. by WSN operator) by developing algorithms able to provide such a target point automatically given a sensor network deployment. Such algorithms entitle "coverage hole detection algorithms" are presented in this chapter. We present two coverage hole detection algorithms that have been developed and compare them with the Grid Scan Algorithm presented in [8]. A comparison between the three coverage hole detection algorithms regarding their computation complexity and hole detection efficiency is illustrated. A novel coverage hole detection algorithm (Zoom algorithm) capable of providing the locations of coverage holes with negligible computation needs has been developed.

Coverage hole detection algorithms require information about the covered areas of the sensor field, such information is represented by a grid map stored in the controller of the mobile nodes. In this chapter it is assumed that all mobiles are “sharing” the same map, however we point out that in forthcoming chapters we relax this assumption.

The main contribution of this chapter is the development of a novel coverage hole detection algorithm, entitled “Zoom algorithm”, capable of providing efficiently the locations of coverage holes in a sensor network with negligible computation needs. This algorithm can be run by an individual mobile sensor to determine a coverage hole in its region (used as a destination point) or by the sink to determine the destinations of all mobiles in the sensor network.

This chapter is organized as follows: Section 4.3 describes the model that has been adopted. Section 4.4 presents and analyzes the two algorithms for detecting coverage holes. and Section 3.5 presents several simulation results using various sensor fields with randomly deployed sensor nodes.

4.3 Model Description

We consider a set $\mathcal{S}$ with $N = |\mathcal{S}|$ stationary sensor nodes that are randomly placed in a rectangular field $\mathcal{R}_x \times \mathcal{R}_y$ at positions $\mathbf{x}_i = (x_i, y_i)$, $i = s_1, \cdots, s_N$. In addition, we assume that a set $\mathcal{M}$ of $M = |\mathcal{M}|$ mobile sensor nodes are available and their position after the kth time step is $\mathbf{x}_i(k) = (x_i(k), y_i(k))$, $i = m_1, \cdots, m_M$, $k = 0, 1, \cdots$. We assume that all sensors know their locations through a combination of GPS and localization algorithms. Each mobile node has a grid map of the field as shown in Fig. 4.1. The entire field area $\mathcal{A} = \mathcal{R}_x \times \mathcal{R}_y$ is divided by congruent rectangles to make up a grid. Each cell in the grid can be addressed by index (i, j) in two dimensions and each vertex has coordinates $(i \times dx, j \times dy)$ in 2D (in sensor field area) for some real numbers dx and dy representing the grid spacing or cell dimensions. For simplicity we set $dx = dy = d\ell$ which means that the cells are square. The dimensions of the grid are $X \times Y$ where $X = \lceil \mathcal{R}_x / d\ell \rceil$ and $Y = \lceil \mathcal{R}_y / d\ell \rceil$. Similarly, the detection range of each node $r = \lceil r_d / d\ell \rceil$. Moreover, we use $i = \lceil x_i / dl \rceil$ and $j = \lceil y_i / dl \rceil$ in order to transform sensor coordinates $\mathbf{x}_i$ into indexes of the Grid. $\lceil z \rceil$ indicates the smallest integer greater or equal to z. Also, for any cell (i, j) we define a *neighborhood* as the set of all cells that are at a distance r from cell (i, j), i.e., for all $1 \leq i \leq X,\ 1 \leq j \leq Y$

$$\mathcal{N}_r(i,j) = \{p, q \ : \ (p-i)^2 + (q-j)^2 \leq r^2 \ \} \tag{4.1}$$

where $1 \leq p \leq X,\ 1 \leq q \leq Y$. In the memory of the mobile node, the grid is represented by a matrix G where each entry $g(i,j)$ in G represents the probability of detecting the event if the event has occurred in the area that corresponds to the (i, j)-

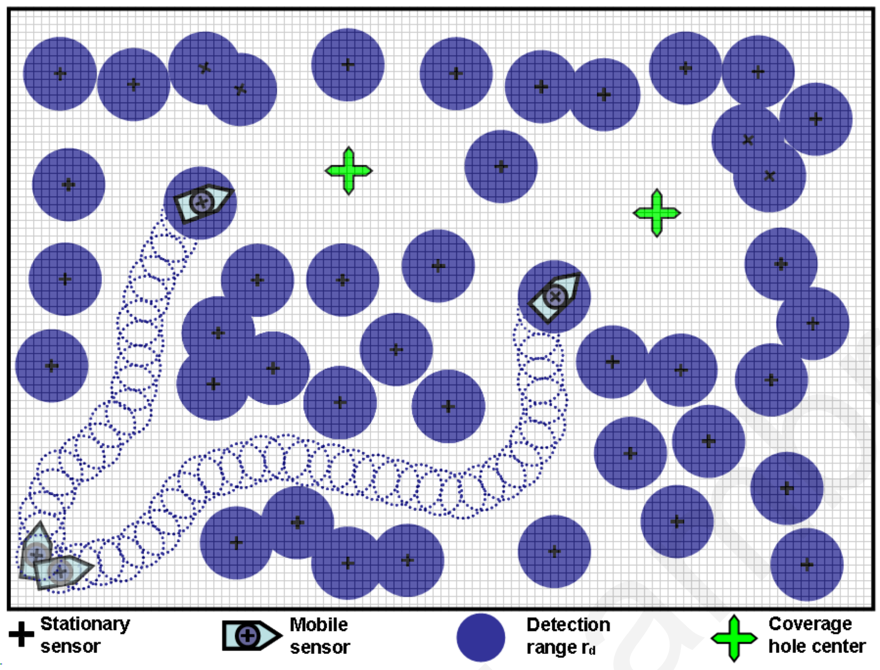


Figure 4.1: Field map for the mobile sensor nodes.

th square of the map. For simplicity, initially, every element of the matrix $g(i,j) = 1$ for all cells that correspond to areas in the detection range of the stationary sensors and $g(i,j) = 0$ otherwise.

Note that, it is unlikely that the mobile will have an accurate picture of the state of *all* stationary sensors in the field. The main idea is to update the map as the mobile node moves around in the field. Thus if the mobile encounters a node not on the map, or if it discovers that a node on the map is no longer functioning, it updates the corresponding entries in the matrix G appropriately. Furthermore, as it moves around, it also samples the environment and thus it increases the values of $g(i,j)$ that corresponds to sampled areas.

Note that in this chapter we assume that all mobiles are "sharing" the same map, however we point out that in forthcoming chapters we have investigated the scenario where each mobile has its own map and study algorithms for merging these maps together when the mobiles come in a communication range.

4.4 Coverage Hole Detection Algorithms

This section presents the two coverage hole detection algorithms developed and a comparison of their complexities with the Grid Scan Algorithm [8]. Using a coverage hole detection algorithm a centralized entity such as the sink can estimate the coordinates $\mathbf{x}_i^t = (x_i^t, y_i^t)$, $i \leq 1, \cdots, M$ of the M biggest coverage holes centers and assign them as the target coordinates of the M mobile nodes. As it is assumed that all mobile nodes share the same grid, thus it is easy to assign different coverage holes (targets) to different mobiles. Since this algorithm may run frequently (as new information regarding the state of the field becomes available) it is required that it is computationally efficient.

We define the *coverage* C as the probability of detecting an event which can occur uniformly in the sensor field. Using a grid similar to the grid map defined in section 4.3, C is given by

$$C = \frac{1}{X \times Y} \times \sum_{\substack{1 \le i \le X \\ 1 \le j \le Y}} g(i,j). \tag{4.2}$$

For the purposes of this chapter, we have already assumed that

$$g(i,j) = \begin{cases} 1 & \text{if } c(i,j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where $0 \le c(i,j) \le N$ is the number of sensors (stationary or mobile) that cover the area of cell (i,j). The aim of this section is to determine where the M mobiles should be placed in order to maximize coverage (i.e., maximize (4.2)). This algorithm can be run by an individual mobile sensor to determine its target destination or by the WSN sink to determine the destinations of all mobiles.

4.4.1 Grid Scan Algorithm

The Grid Scan algorithm [8] estimates the coverage holes as describe below: For each available mobile node Grid Scan scheme finds $c(i,j)$ for each cell (i,j). Then, for each cell (i,j), it finds the number of non covered cells (i.e., $c(p,q) = 0$), in the sensing range of radius r. In other words, it counts the cells with $c(p,q) = 0$ in $\mathcal{N}_r(i,j)$. We denote this by $h(i,j)$. The center of the hole is the center of the cell $Z = (i,j)$ which has the maximum number of neighboring non covered cells, i.e., Z is the cell (i,j) where

$$Z = (i^*, j^*) = \arg \max_{1 \le i \le X, 1 \le j \le Y} \{h(i,j)\}. \tag{4.4}$$

The details of the algorithm are listed in Fig. 4.2. Although the Grid Scan scheme estimates the coverage holes accurately, its computational complexity is fairly high for the context of WSNs. Another weakness of this algorithm appears in the case where more than one cells have the same max value. In that case the algorithm selects the first cell by default and as a result most estimated holes fall very near to each other. The running time of Grid Scan, if $X = Y = L$, is

$$\begin{aligned} T_{GS}(M, L, N, r) &= M \times \big(((N + (M-1)/2) \times L^2) \\ &\quad + (L^2 \times 2r \times 2r) + L^2\big). \end{aligned} \tag{4.5}$$

Given that $N \gg M$ the computation complexity is

$$\mathcal{O}(M \times L^2 \times (N + r^2)) \quad (4.6)$$

Grid Scan Algorithm

```
1:  for each mobile sensor $m \in \mathcal{M}$
2:      for each sensor $n \in \mathcal{S}$
3:          for each cell $(i,j) \in (X,Y)$
4:              if $(i,j) \in \mathcal{N}_r(x_n, y_n)$
5:                  $c(i,j) = c(i,j) + 1$
6:              end
7:          end
8:      end
9:      for each cell $(i,j) \in (X,Y)$
10:         for each $(p,q) \in \mathcal{N}_r(i,j)$
11:             if $c(p,q) == 0$
12:                 $h(i,j) = h(i,j) + 1$
13:             end
14:         end
15:     end
16: $(i^*, j^*) = \arg\max h(i,j)$
17: $x_m = i^*,\ y_m = j^*$
18: $\mathcal{S} = \mathcal{S} \cup \{m\}$
19: end
```

Figure 4.2: Pseudo code for Grid Scan Algorithm

4.4.2 One Scan Algorithm

The One Scan algorithm is a simple heuristic that improves the computational efficiency of the Grid Scan algorithm. One Scan scheme finds the $c(i,j)$ value for each cell in the Grid and for each cell (i,j) it finds the number of the neighboring non covered cells ($c(p,q) = 0$ in the sensing range of radius r, i.e $h(i,j)$ values). This computation take place only once. The new detected hole is the center of the cell $Z = (i,j)$ which has the maximum number of neighboring non covered cells, i.e Z is the cell (i,j) given by (4.4). Subsequently, we set $h(i,j) = 0$ for all cells that are in a neighborhood with radius $2r$ from the detected hole center. Finally, we continue finding the next Z using equation (4.4) until we determine all required coverage holes. The details of the algorithm are listed in Fig. 4.3. The running time of One Scan if $X = Y = L$ is

$$\begin{aligned} T_{OS}(M, L, N, r) &= (N \times L^2) + (L^2 \times (2r)^2) \\ &\quad + M \times ((4r)^2 + L^2) \end{aligned} \tag{4.7}$$

and since $N \gg M$, the computational complexity will be

$$\mathcal{O}(L^2 \times (N + M + r^2)) \approx \mathcal{O}(L^2 \times (N + r^2)). \tag{4.8}$$

One Scan Algorithm

```
1:  for each sensor $n \in \mathcal{S}$
2:      for each cell $(i, j) \in (X, Y)$
3:          if $(i, j) \in \mathcal{N}_r(x_n, y_n)$
4:              $c(i, j) = c(i, j) + 1$
5:          end
6:      end
7:  end
8:  for each cell $(i, j) \in (X, Y)$
9:      for each $(p, q) \in \mathcal{N}_r(i, j)$
10:         if $c(p, q) == 0$
11:             $h(i, j) = h(i, j) + 1$
12:         end
13:     end
14: end
15: for each mobile sensor $m \in \mathcal{M}$
16: $(i^*, j^*) = \arg\max h(i, j)$
17: $x_m = i^*,\ y_m = j^*$
      /* scans only cells included in 2r square region */
18:     for each cell $(p, q) \in \mathcal{N}_{2r}(x_m, y_m)$
19:         $h(p, q) = 0$
20:     end
21: end
```

Figure 4.3: Pseudo code for One Scan Algorithm

4.4.3 Zoom Algorithm

Using the principle of divide and conquer we propose the Zoom algorithm which is very efficient in computation complexity, time and memory. The idea is to divide the Grid in four equal segments, and choose the segment with the maximum number of empty cells i.e. the segment with the maximum number of cells with $c(i,j) = 0$[1]. Then, this procedure is repeated until either the segment size is equal to a single cell or until all segments have the same number of empty cells. In the first case the hole center position will be the center of the cell. In the second case, the hole center position will be the lower right corner of the upper left segment (the center of the segment during the previous iteration). Fig. 4.4 illustrates the idea of zooming for hole detection. The details of the algorithm are listed in Fig. 4.5. To evaluate the complexity of the Zoom

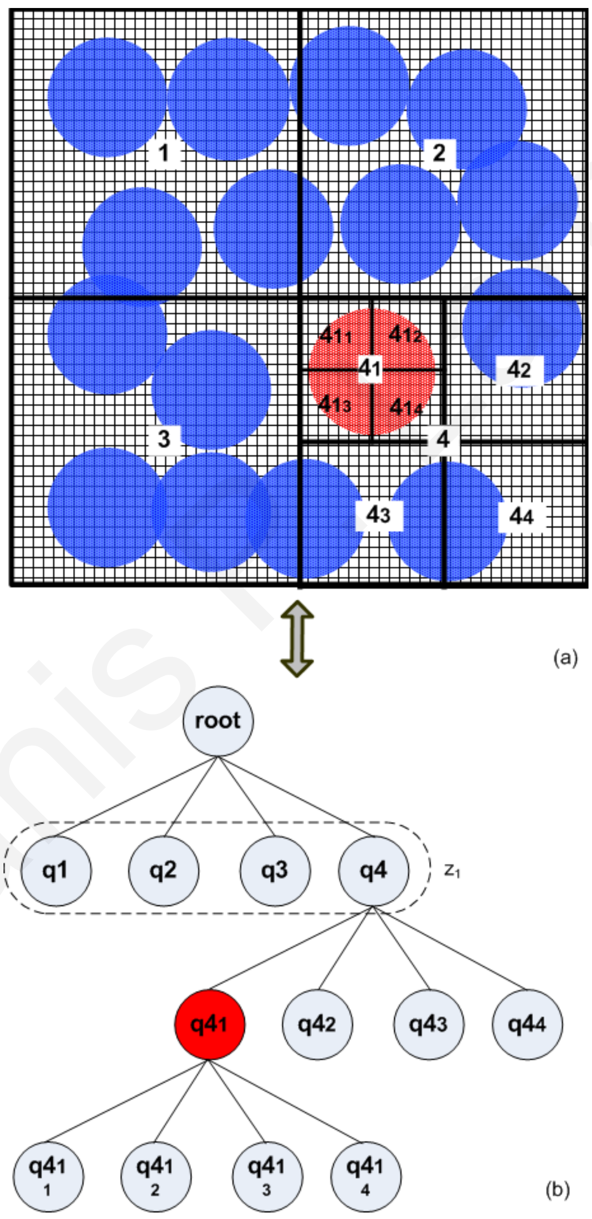


Figure 4.4: Illustration of the Zoom Algorithm (a) Grid Segmentation (b) Generated Tree

algorithm, we need to determine the number of times that the Grid will be divided. In the worst case, i.e., when the algorithm will stop with a single cell the Grid will be

[1]In more general one can choose the segment with the least coverage as defined by (4.2).

divided at most κ times, such that

$$\frac{L}{2^\kappa} \geq 1 \Rightarrow \kappa \leq \lg L \tag{4.9}$$

where κ is the number of iterations (height of the generated tree (see Fig. 4.4), with $\kappa = \lfloor \lg L \rfloor \in \mathcal{Z}^+$, and $\lg L$ is the binary logarithm ($\log_2 L$). Note that again we assumed that $L = X = Y$. As a result, the running time function of Zoom algorithm in the worst case will be

$$\begin{aligned} T_Z(M, L, N, r) &= (2r)^2 \times (M + N) \\ &+ 4M \times \lg L \times \left(1 + \sum_{i=1}^{\lfloor \lg L \rfloor} \left(\frac{L}{2^i}\right)^2\right) \end{aligned} \tag{4.10}$$

and since $N \gg M$ and $L \gg r$, its complexity is

$$\mathcal{O}(L^2 \times \lg L \times M + r^2 \times (N + M)) \approx \mathcal{O}(L^2 \times \lg L \times M). \tag{4.11}$$

4.5 Path Planning Algorithm

In this section we present a path planning algorithm utilized by each mobile sensor in order to navigate towards its target. The requirements for the navigation algorithm are shown below but first we note that in order to ease the notation, we dropped the index for each mobile, i.e., $\mathbf{x}(k)$ refers to the position of the i-th $i \in \mathcal{M}$ mobile sensor.

1. Guide each mobile sensor node in the sensor field from its initial position $\mathbf{x}(0)$ to its target position (e.g., the center of the coverage hole) $\mathbf{x}^t$.
2. Collaborate with stationary and other mobile nodes in order to improve the area coverage by sampling areas not covered by other sensors.

The algorithm presented in this section is motivated by [17] and is based on a receding-horizon approach (see chapter 3).

At each step k the mobile node evaluates a cost function $J_i(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k+1) = \mathbf{y}_{i^*}$ where i^* is the index that minimizes $J_i(\mathbf{y}_i)$,

$$J_{i^*}(y_{i^*}) = \min_{1 \leq i \leq \nu} \{J_i(\mathbf{y}_i)\}.$$

The cost function $J_i(\cdot)$ is in the form

$$J_i(\mathbf{y}_i) = \sum_{j \in \{t,c,m\}} w_j J_i^j(\mathbf{y}_i) \tag{4.12}$$

Zoom Algorithm

```
1:  for each sensor $n \in \mathcal{S}$
2:      for each cell $(i,j) \in \mathcal{N}_r(x_n, y_n)$
3:          $c(i,j) = c(i,j) + 1$
4:      end
5:  end
6:  for each mobile sensor $m \in \mathcal{M}$
7:      for each zooming step $z_x,\ x = 1, ..., \kappa$
8:          for each segment $q_s,\ s = 1, ..., 4 \in \mathcal{Z}_x$
            /* each segment has $L/2^{z_x} \times L/2^{z_x}$ cells */
9:              for each cell $(i,j) \in \mathcal{Q}_s$
10:                 if $c(i,j) == 0$
11:                     $a(q_s) = a(q_s) + 1$
12:                 end
13:             end
14:         end
15:         if $a(q_1) == a(q_2) == a(q_3) == a(q_4)$
16:             $x_m = \max\{i : (i,j) \in \mathcal{Q}_1)\}$
17:             $y_m = \min\{j : (i,j) \in \mathcal{Q}_1)\}$
18:             break
19:         end
20:         $(q_s^*) = \arg\max a(q_s)$
            /* select next region to segment */
21:         $x_m = \min\{i : (i,j) \in \mathcal{Q}_s^*\}$
22:         $y_m = \min\{j : (i,j) \in \mathcal{Q}_s^*\}$
23:     end
24:     place mobile sensor at $(x_m, y_m)$
25:     for each cell $(p,q) \in \mathcal{N}_r(x_m, y_m)$
26:         $c(p,q) = c(p,q) + 1$
27:     end
28: end
```

Figure 4.5: Pseudo code for the Zoom Algorithm

where the functions J_i^t, J_i^c, J_i^m, are defined to achieve certain objectives as defined next and w_t, w_c and w_m are positive weights such that $w_t + w_c + w_m = 1$ and are selected such that a desirable mobility performance is achieved (for example, if it is desired that a mobile quickly moves to its target destination, then w_t is made large).

The cost $J_i^t(\mathbf{y}_i)$ is a function that pulls the mobile towards the target location and is a function of the distance between mobile node and target position. This function should take a smaller value as the mobile moves towards the target destination thus for the purposes of this chapter it is given by

$$J_i^t(\mathbf{y}_i) = \frac{1}{\sqrt{A}} \|\mathbf{y}(i) - \mathbf{x}_t)\| \tag{4.13}$$

where A is the sensor field area.

The cost $J_i^c(\mathbf{y}_i)$ is a function that pushes the mobile away from covered areas (either by stationary or mobile sensors). This function should take a larger value if the candidate position is adequately covered by other sensors and a small value otherwise. Thus, for this chapter, the following cost is used

$$J_i^c(\mathbf{y}_i) = \frac{1}{\pi r_d^2} \sum_{\{i,j\} \in N_{r_d}(\mathbf{y}_i)} g(i,j) \tag{4.14}$$

where N_{r_d} is given by (4.1) and recall that r_d is the detection range of the sensor.

Finally, to facilitate the collaboration between mobiles, we use the cost function $J_i^m(\mathbf{y}_i)$ which penalizes each candidate position $\mathbf{y}_i$ that is close to other mobiles that are heading towards (or returning from) the same direction as the mobile tries to determine its next position. Specifically, when determining its next position, the mobile defines the set Λ that includes all other mobiles that are in its communication range and satisfy the following two conditions. 1) The mobiles that do not follow behind and 2) the mobiles that have a heading direction ξ such that $|\theta - \xi| \leq \varphi$ (the two mobiles are heading towards the same direction) or $|\theta - \xi| \geq 180^o - \varphi$ (the two mobiles are heading towards opposite directions), where φ is the maximum allowed difference in heading angle. For this chapter the collaboration function is given by

$$J_i^m(\mathbf{y}_i) = \sum_{\lambda \in \Lambda} \beta \exp^{-\frac{r_{i,\lambda}}{2}} \tag{4.15}$$

where β is a positive design constant and $r_{i,\lambda}$ is the distance between the candidate position $\mathbf{y}_i$ and the mobile λ.

4.6 Simulation Results

In this section we present some simulation results where we first compare the performance of the three hole detection algorithms presented earlier and also show a representative scenario with the movement of a set of two mobile nodes.

In the first scenario, there are 300 randomly deployed stationary sensors in an area of $500m \times 500m$. The coverage area of each sensor is a disc with radius $r_d = 20m$ and we set $d\ell = 1m$. For this scenario we investigate the required computation time and the coverage improvement of the three algorithms by assuming that each mobile node is simply placed in the center of the detected hole. As seen in Fig. 4.6 the Grid Scan algorithm can achieve slightly better results in terms of coverage than the One Scan and Zoom algorithms since it can more accurately detect the center of a hole (Grid Scan and One Scan achieve almost the same coverage). On the other hand, Fig. 4.7 indicates that the computational time requirements of the zoom algorithm are

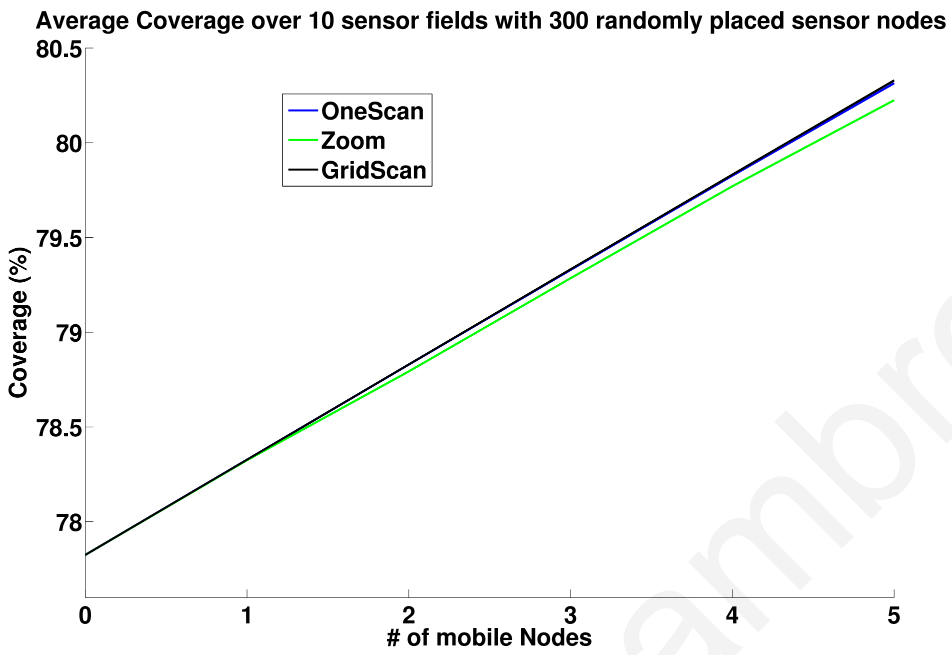


Figure 4.6: Average Coverage over 10 fields

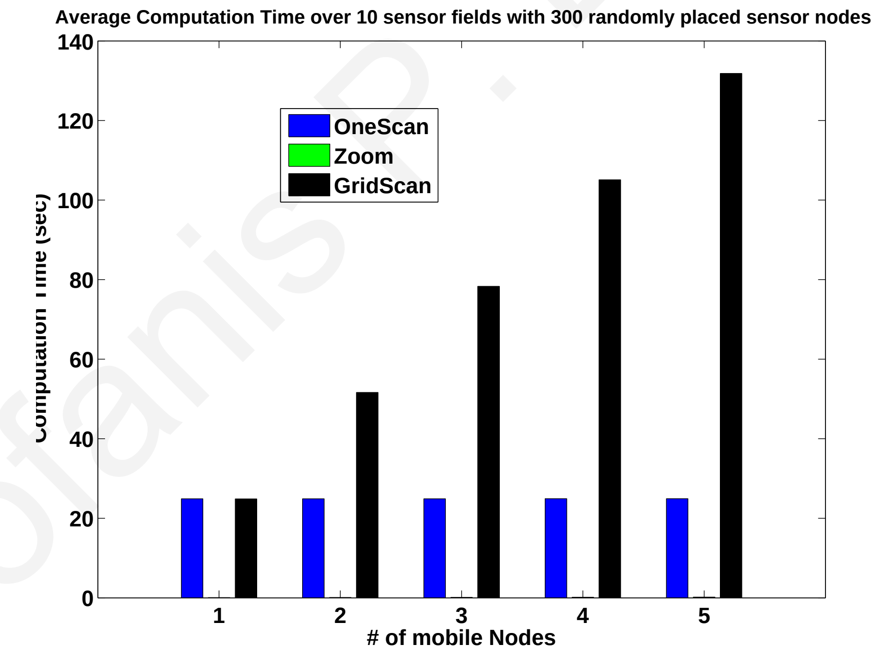


Figure 4.7: Average Computation Time over 10 fields

negligible (zoom bars are too short to be seen) compared to the requirements of the One Scan and Grid Scan algorithms. For all experiments we used MATLAB on an Intel Pentium 4, 3.6 GHz CPU machine. The actual time taken for each experiment is shown in Table 4.1. We emphasize that the efficiency of the Zoom algorithm allows it to easily run on a mobile node so that it can dynamically detect new holes (due to node failures that the mobile node discovers in its path).

In a second scenario we compare the computational requirements of the three algo-

Table 4.1: Average Computation Time over 10 fields

# Mobile	One Scan	Zoom	Grid Scan
1	24.8768 sec	0.0556 sec	24.8538 sec
2	24.8822 sec	0.0776 sec	51.6161 sec
3	24.8875 sec	0.1001 sec	78.3004 sec
4	24.8928 sec	0.1238 sec	105.0711 sec
5	24.8982 sec	0.1454 sec	131.8238 sec

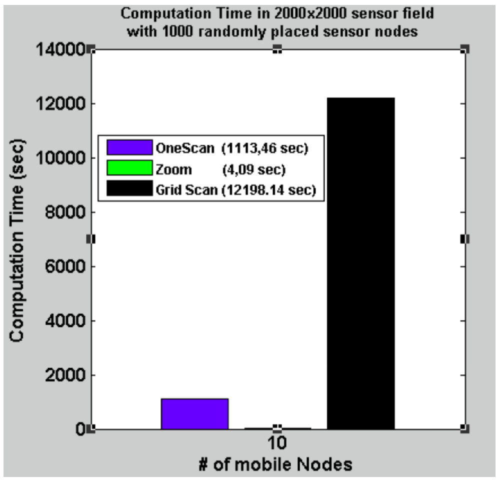
rithms for a larger field $2km \times 2km$ with 1000 stationary sensors. We set $d\ell = 1$ and the detection range for each sensor $r_d = 30m$. We assume that we try to determine the 10 biggest holes in the field. Table 4.2 shows the relative computational times, both from the simulation results as well as the derived running time functions (theoretical results column). From these results, we see that the zoom algorithm is 3 to 4 orders of magnitude faster than the other algorithms. Note that the discrepancy between the experimental and theoretical results of the zoom algorithm is because the theoretical results assume the worst case scenario. Finally, note that even though the Grid Scan achieves a slightly better coverage (see Fig. 4.6), for fields with low to moderate density, the detected holes fall very close to each other. On the other hand, the Zoom algorithm identifies holes that are more uniformly distributed in the field as shown in Fig. 4.8. In the last simulation experiment we use a team of two mobile nodes to

Table 4.2: Relative computational times

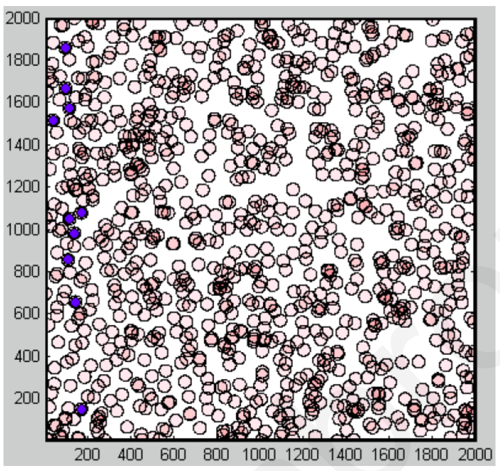
Algorithm	Experimental results	Theoretical results
GridScan T_{GS}/T_{GS}	1	1
OneScan T_{OS}/T_{GS}	0.09	0.08
Zoom T_Z/T_{GS}	0.3×10^{-3}	3.2×10^{-3}

illustrate the behavior of the proposed path planning algorithm. We assume a field with 100 randomly deployed stationary sensors in a $300m \times 300m$ area. The detection radius of all sensors is $r_d = 10m$ and it is indicated by the dotted circles. In this simulation scenario two mobile sensors navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors, moving towards their target locations, which are computed using the zoom algorithm. For this scenario the following parameters have been used: $\rho = 5m$, $\phi = 30°$, $\nu = 10$ and $r_c = 2.5 \cdot r_d$, where r_c is the communication range of the mobile nodes. Moreover we set $d\ell = 1$, $\varphi = \phi/2$, $\beta = 10$. In Fig. 4.9 we show the paths that the mobiles follow for two sets of weights w_t, w_c,and w_m. In the first set we have $w_t = 1$, $w_c = 0$,and $w_m = 0$, i.e., the objective is to send the mobiles to the targets as soon as possible. The path followed is show with red color in Fig 4.9 where one can see that the mobiles move in straight lines towards their targets and there is no collaboration between the sensors (both mobiles cover similar areas parts, which are also covered by the stationary sensors).

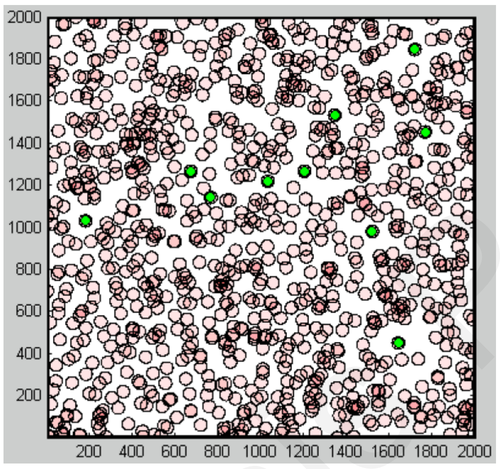
Fig. 4.9 also show the paths of the two mobiles when the weights are set to $w_t = 0.1$, $w_c = 0.4$, and $w_m = 0.5$ (black paths). As seen from the paths followed by the two mobile sensors, there is collaboration between mobile and stationary sensors in the sense that the mobiles have found two different paths that are least covered by the



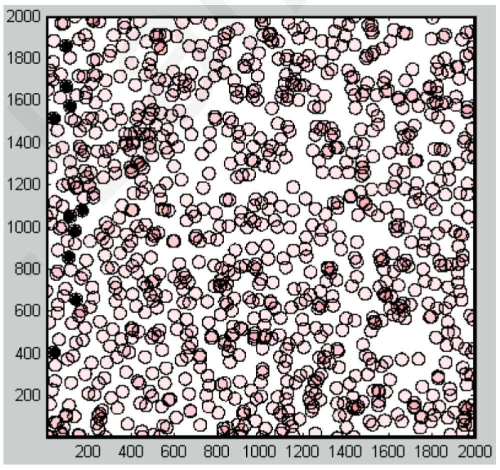
(a) Execution time



(b) Blue dots show the hole centers estimated by one scan



(c) Green dots show the hole centers estimated by zoom



(d) Black dots show the hole centers estimated by grid scan

Figure 4.8: Experimental execution time for 10 coverage holes detection and the corresponding target coordinates for mobile sensors

stationary sensors. Also notice how the two mobiles repelled each other due to J^m at the beginning of their motion.

4.7 Conclusion

In this chapter we propose two coverage hole detection algorithms able to provide the coordinates of coverage holes in a WSN which can be used as target points for the navigation of mobile sensors. We present theoretical analysis regarding the computation complexity of the algorithms developed and simulation results regarding the coverage hole detection efficiency. A novel algorithm entitled "Zoom algorithm" developed provides efficiently the locations of coverage holes in a sensor network with negligible computation needs. This algorithm can be run by an individual mobile

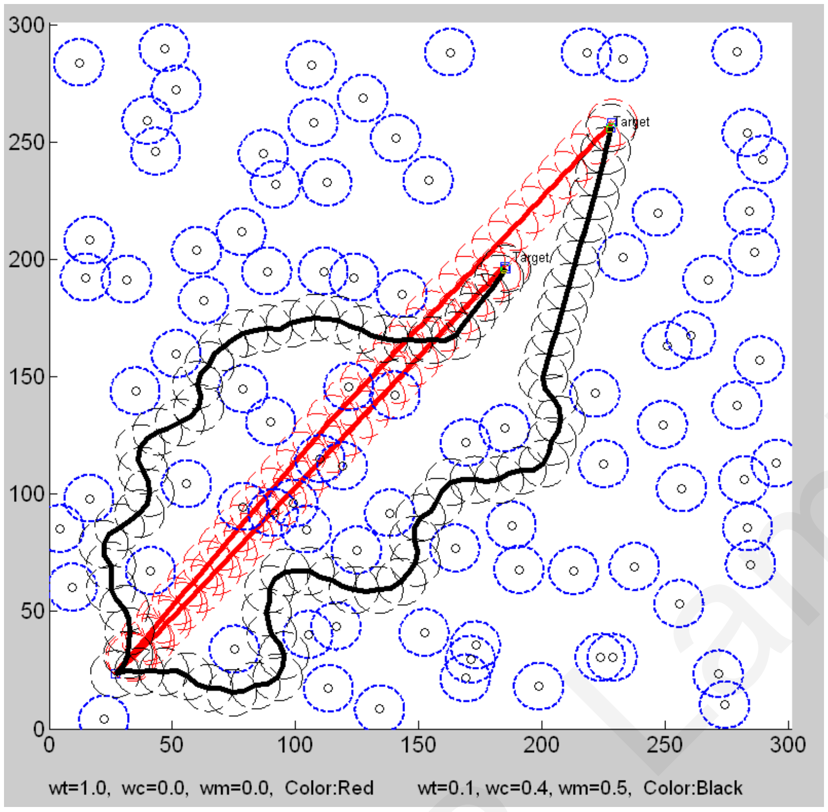


Figure 4.9: Dynamic path planning using a team of two mobile nodes.

sensor to determine a coverage hole in its region (used as a destination point) or by the sink to determine the destinations of all mobiles in the sensor network. Finally a path planning algorithm for navigating mobile nodes towards coverage holes is also illustrated.

Chapter 5

Complete Coverage in Sparse Sensor Networks using a single Mobile Node

5.1 Summary

This chapter investigates the complete coverage path planning problem for improving the area coverage of a sparse stationary sensor network using a mobile sensor node. We present a case study for complete coverage of the uncovered regions in a sparse sensor network. The algorithm consist of two major components, the global estimation of all the coverage holes in the sensor field and a path decision method for visiting all the coverage holes while avoiding passing over the region covered by stationary sensor nodes. We present simulation results based on a real sparse stationary WSN deployment.

5.2 Introduction

In the previous chapter we present the development of two coverage hole detection algorithms for finding the coordinates of a number of coverage holes in a sensor network deployment. In this chapter we modified the One scan Algorithm, proposed previously (see paragraph 4.4), in order to find the coordinates of all coverage holes in the sensor field. The estimated coverage holes positions can be used as way-points for complete area coverage by a single mobile node.

This chapter presents a case study for complete area coverage of a given sparse sensor network deployment shown in Fig. 5.1. This figure presents real WSN with 60 stationary sensor nodes monitoring the temperature in the ground floor of a building. Although it is possible to estimate the temperature at several points in the area, there are uncovered zones where we can not estimate the presence of some events with certain accuracy; for instance a fast increase of the temperature at a certain point due to a shortcut in the power supply. To overcome these problems, we could either increase the number of static nodes (i.e to get a dense WSN) or place sensors with more sensitivity. The former choice has the inconvenience of cost and excessive interference and the latter is not feasible with today's sensors and also depends on the propagation of the sampled signals in the environment. To circumvent these problems, in this chapter a mobile sensor node is used to allow complete area coverage of the uncover zones and also increase the probability of detecting any possible event.

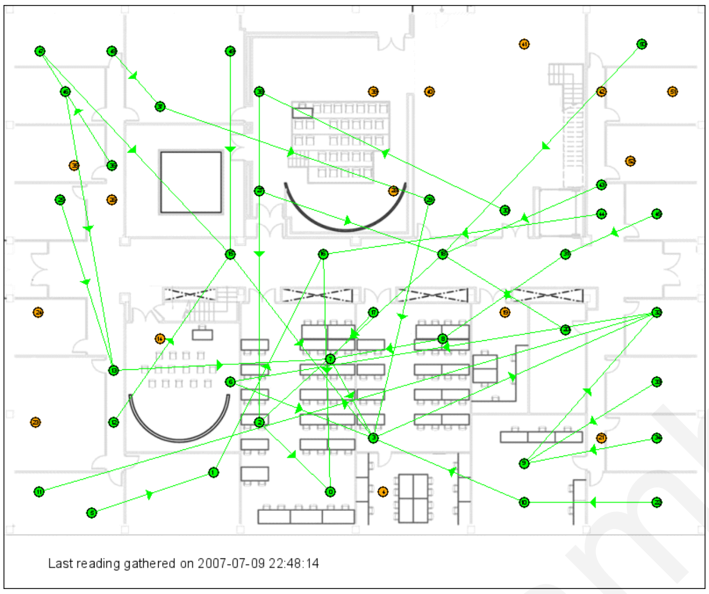
The main contribution of this chapter is the development of a complete coverage path planning algorithm for a mobile node which allows the mobile node to improve the area coverage and event detection in a real sparse stationary WSN. This algorithm is based on the estimation of the coverage holes coordinates in a sparse stationary WSN field. This estimation is done by the mobile node (or the sink) which has a map with the location of the already deployed static nodes and their sensing range. Once the coverage holes have been estimated, the mobile sensor visits all the coverage holes and at the same time avoids passing over covered regions using the path planning algorithm. Our aim is to improve the coverage of the network in a minimum amount of time.

The remaining of the chapter is organized as follows. Section 5.3 presents the model and the assumptions made. Section 5.4 describes the proposed algorithm and Section 5.5 presents the path followed by the mobile sensor node.

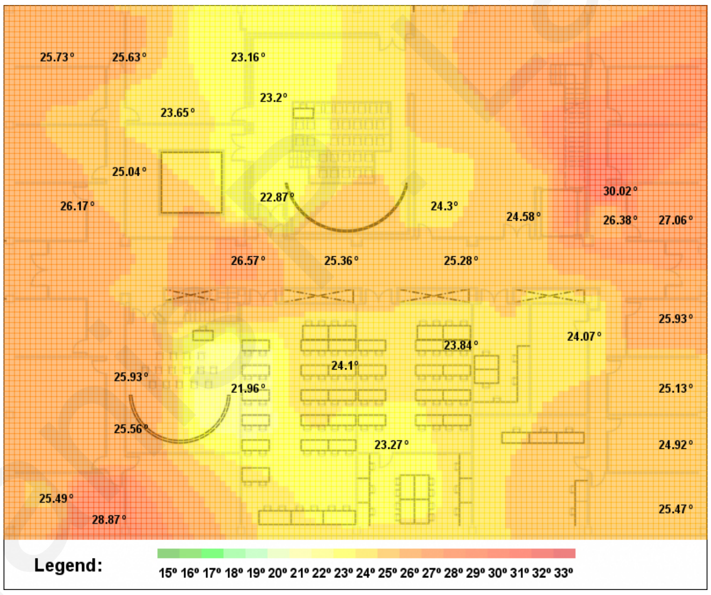
5.3 Assumptions and Definitions

For the definition of the problem we assume an area $\mathcal{A} = \mathcal{R}_x \times \mathcal{R}_y$ to be monitored by a stationary sensor field using a set $\mathcal{S}$ of $S = |\mathcal{S}|$ static sensor nodes are placed in $\mathcal{A}$ at positions $\mathbf{s}_i$, $i = 1, \cdots, S$. All static sensor and the mobile sensor have a common sensing range r_s and a communication range r_c. We assume $r_c > r_s$ and all nodes can reach the sink using multihop routing.

The sensor field area $\mathcal{A}$ is discretized into a into a $X \times Y$ grid $\mathcal{G}$. Each cell in the grid can be addressed by index (i, j) and each vertex has coordinates $(i \times dx, j \times dy)$ in 2D for some real numbers dx and dy representing the grid spacing. The sensing range of each node $r = \lceil r_s / \min\{dx, dy\} \rceil$. For simplicity in our case we set $dx = dy = 1$. Each cell of the grid $c(i, j)$ gets a value 0 or 1. $c(i, j) = 0$ means that the cell is not covered by any static sensor in the field. $c(i, j) = 1$ means that the cell is covered by at least one sensor. Fig 5.2 shows the grid of the sensor field of Fig 5.1.



(a) Connectivity Graph



(b) Temperature Histogram

Figure 5.1: Example of a real Wireless Sensor Network (WSN) testbed for temperature monitoring (100 × 80 square meters floor space). Connectivity Graph (above) and Temperature Histogram (below).

We define the *neighborhood region* in the grid $\mathcal{G}$, of a cell (i, j) for a distance r, as the set of all cells that their Euclidean distance is less than r,

$$\mathcal{N}_r(i,j) = \{p, q \ : \ (p-i)^2 + (q-j)^2 \leq r^2 \ \} \tag{5.1}$$

We define a set $\mathcal{H}$ of $H = |\mathcal{H}|$ *coverage holes*. The coordinates $\mathbf{h}_k$ of a *coverage hole* k are represented by the coordinates of the cell $c(i, j)$ which has the following properties:

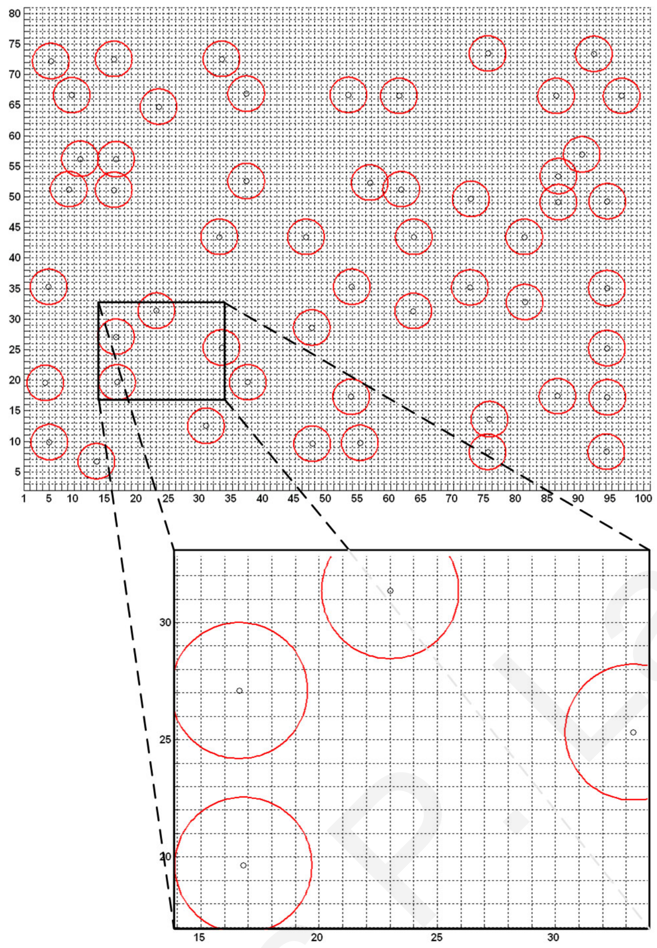


Figure 5.2: The Grid map of the static sensor network testbed

1. All the cells $c(p,q)$ in $\mathbf{h}_k$ *neighborhood region* have a zero value[1].

$$\sum_{\{p,q\}\in N_{r_s}(\mathbf{h}_k)} c(p,q) = 0$$

2. The minimum distance between any two *coverage holes* is $r_h \geq 2r_s$, which means that there is no overlapping between their *neighborhood regions*.

Fig 5.3 shows how to estimate the *coverage holes*.

[1]Alternatively, one may define a larger threshold, i.e., $\sum_{\{p,q\}\in N_{r_s}(\mathbf{h}_k)} c(p,q) \leq h$ to allow the algorithm to detect holes that are smaller than the sensing range of the sensors.

5.4 Complete Area Coverage Path Planning Algorithm

The path planning component of the algorithm requires the positions of *all* coverage holes in the sensor field. The coverage hole coordinates detection method is based on One Scan Algorithm developed in the previous chapter (see paragraph 4.4), where the Grid map $\mathcal{G}$ is scanned and updated in order to identify *all* the coverage holes coordinates $\mathbf{h}_i$, $i = 1, \cdots, H$ in the field. The details of the algorithm are listed in Fig. 5.3.

Coverage Hole Detection Algorithm

```
1:  for each sensor $s \in \mathcal{S}$
2:      if $(i, j) \in \mathcal{N}_r(x_s, y_s)$
3:          $c(i, j) = 1$
4:      end
5:  end
6:  for each cell $(i, j) \in (X, Y)$
7:      $Sum = 0$
8:      for each $(p, q) \in \mathcal{N}_r(i, j)$
9:          $Sum = Sum + c(p, q)$
10:     end
11:     if $Sum == 0$
12:         $k$++
13:         $\mathbf{h}_k = (i, j)$
14:         for each cell $(p, q) \in \mathcal{N}_r(\mathbf{h}_k)$
15:             $c(p, q) = 1$
16:         end
17:     end
18: end
```

Figure 5.3: Pseudo code for detecting the coverage holes coordinates

In this case study it is assumed that events will appear randomly in space and time and the minimum time t_e that the event is active, or the minimum time we need for the reaction when the event appears is large enough to allow the the mobile node to visit all the coverage holes in the sparse network and travel from the final position to the start position t_m. Nevertheless, this assumption sets the requirement for visiting all the coverage holes in the minimum amount of time.

Contrarily if $t_m > t_e$ then the sensor field has to be split into a number of working areas and each working area will be assigned to a mobile node in order to reliably detect events (because in each working area $t_m < t_e$). One simple heuristic way to split the sensor field into working areas is to divide the field into a number of rectangular segments with the same number of coverage holes, but this is out of the scope of this chapter.

Once the coverage holes coordinates are being estimated, the mobile node will execute the proposed *path-planning* algorithm, described in Fig 5.4, to improve the coverage and the event-detection. The mobile node will return to the sink when it is called by this entity or when it runs out of battery. In our scenario the mobile node has

Path-Planning Algorithm

```
1: void Exception() {
     /* when the sink requests to come back or it is run out of battery */
        Finish=TRUE;
   }
2: void Main() {
      if (NextTarget == NIL)
         /* when mobile has visited all holes in its working area*/
         List=InitialList;
         NextTarget=List(head);
      end
      while (NOT Finish)
         /* Path Planning Code*/
           for each CoverageHole h in List
                for each MovingStep k
                     Target = nearest unvisited CoverageHole in List;
                     for each CandidatePosition i
                        /* Solving Optimization Problem*/
                           $J_i = w_t.J_i^t + w_s.J_i^s$
                     end
                     Move to the CandidatePosition $i^*$ which has $min(J_i)$
                     if (MobilePosition ≈≈ Target)
                           Remove visited target from List
                           break;
                     end
                end
           end
      end
   } /*end-main*/
```

Figure 5.4: Pseudo code for the Path-Planning algorithm executed by mobile node in order to improve the coverage.

to visit all coverage holes in the sensor field. For each moving step the mobile node will calculate the nearest hole and move towards it and at the same time it will avoid passing over regions that are already covered by static sensors. Once the mobile node has visited a coverage hole it will remove this hole from its list and continue towards another nearby coverage hole. When it visits all coverage holes it will go to the first visited coverage hole and repeat. The next position of a mobile node is decided by solving an optimization problem. This optimization problem consist of two objectives. The first one it to select the next position such that the distance from the coverage hole is minimized and the second one is to select the next position that is away from the near stationary nodes (has a minimum overlap with static nodes).

The path planning decision method is based on multi-objective optimization (see chapter 3), where during the kth step, the mobile node is at position $\mathbf{x}(k)$ and it is heading to a direction θ. The next possible points to move are the ν points $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu))$ that are uniformly distributed on the arc that is ρ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$. At the kth position, the mobile node evaluates a cost function $J_i(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k+1) = \mathbf{y}_{i^*}$ where i^* is the index that minimizes $J_i(\mathbf{y}_i)$,

$$J_{i^*}(y_{i^*}) = \min_{1 \le i \le \nu} \{J_i(\mathbf{y}_i)\}.$$

The cost function $J_i(\cdot)$ is in the form

$$J_i(\mathbf{y}_i) = \sum_{j \in \{t,s\}} w_j J_i^j(\mathbf{y}_i) \tag{5.2}$$

where the functions J_i^t, J_i^s, are defined to achieve certain objectives as defined next and w_t and w_s are positive weights such that $w_t + w_s = 1$ and are selected such that a desirable mobility performance is achieved (for example, as described below, if it is desired that a mobile quickly moves to its target destination, then then w_t is made large).

The cost $J_i^t(\mathbf{y}_i)$ is a function that pulls the mobile towards the nearest coverage hole location and is a function of the distance between mobile node and target (nearest coverage hole) position. This function should take a smaller value as the mobile moves towards the target destination thus for the purposes of this chapter it is given by

$$J_i^t(\mathbf{y}_i) = \frac{1}{\sqrt{A}} \|\mathbf{y}(i) - \mathbf{h}_t)\| \tag{5.3}$$

where A is the sensor field area and it is used for normalization purposes.

The cost $J_i^s(\mathbf{y}_i)$ is a function that pushes the mobile away from covered areas. This function should take a larger value if the candidate position is near to stationary sensor nodes and small value otherwise. Thus, for this chapter, the following cost is used

$$J_i^s(\mathbf{y}_i) = \sum_{k \in N_{r_c}(\mathbf{y}_i)} \frac{1}{\|\mathbf{y}(i) - \mathbf{s}_k\|} \tag{5.4}$$

where N_{r_c} is given by (5.1) and recall that r_c is the communication range.

5.5 Simulation results

Using the deployed WSN in Fig 5.1, we create a simulation environment using a 100×80 grid where $X = 100$ and $Y = 80$ as shown in Fig 5.5. Once the grid map $\mathcal{G}$ is derived we can estimate the coverage holes using the algorithm of fig 5.3. The estimated coverage holes positions are indicated by green crosses in Fig 5.5. Finally,

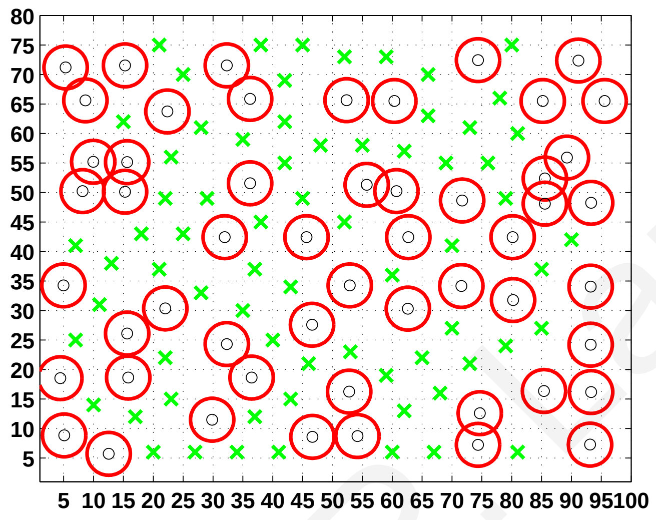


Figure 5.5: Uncovered zones in the sensing field, where $r_s = 3m$. Coverage holes coordinates are indicated by $\times$ marks.

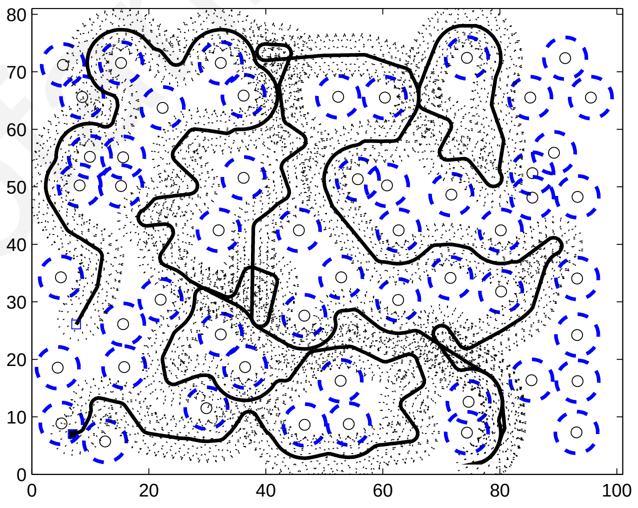


Figure 5.6: The path followed by the mobile node using the proposed path planning algorithm for complete coverage of the sensor field.

Fig 5.6 shows the path that the mobile sensor node follows for complete coverage of the sensor field. As shown in the figure 5.6 the mobile node visits all the coverage holes in the sensor field and at the same time it tries to avoid overlapping with static nodes by evaluating a cost function (solving an optimization problem).

5.6 Conclusions

In this chapter we present a case study for improving the area coverage in sparse sensor networks. The proposed method requires global information to solve the problem (positions of all static sensor nodes) and thus one can classify this solution as centralized. The path planning method developed succeeds complete area coverage of a sparse WSN with static nodes using a single mobile node. In the next chapter we proposed a decentralized-distributed complete coverage method that succeeds even better results regarding coverage time and complexity.

Chapter 6

Distributed Collaborative Path Planning for Complete Area Coverage in WSNs

6.1 Summary

This chapter investigates the collaborative complete coverage path planning problem in sparse stationary sensor networks using multiple mobile sensor nodes. We present the development of a distributed collaborative framework for complete area coverage where a small set of mobile nodes collaborate with the stationary sensor nodes and with each other in order to cover areas not covered (monitored) by stationary sensors. An important element of the proposed system is the ability of each mobile node to *autonomously* decide its path based on local information, which is essential in the context of large, distributed WSNs. The contribution of the chapter is the development of a distributed algorithm that allows mobile nodes to autonomously navigate through the field and improve the area coverage. We present simulation results for complete area coverage of sparse stationary WSN deployments and compare the algorithm with other well known approaches.

6.2 Introduction

In the previous chapter we present the development of centralized method for approximately complete area coverage of a stationary sensor network deployment using a single mobile sensor node. In this chapter we proposed a decentralized - distributed algorithm to solve this problem using a set of cooperative mobile sensor nodes.

Monitoring large areas with stationary sensor networks, typically requires a vast number of nodes to reliably cover the given region. Such an approach however, implies excessive (radio) interference and a prohibitive cost. Furthermore, in certain applications these nodes are manually deployed, which makes the deployment cost even higher. In such applications, involving a large area, coverage holes (areas not sufficiently monitored) are inevitable; either due to an effort to reduce the overall cost, or due to random failure of some nodes. To overcome the problem of coverage holes, we could either increase the number of static nodes (i.e. make a dense WSN) or increase the sensitivity of the sensors (increase the sensor sensing range). The former choice implies a higher cost and excessive radio interference while the latter may not be feasible either due to the signal propagation characteristics that an event emits in the environment or possible increase of the false alarm rates of the system. An alternative approach to address the problem is to employ mobile nodes, e.g. nodes mounted on robots. The mobile nodes can sample areas poorly covered by the stationary sensors. This approach that includes both static and mobile nodes is referred to as *mixed WSN*.

The main contribution of this chapter is the development of a distributed path planning algorithm, which allows each mobile node to autonomously navigate through the field and sample the areas least covered by the stationary sensor nodes, thus improving the area coverage. The algorithm is simple and can be implemented on any robot with very few resources. Furthermore, the mobile path is computed dynamically using only "local" information (i.e., information available from the stationary or mobile nodes within mobile's communication range). Such quality is particularly important in the context of large WSNs since it is generally not feasible to have an accurate view of the state of the field in order to precompute an optimal trajectory for each mobile node (some nodes may fail or be carried away). Finally, the proposed algorithm allows for collaboration between sensor nodes (stationary and mobile) in the sense that each mobile tries to avoid areas covered by other sensors.

The remaining of this chapter is organized as follows: Section 6.3 presents the model and the assumptions made. Section 6.4 describes the proposed algorithm and Section 6.5 presents simulation results for evaluating the algorithm in different sensor field deployments.

6.3 Assumptions and Definitions

In this section we present the modeling assumptions and define some concepts and objectives that will be used in the sequel. Furthermore, we present the information structure that is needed by the mobile nodes in order to run the path-planning algorithm. For the definition of the framework we make the following modeling assumptions:

A1. We assume the sensor field area is $\mathcal{A} = R_x \times R_y$.

A2. A set $\mathcal{S}$ of $S = |\mathcal{S}|$ static sensor nodes are placed in $\mathcal{A}$ at positions $\mathbf{x}_i = (x_i, y_i)$, $i = 1, \cdots, S$. It is assumed that all nodes know their coordinates.

A3. A set $\mathcal{M}$ of $M = |\mathcal{M}|$ mobile sensor nodes are available and their position after the k-th time step is $\mathbf{x}_i(k) = (x_i(k), y_i(k))$, $i = 1, \cdots, M$, $k = 0, 1, \cdots$.

A4. All static and mobile nodes have a common (known) sensing range r_s. This range can be computed from the event propagation model as well as the tolerated false alarm rate, e.g., see [130].

A5. All nodes have a common communication range $r_c > r_s$ and all nodes can reach the sink using multihop communication.

Next, for convenience, we define the set of all sensor nodes $\mathcal{N} = \mathcal{S} \cup \mathcal{M}$ and re-index all mobile nodes as $m = S+1, \cdots, S+M$. Furthermore we define the neighborhood of a sensor s as the set of all sensors that can be reached using single hop communication. In other words, the neighborhood of sensor $s \in \mathcal{N}$ is the set of all sensors that are in the disc centered at $\mathbf{x}_s$ with radius r_c.

$$\mathcal{H}_{r_c}(s) = \{j \ : \ \|\mathbf{x}_s - \mathbf{x}_j\| \leq r_c, \ j \in \mathcal{N}, j \neq s\} \tag{6.1}$$

for all $s = 1, \cdots, S+M$.

As already mentioned, the main objective of the chapter is to maximize the area coverage. To make the concept of coverage more concrete, we divide the field area in small squares with side da. In other words, we transform the sensor field area $\mathcal{A}$ into a grid $\mathcal{G}$ of size $X \times Y$, where $X = \lceil R_x/da \rceil$ and $Y = \lceil R_y/da \rceil$. Thus, we assume that any sensor $s \in \mathcal{N}$ is located in the cell $\mathbf{z}_s = (i, j)$, $i = \lceil x_s/da \rceil$ and $j = \lceil y_s/da \rceil$ (i.e., $\mathbf{z}_s$ is the discretized coordinate corresponding to $\mathbf{x}_s$). Furthermore, we assume that a sensor located in the cell $\mathbf{z}_s$, depending on the sensing range $\bar{r}_s = \lceil r_s/da \rceil$, covers a neighborhood of cells $\mathcal{D}_{\bar{r}_s}(\mathbf{z}_s)$,

$$\mathcal{D}_{\bar{r}_s}(\mathbf{z}_s) = \left\{(p, q) \ : \ (p-i)^2 + (q-j)^2 \leq l_s^2, \ \bar{\mathbf{x}}_s = (i, j), \ 1 \leq p \leq X, \ 1 \leq q \leq Y\right\} \tag{6.2}$$

where p and q are integers. We associate with the grid $\mathcal{G}$, an $X \times Y$ matrix G_k, $k = 0, 1, \cdots$, where each element of G_k captures our "confidence" that if an event occurs in the corresponding area of the field, it will be detected by the sensor network. If the (i, j)-th cell falls in the sensing range of a static sensor, then the corresponding

$G_k(i,j) = 1$, otherwise, $G_k(i,j) = 0$. As the mobile nodes move around, if they sample areas not covered by the static sensors, then our confidence increases and continues to increase as we take more samples. Furthermore, if a cell has not been sampled for some time, then it is possible that our confidence will be reduced. Thus at every step, we use the following updating rule for every element of matrix G_k.

$$G_{k+1}(i,j) = \begin{cases} 0.5 \cdot G_k(i,j) + 0.5, & \text{if } (i,j) \in D_{\bar{r}_s}(\bar{\mathbf{x}}_s), s \in \mathcal{N}. \\ f \cdot G_k(i,j), & \text{otherwise} \end{cases} \tag{6.3}$$

where $0 \le f \le 1$ is the "forgetting" factor. Thus, we define *coverage* as

$$C_k = \frac{1}{X \times Y} \times \sum_{\substack{1 \le i \le X \\ 1 \le j \le Y}} G_k(i,j). \tag{6.4}$$

Finally, to conclude this section, we describe the information required by each mobile in order to run the proposed path planning algorithm. Each mobile uses an $X \times Y$ matrix P_k^m, $m \in \mathcal{M}$ where it keeps the state of the field. Ideally P_k^m should remain $P_k^m = G_k$ at all times k, since the matrix G_k represents the accurate global state of the field which is used for the computation of the field coverage C_k. Clearly, in a dynamic environment where several sensors move, fail or more sensors are added, it is impossible to guarantee that $P_k^m = G_k$ at all times. However, we emphasize, that the proposed algorithm, that will run by a mobile located at some $\bar{\mathbf{x}}_m(k)$, computes its path based *only* on local information, i.e., information in the submatrix of P_k^m that corresponds to the cells $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$, where $\bar{r}_c = \lceil r_c/da \rceil$ and thus, it is sufficient to have accurate information only for the $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ submatrix. This is easily attainable since the required information can be obtained from the one-hop neighbors.

6.4 Dynamic Path Planning Algorithm

The path planning algorithm is motivated by the receding horizon approach presented in [17]. In this section in order to simplify the notation we have dropped the index of the mobile node, i.e. $\mathbf{x}(k)$ refers to the $\mathbf{x}_m(k)$, $m \in \mathcal{M}$. Mobile nodes make path planning decisions (execute the path planning algorithm) at discrete time intervals. At every step k, the mobile nodes decide their next position based on the solution of an optimization problem that captures the desired qualities of the path that is to be followed. The objective function is of the form

$$J(\mathbf{y}) = \sum_j w_j J_j(\mathbf{y}) \tag{6.5}$$

where $\mathbf{y}$ is any position in $\mathcal{A}$, $J_j(\cdot)$ is a specific objective and w_j's are non-negative constant weights such that $\sum_j w_j = 1$ and are selected so that a desirable mobility performance is achieved. For the purposes of this chapter, two specific functions will be used: $J_t(\cdot)$ which penalizes positions that are away from large coverage holes and $J_s(\cdot)$ which penalize positions that are close to static sensors. How $J_t(\cdot)$ and $J_s(\cdot)$ are computed will be presented in the sequel. It is worth pointing out that one can use several other functions (see for example chapter 3 where more functions are considered). Nevertheless, we found that these two are fairly simple and provide satisfactory results.

Suppose, that during the k-th step, the mobile node is at position $\mathbf{x}(k)$ and it is heading towards a direction θ. If the mobile moves at a constant speed, then at the next step, it will be located ρ meters away from $\mathbf{x}(k)$. Thus, the next possible positions are the ν points $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ that are uniformly distributed on the arc that is ρ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$ as shown in Fig. 6.1. Note that the parameters ρ and ϕ can be used to model the maneuverability constraints of the mobile platform. Also note that often it is desirable to have $\phi < \pi/2$ to avoid possible oscillations in the path of the mobile. At the k-th position, the mobile node evaluates the cost function $J(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k+1) = \mathbf{y}_{i^*}$ where i^* is the index that minimizes $J(\mathbf{y}_i)$,

$$J(y_{i^*}) = \min_{1 \le i \le \nu} \{J(\mathbf{y}_i)\}\,. \tag{6.6}$$

Next we present how the specific cost functions $J_t(\cdot)$ and $J_s(\cdot)$ used in (6.5) are computed.

6.4.1 Objective Functions

In order to improve the area coverage, the mobiles should move towards large uncovered regions and on their path, they should try (to the extend possible) to avoid areas that are covered by static sensors or have been covered by other mobile nodes. The two specific functions $J_t(\cdot)$ and $J_s(\cdot)$ are selected to achieve these two objectives.

Given the mobile's position $\mathbf{x}(k)$ and matrix P_k, then one can determine the point which is at the center of the biggest coverage hole in the area. An efficient algorithm for finding this point will be presented in the next section. We also point out, that one can find the biggest coverage hole of the entire area or of any sub-area (e.g., the neighborhood of cells around the current mobile position $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}(k))$). Let $\mathbf{x}_t$ denote the coordinates of the center of the obtained largest coverage hole, then this point becomes a candidate target for the mobile node. Thus, the specific cost $J_t(\mathbf{y})$ is a function that pulls the mobile node towards the target location and is a function of the distance between the candidate position $\mathbf{y}$ and target position $\mathbf{x}_t$. This function is given by

$$J_t(\mathbf{y}) = \frac{\|\mathbf{y} - \mathbf{x}_t\|}{r_c} \tag{6.7}$$

where r_c is the communication range and it is used for normalization purposes.

The cost $J_s(\mathbf{y})$ represents the cost when the candidate position $\mathbf{y}$ is very near to a sensor node located at $\mathbf{x}_j$, $j \in \mathcal{H}_{r_c}(m)$, where $\mathcal{H}_{r_c}(m)$ is the set of sensors in the neighborhood of mobile m, given by (6.1).

$$J_s(\mathbf{y}) = \max_{j \in \mathcal{H}_{rc}(m)} \left\{ \exp\left(- \frac{\|\mathbf{y} - \mathbf{x}_j\|^2}{r_s^2} \right) \right\} \tag{6.8}$$

This function serves as a local repulsion force. It repels the mobile node from its nearest neighboring sensor node. The sensing range r_s specifies the region size around the mobile node to be repelled by its neighbors. When $\|\mathbf{y} - \mathbf{x}_j\|$ is big relative to r_s, the whole term approaches zero.

At every step, the mobile node m computes (6.7) and (6.8) for all candidate points $\mathbf{y}_1, \cdots, \mathbf{y}_\nu$ and moves to the one that minimizes the overall function (6.6). In the next section, we present an efficient algorithm for estimating the center of the largest coverage hole in an area which will correspond to the point $\mathbf{x}_t$ in (6.7).

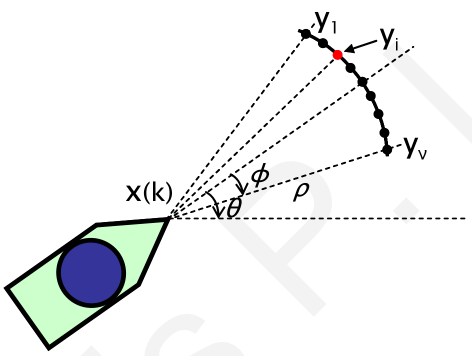


Figure 6.1: Evaluation of the mobile node's next step.

6.4.2 Coverage Hole Estimation Algorithm

Given the matrix P_k or any sub-matrix of P_k, a *coverage hole* is defined as the set of contiguous cells with $P_k(i,j) \leq p_{min}$, where p_{min} is some threshold and for the purposes of this chapter we use $p_{min} = 0$. Each mobile node calculates the coordinates of the center of the biggest coverage hole inside its communication range r_c at each moving step k. So, if the mobile is currently in the cell $\bar{\mathbf{x}}(k)$, it uses the "Zoom" algorithm, described next to determine the coordinates of the center of the largest coverage hole in $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}(k))$. The "Zoom" algorithm uses the principle of divide and conquer and is thus very efficient in computation complexity, time and memory. The algorithm starts by dividing the neighborhood $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}(k))$ in four equal segments, and chooses the segment with the maximum number of empty cells i.e the segment with the maximum number of cells with $P_k(i,j) = 0$. Then, this procedure is repeated on the selected region, until either the segment size is equal to a single cell or until all segments have the same number of empty cells. In the first case the hole center position will be the center of the cell. In the second case, the hole center position will be the lower right corner of the upper left segment (the center of the segment during the previous iteration). Fig.

6.2 illustrates the algorithm for hole detection. More details on the “Zoom” algorithm can be found in paragraph 4.4 of chapter 4.

The output of the “Zoom” algorithm is the point used as the target position $\mathbf{x}_t$ for the next iteration. We point out that at each step k, each mobile node estimates a new target in a receding horizon fashion. Using this target estimation, the mobile node always moves towards the biggest uncovered area in its neighborhood. Also this *dynamic* target behavior is useful for the collaboration between mobile nodes as will be described in Section 6.4.3.

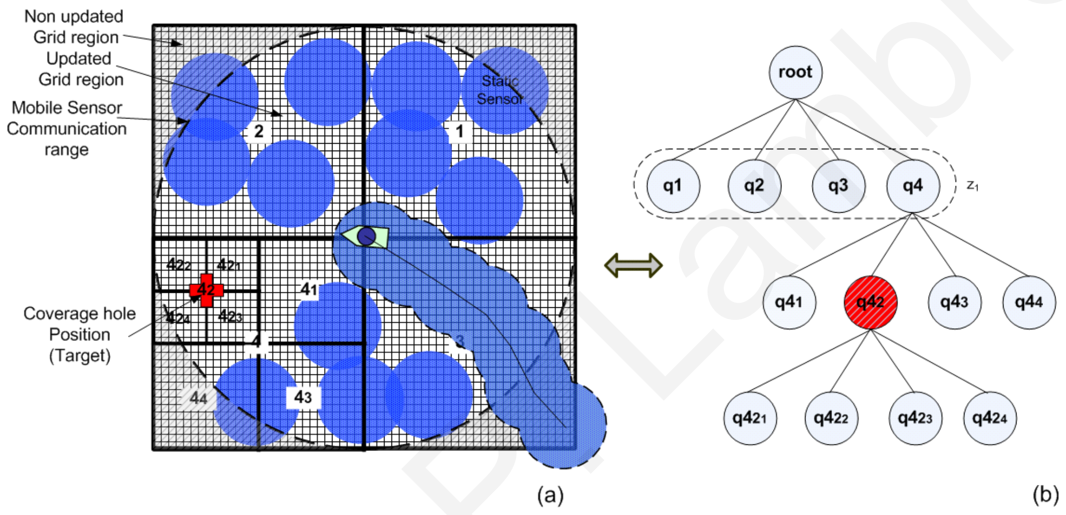


Figure 6.2: Illustration of the “Zoom” coverage hole estimation algorithm (a) Grid region segmentation (b) Generated tree.

In the next section we present a simple scheme that allows the mobiles to collaborate in the sense that they avoid each other and sample different regions.

6.4.3 Collaboration Between Mobile Nodes

In the proposed scheme, each mobile node autonomously estimates its target (using the “Zoom” algorithm described above) and when moving towards the target, it tries to avoid areas covered by neighboring stationary sensors. Therefore, if two mobiles are located in the same area, it is very likely, that both will estimate the same target and will move towards the same point. Thus, to avoid this behavior, it is necessary to execute a simple collaboration scheme that will enable them to coordinate their movements and sample different areas. The proposed collaboration scheme is described below:

1. When two or more mobile nodes come in communication range (for the first time) they share their P_k matrices; this sharing is useful in order to avoid regions that are already covered (explored) by the other mobiles.
2. At every step, before estimating the next target position (location of the coverage

hole center), a mobile queries all mobiles in its communication range for their target locations and their $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ cells. The target locations are treated as covered areas; like placing a static sensor in the target position. Subsequently, the "Zoom" algorithm is executed normally by the mobile node.

Note that since a mobile considers the targets of the other mobiles as covered areas, then it will never estimate the same target position, thus collaboration is easily achieved. A possible drawback of the algorithm is that the mobiles need to exchange their targets at every step, however, we point out that this requirement is only for mobiles that are in the communication range of each other. The pseudo code for the dynamic path-planning algorithm executed by each mobile node is presented in Fig. 6.3.

Dynamic Path Planning Algorithm

```
1: interrupt()
    /* when the sink requests to come back or it is run out of battery */
   {
       finish
   }
2: main()
   {
       Initialize matrix $P_k$
       while (! finish)
           for each time-step $k$
               Query neighboring nodes for their positions $\mathbf{x}_s$, $s \in \mathcal{H}_{r_c}(x(k))$
               if other mobiles $m$ exist in communication range $r_c$, $m \in \mathcal{H}_{r_c}(\mathbf{x}(k))$
                   if a new mobile node is discovered
                       Share $P_k$
                   end
                   Query neighboring mobile nodes for their $\mathbf{x}_t^m$ and $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$
               end
               Update $P_k$ given information send by neighboring nodes
               Estimate the coverage hole position $\mathbf{x}_t$
               /* Solving Optimization Problem*/
               for each candidate position $\mathbf{y}_i$, $1 \leq i \leq \nu$
                   $J_i(\mathbf{y}_i) = w_t.J_t(\mathbf{y}_i) + w_s.J_s(\mathbf{y}_i)$
               end
               $\mathbf{y}_{i^*} = \arg\min(J(\mathbf{y}_i))$, $1 \leq i \leq \nu$
               Move to the position $\mathbf{y}_{i^*}$
           end
       end
   }
```

Figure 6.3: Pseudo code for the dynamic path-planning algorithm executed by a mobile node in order to improve the coverage.

6.5 Simulation results

Using a sparse stationary sensor network deployment, we create a simulation environment using a 100×80 grid where $X = 100m$, $Y = 80m$ and $da = 1m$ as shown in Fig 6.4 (we use the same deployment as in chapter 5). The sensing range of each sensor is set to $r_s = 2.5m$ and the communication range $r_c = 4 \times r_s$. For the purposes of the results presented in this section, the following parameters were also used $\rho = 1m$, $\phi = 25°$, $\nu = 10$, and the weights are set to $w_t = w_s = 0.5$. Furthermore, it is assumed that the forgetting factor in (6.3) is set to $f = 1$.

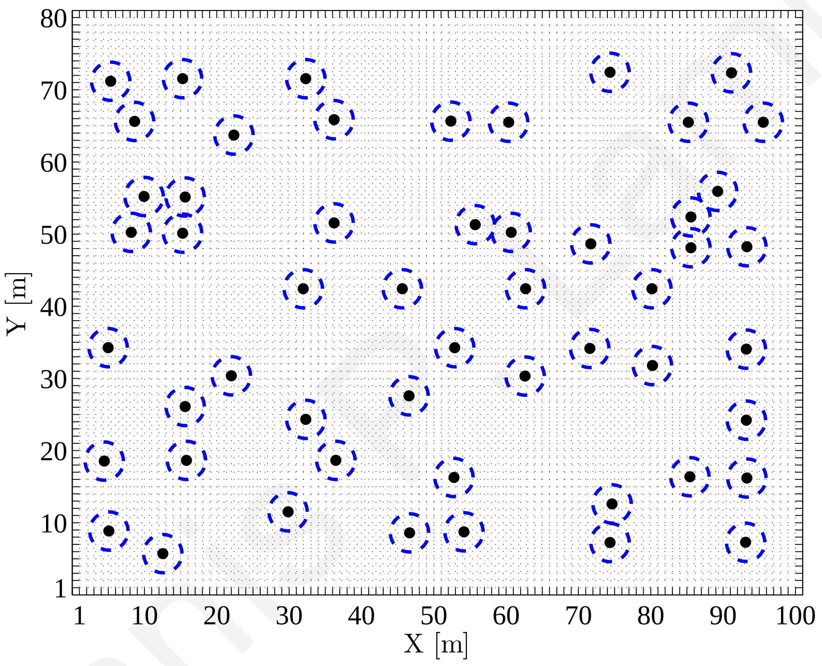
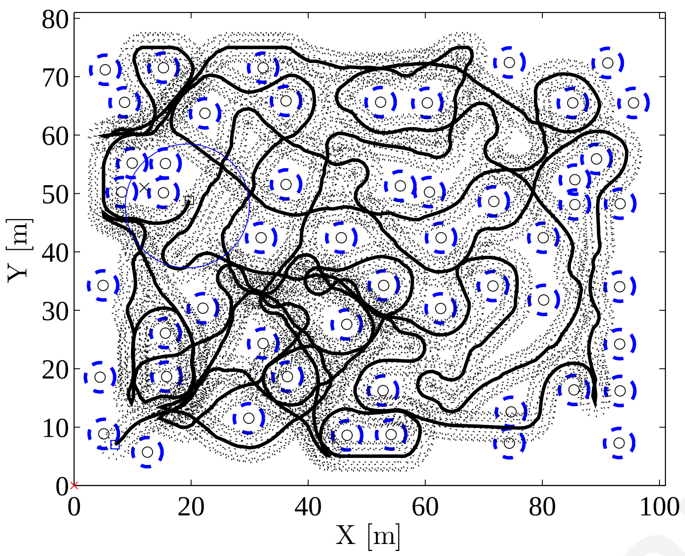
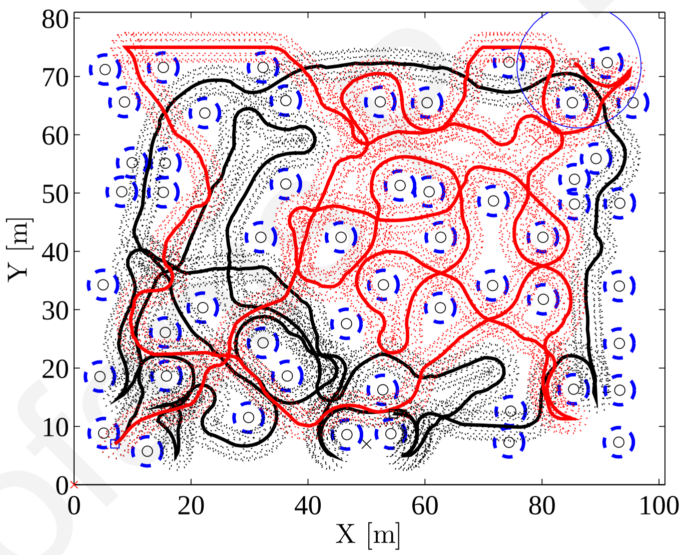


Figure 6.4: The Grid map $\mathcal{G}$ derived from a sparse stationary sensor network deployment with sensing range $r_s = 2.5m$.

In the first simulation experiment (Fig. 6.5) we show the paths that the mobile nodes follow in the sensor field shown in Fig 6.4 when the proposed complete coverage path planning algorithm (see Fig. 6.3) is used. Fig. 6.5(a) shows the path followed when only one mobile node is used and tries to improve the coverage. As shown in Fig. 6.5(a), the mobile node passes through the uncovered regions of the sensor field and at the same time it tries to avoid overlapping with static nodes. The path is estimated on-line and can adapt to the sensor field changes. Fig. 6.5(d) shows that the coverage was improved to 95% after 1487 steps. Figures 6.5(b) and 6.5(c) show the paths followed by two and three mobile nodes respectively in order to improve the coverage. These figures illustrate how the mobiles collaborate in order to improve the coverage of the field; all mobiles typically avoid the paths of each other and they also avoid stationary sensors. This collaboration is also indicated in Fig. 6.5(d) which shows that one mobile node needs 1487 steps in order to improve the coverage of the sensor field to 95% while 719 and 470 steps are needed by a group of two and three mobile nodes respectively.



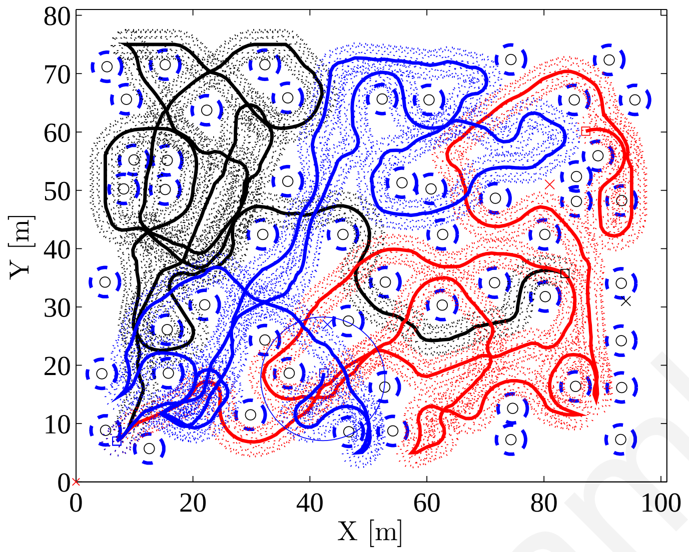
(a) Path followed using one mobile node. After 1487 moving steps 95% of the sensor field is covered.



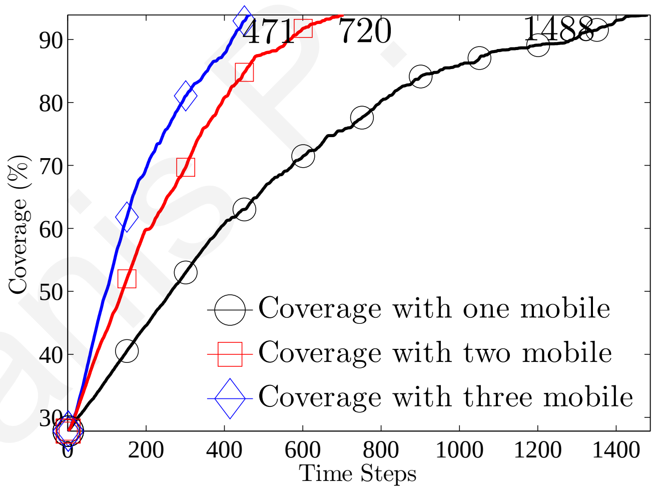
(b) Path followed using two mobile nodes. After 719 moving steps for each node 95% of the sensor field is covered.

Figure 6.5: The paths of the mobile nodes and their corresponding coverage over the WSN using the proposed path planning algorithm. The path of each mobile node is indicated by a different color

Fig. 6.6 shows the paths followed using two other moving patterns, referred to as the "random" movement and the "standard" movement. In the random movement, the next position of a mobile node is $\rho = 1m$ away from the previous one and at random heading direction $0 \leq \theta \leq 2\pi$. In the standard movement the mobile scans exhaustively the entire field using an S-shaped pattern. The standard movement is based on the so-called zamboni coverage pattern [131]. Fig. 6.6(a) shows typical trajectories of



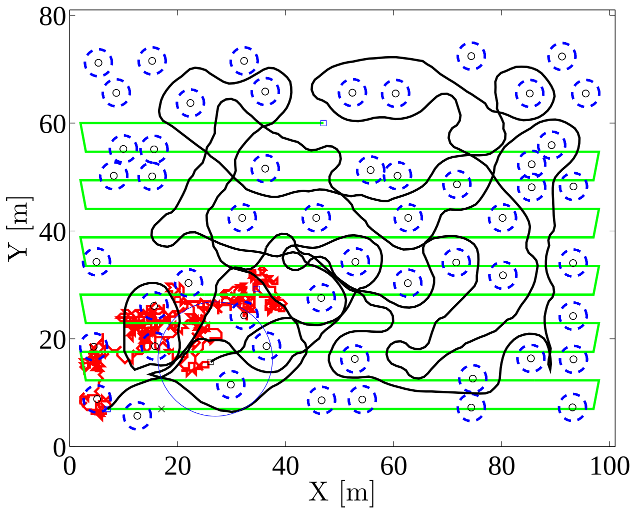
(c) Path followed using three mobile nodes. After 470 moving steps for each node 95% of the sensor field is covered.



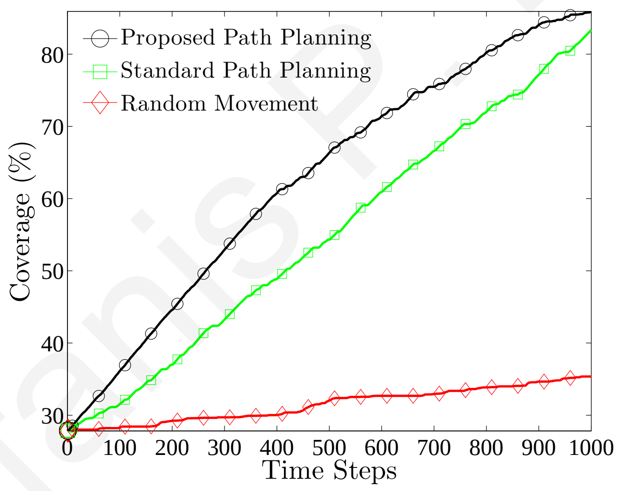
(d) The coverage improvement of the sensor field.

Figure 6.5: The paths of the mobile nodes and their corresponding coverage over the WSN using the proposed path planning algorithm. The path of each mobile node is indicated by a different color(cont.)

the three algorithms. Fig. 6.6(b) shows the coverage improvement over time for the path followed using the proposed path planning algorithm, the path followed using the "standard" movement and the path followed by the "random" movement. The ability of the proposed path planning algorithm to navigate the mobile node to the uncovered areas and avoid overlapping with the stationary sensors results in a faster rate of coverage improvement as indicated by the slope of the graphs in Fig. 6.6.



(a) Paths followed by a mobile node over the WSN for the three moving patterns.



(b) Coverage improvement for the three moving patterns.

Figure 6.6: Comparison of the coverage improvement for the three moving patterns. The path followed using the proposed coverage path planning algorithm is indicated by the black color, the path followed using the "standard" planning is indicated by green color and the path followed by the "random" movement is indicated by red color.

Finally, to demonstrate the effectiveness of the proposed path planning algorithm in more general scenarios, we consider a sensor field with 100 sensors randomly deployed in a 100 by 80 meters area. We consider 100 random field deployments and assume a group with $M = 2$ mobile sensors. Fig. 6.7 shows the average coverage accomplished after each step for the proposed algorithm, as well as the "random" and the "standard" algorithms. For the "standard" algorithms it is assumed that the two mobiles achieve "perfect" collaboration in the sense that they move in parallel and there is no overlap between their sensing ranges. Fig. 6.7 shows that the proposed algorithm is significantly better than both the "random" and "standard" algorithms.

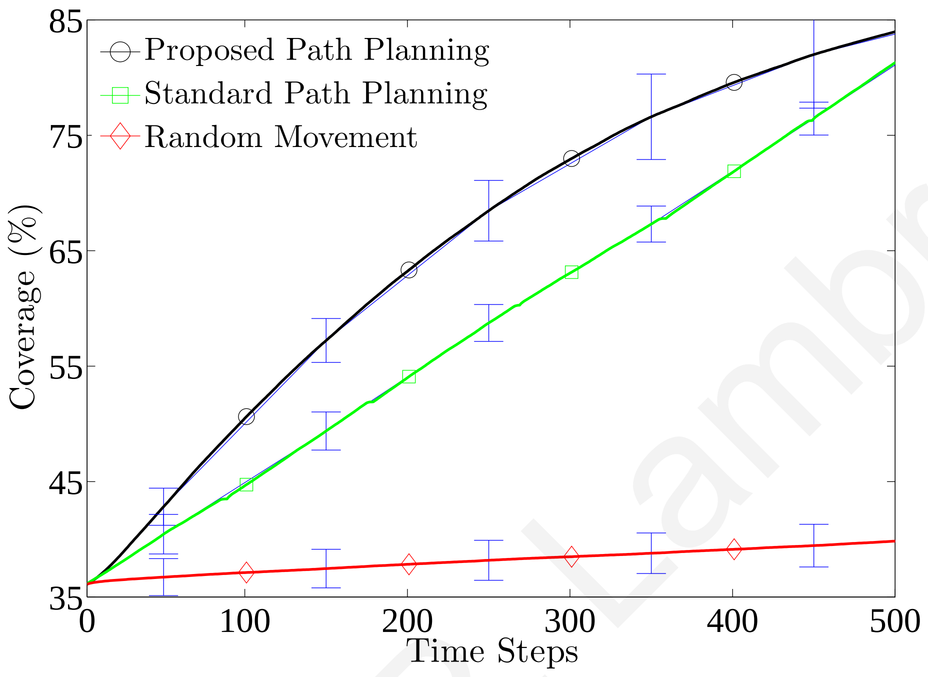


Figure 6.7: The average coverage accomplished over 100 sensor fields with 100 randomly distributed stationary sensors by a group of two mobile nodes.

6.6 Conclusions

In this chapter we propose a method to improve the area coverage in sparse WSNs with static nodes, using a group of collaborating mobile nodes. The proposed algorithm is simple, distributed and dynamic and allows each mobile to autonomously navigate through the sensor field in order to improve the area coverage. The path is estimated on-line and can adapt to the sensor field changes. Note that such algorithms are most appropriate in the context of WSNs. We have shown several results using a sparse stationary WSN deployment as well as randomly deployed sensor fields. As indicated by the obtained results, the proposed algorithm is able to significantly improve the area coverage compared to "random" and "standard" path planning algorithms. In the next chapter, we will further investigate the different parameters that play a role in the proposed path planning algorithm and particularly parameters associated with the information exchange between mobile nodes.

Chapter 7

Distributed Collaborative Path Planning with Minimal Communication

7.1 Summary

This chapter investigates the information exchange between mobile nodes in the distributed on-line path planning algorithm for complete coverage in WSNs proposed in the previous chapter. The main objective of this chapter is to investigate different collaboration schemes between the sensor nodes such that the amount of information that needs to be exchanged between nodes is reduced without significant loss of the area coverage performance.

7.2 Introduction

In the previous chapter we presented the development of an efficient distributed collaboration scheme for a team of autonomous mobile sensor nodes which enables them to navigate through a sparse sensor network with stationary nodes and improving area coverage. Each mobile sensor node autonomously plan its trajectory on-line based on local information. This local information consists of the mobile node's beliefs and measurements as well as information collected from the nodes, stationary or mobile, that are in a neighborhood around the mobile. This information, which represents the state of the environment, is stored in each mobile's memory and it is locally updated. This information will be refereed as the mobile's "cognitive map".

The main contribution of this chapter is that it investigates possible ways for efficient collaboration by which the information exchange or the communication cost is sufficiently minimized yet the performance in terms of area coverage is no significantly affected. Different type of information to be exchanged as well as the timing are being considered and different conclusions based on the communication ranges are derived.

In this chapter we use exactly the same notation, modeling assumptions and information structure needed by the mobile nodes to run the path-planning algorithm as in the previous chapter. The only modification made is the introduction of the parameter r_z which represents the radius of the search area in the cognitive map where the dynamic target (coverage hole) is evaluated using the zoom algorithm. This enable us to distinguish between the communication range r_c for information transmission and the range r_z where the coverage hole is found using the information available in the corresponding submatrix $\mathcal{D}_{\bar{r}_z}(\bar{\mathbf{x}}_m(k))$ of the Grid G_k of the mobile node. Also, note that $\bar{\mathbf{x}}_m(k)$ is the discretized coordinates corresponding to position of the mobile sensor m during the k-th step, $\mathbf{x}_m(k)$.

We plan to investigate the type of information to be exchanged as well as the timing. The mobiles may exchange their entire cognitive map or just a small part of the map, or they can only exchange their target locations. The information exchange can occur at every step or it can occur periodically (every k time steps) or once the mobiles move sufficiently close to each other. In the following section we try to minimize the communication cost (information exchange) without seriously affecting the system's performance (area coverage). Our aim is to better understand the tradeoff involved between information exchange and area coverage. Towards this goal, we investigate schemes that uses the minimum amount of information exchange under certain communication conditions that enhance mobile cooperation and area coverage.

7.3 Simulation Results

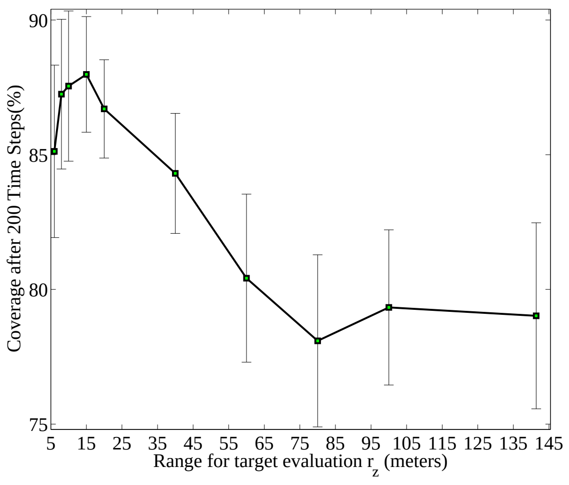
This section presents some simulation results in order to compare performance and analyze the parameters of the collaboration mechanism. Our aim is to reduce the amount of information needed to be exchanged between sensor nodes without serious loss of the area coverage performance.

Unless otherwise stated, all experiments refer to Monte Carlo simulations of 100 WSN deployments. Each WSN is deployed in a $\mathcal{A} = 100m \times 100m$ square region and consists of 100 randomly placed stationary sensor nodes with sensing radius $r_s = 4m$. A set of 5 mobile sensor nodes is used in order to improve the area coverage of each WSN. The mobile nodes maneuverability parameters are set to $\rho = 1m$ and $\phi = 30°$ while for every decision $\nu = 10$ candidate next positions are considered. Moreover the weights are set to $w_t = w_s = 0.5$ and it is assumed that the forgetting factor in (6.3) is set to $f = 1$. Finally it is assumed that when a mobile node receives a message with position coordinates the received payload data is $2 \times b$ bits and when it receives messages concerning a "cognitive map", the received payload data for each matrix element is b bits, i.e., for the entire matrix $X \times Y \times b$ bits are needed. For the simulations it is assumed that $b = 32$.

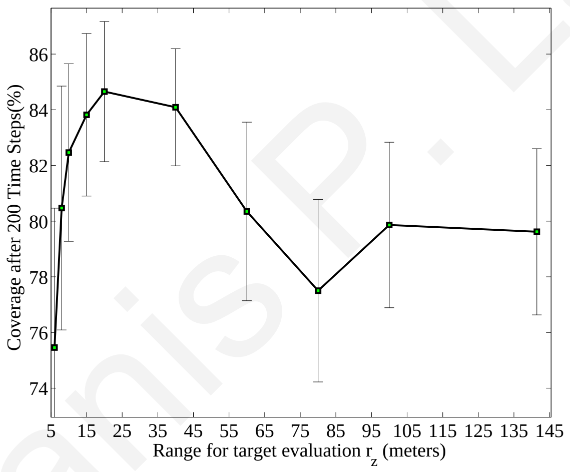
In the first simulation experiment (Fig. 7.1(a)) we would like find the optimum search area range r_z where the dynamic target (coverage hole) is evaluated in order to improve cooperation and area coverage assuming that the mobile nodes can exchange-merge their maps using global inter-mobile communication ($r_c = \sqrt{2\mathcal{A}}$)). Note that r_z range indicates the range where the coverage hole is found. Fig. 7.1(a) shows the average coverage succeeded by mobiles after 200 moving steps. It turns out that the optimal r_z is about $15m$ (or $15\% \times \sqrt{\mathcal{A}}$). If r_z is smaller then collaboration with static nodes is poor resulting in less coverage. If r_z is bigger than optimal then collaboration is reduced because mobile nodes tend to move towards the same locations (larger coverage holes) for long times which sometimes results in paths overlapping. Furthermore, if targets are found in larger areas (larger r_z), larger holes may dominate and as a result smaller holes close to the mobile's path are ignored.

Fig. 7.1(b) shows the same scenario when $r_c = r_z + r_s$. Note that since a mobile is searching for a hole in an area with radius r_z, it needs to have the most accurate state of the field in that range. Any sensor that is r_s meters outside r_z, covers some area that falls inside the mobile's search area, therefore such information may improve the collaboration between the static and the mobile sensors. Static sensors located at a distance greater than $r_z + r_c$ do not provide any information that can be used when searching for the target. Therefore, to achieve the best possible collaboration between static and a mobile, it is necessary that $r_c \geq r_z + r_s$ and in the sequel we will use this as a lower bound for the communication range. Even for this case, it turns out that again the optimal r_z is about $15 - 20m$ (or $15 - 20\% \times \sqrt{\mathcal{A}}$) for the same reasons mentioned above, although bigger r_c (i.e., $r_c > r_z + r_s$) can achieve better results due to better collaboration between the mobiles. After extensive evaluations we found out that the value of $r_z \approx 15m$ achieves very good results and is not affected by the number of nodes available (mobiles or static) in the WSNs. Also note that a smaller r_z is advantageous since it implies less information is needed for the coverage hole estimation.

In the second simulation experiment (Fig. 7.2) we would like to investigate how the coverage performance is affected by the communication range r_c. The communication range r_c defines the maximum distance between mobiles such that they are able to communicate and exchange/merge their maps. In this scenario it is assumed that if mobiles are in communication range they exchange their maps at each moving step. Also, r_z was set to $r_z = 15m$. Fig. 7.2(a) shows the average coverage achieved by mobiles after 200 moving steps. It turns out that in this setup there exists a critical transmission range $r_c = 35m$ where above this range there is no significant improvement in area coverage but rather resulting in communication waste (see Fig. 7.2(b)). The transmission range $r_c = 35m$ indicates the value where the "cognitive maps" of the mobiles are almost the same all the time since the mobiles are exchanging/merging their



(a) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes after 200 moving steps when $r_c = \sqrt{2} \times 100$.

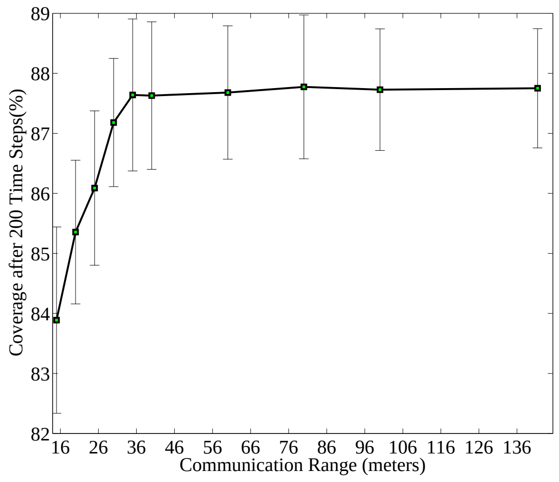


(b) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes after 200 moving steps when $r_c = r_z + r_s$.

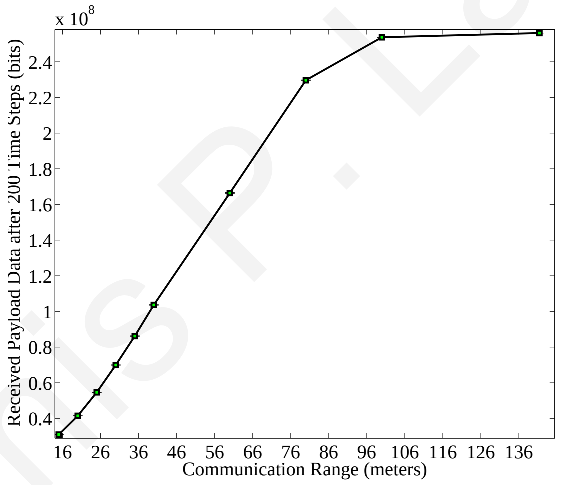
Figure 7.1: Evaluation of the dynamic target optimum range r_z.

maps in multi-hop manner (say at time k mobile m_1 can communicate with m_2 and they merge their maps, then, at $k+1$, due to the large communication range, m_2 may come to communication range with m_3 and during the information exchange, m_2 will pass to m_3 the information that it received from m_1). This simulation indicates that there is no point to have $r_c > 35$ since there will be higher communication cost without any benefit in the achieved area coverage. Furthermore, Fig. 7.2 shows the tradeoff between coverage and communication cost; to achieve a 3.5% coverage improvement, using a scheme where the entire matrix is exchanged at every step, requires a heavy communication cost.

In the next simulation experiment (Fig. 7.3(a), 7.3(b)) we study the case when the map exchanges between mobile nodes are again periodic but less frequent. The



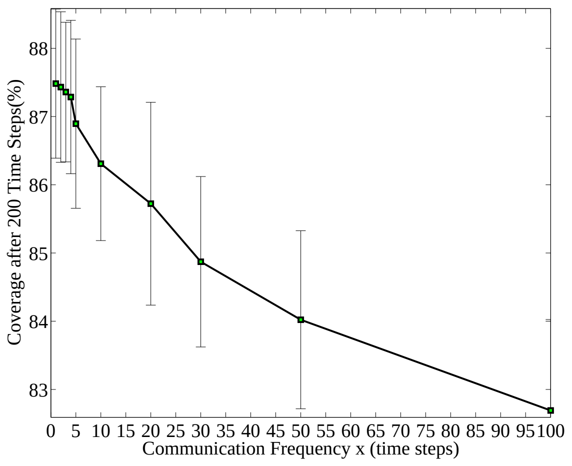
(a) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes after 200 moving steps



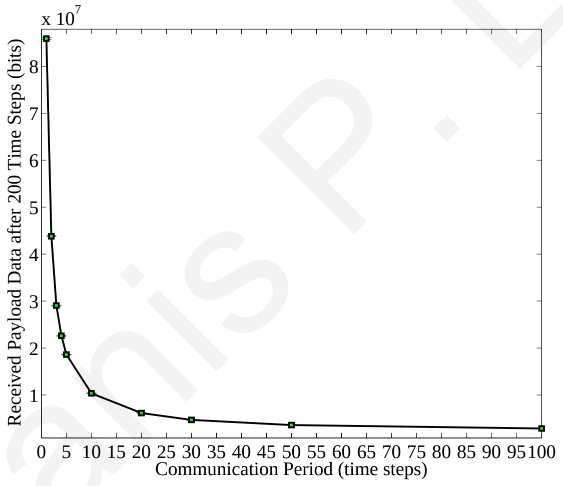
(b) Average total payload received by each mobile node after 200 time steps from other nodes

Figure 7.2: Evaluation of the critical transmission range of r_c

communication range was set to $r_c = 35m$ and the target evaluation range was set to $r_z = 15m$. In this scenario if mobile nodes come into communication range for the first time they exchange their maps and as long as they remain in communication range for more steps (continuously) they exchange their maps once every k time steps where $1/k$ indicates the communication frequency. As shown from Fig. 7.3 if the maps are updated less frequently (not continuously, i.e every 5 time steps) there is no serious loss of performance in terms of area coverage, however, the communication cost is significantly improved (see Fig. 7.3(b)). It is worth to mention that a soft threshold, indicating how much the map of the mobile has been changed, could be defined in order to find out when each mobile node must share (transmit) its “cognitive” map to other mobiles when they come into communication range. In other words, one can use event-driven exchange (rather than time driven) to further reduce the communication cost.



(a) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes after 200 moving steps when $r_c = 35$



(b) Average total payload received by each mobile node after 200 time steps from other nodes when $r_c = 35$

Figure 7.3: Evaluation of the communication period when the mobiles are into communication range

In the previous experiments, we have studied and evaluated how the system performance is affect by two important parameters, the communication range r_c where the "cognitive" maps are exchanged and the maximum range for the dynamic target evaluation r_z. In the following simulation experiment we investigate the performance when less information is exchanged. Assuming the distributed path planning scenario (as previously) with $r_c = r_z + r_s$ we study the area coverage improvement when five different communication schemes (CS) are applied:

CS1: If mobile nodes are in communication range r_c, at every step, they exchange their *entire maps* P_k^m and also exchange their *positions* $\mathbf{x}^m(k)$ and dynamic *target coordinates* $\mathbf{x}_t^m(k)$.

CS2: If at step k, the mobile nodes come into communication range when they were out of range in step $k-1$, they exchange their *entire maps* P_k^m and their *positions* $\mathbf{x}^m(k)$ and *target coordinates* $\mathbf{x}_t^m(k)$. If they are in r_c at time $k-1$ then they exchange their *sub-matrices* $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ corresponding to their r_c range and their *positions* $\mathbf{x}^m(k)$ and dynamic *target coordinates* $\mathbf{x}_t^m(k)$.

CS3: If at step k, the mobile nodes come into communication range when they were out of range in step $k-1$, they exchange their *entire maps* P_k^m and their *positions* $\mathbf{x}^m(k)$ and *target coordinates* $\mathbf{x}_t^m(k)$. If they are in r_c at time $k-1$ then they only exchange their *positions* $\mathbf{x}^m(k)$ and dynamic *target coordinates* $\mathbf{x}_t^m(k)$.

CS4: If at step k, the mobile nodes come into communication range when they were out of range in step $k-1$, they exchange their *entire maps* P_k^m *only* if their coverage exceeds a predefined threshold τ_C since the last time [1] communicated. If they are in r_c at time $k-1$ then they only exchange their *positions* $\mathbf{x}^m(k)$ and dynamic *target coordinates* $\mathbf{x}_t^m(k)$. In this scheme the P_k^m is exchanged only in an event driven way.

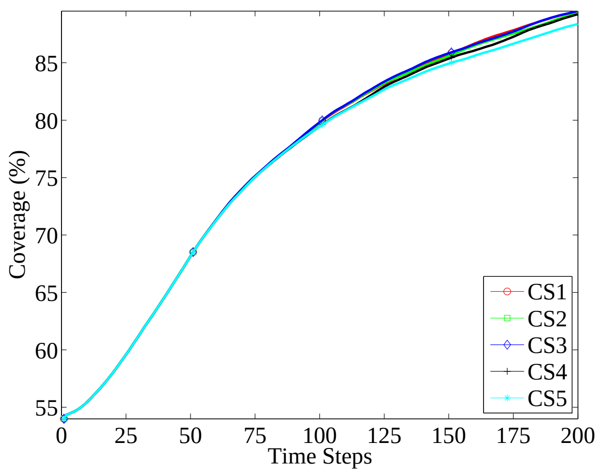
CS5: If mobile nodes are in communication range r_c they *only* exchange their *positions* $\mathbf{x}^m(k)$ and dynamic *target coordinates* $\mathbf{x}_t^m(k)$.(they never exchange either the P_k^m or the $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$)

Note that the merging policy that each mobile is utilized when it receives the entire or a part of its neighboring mobiles maps is based on a cell value maximization rule. In other words, each cell in the map of the mobile is updated to the maximum value of the corresponding cells of the other maps received. This policy is not very optimistic, however avoids erroneous updates that could occurred when mobiles come in communication multiple times. More accurate merging policies would require each cell to be associated with a time-stamp and mobile id which will require excessively additional memory [2]. Also recall that for all communication schemes, the mobiles also receive the position coordinates of the stationary sensors in their communication range r_c.

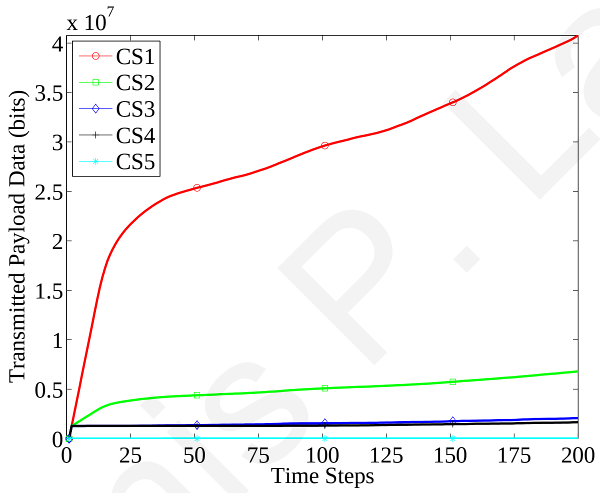
Fig. 7.4(a) shows the average coverage improvement at each time step when $r_c = r_z + r_s = 19m$ and five mobiles are collaborating using the information described in the communication schemes above. The comparison between CS1,CS2, CS3 and CS4 (see figures 7.4(a) and 7.4(b)), shows that although the communication cost in CS4 with τ_C=5% (or CS3 in general) is seriously minimized, the area coverage improvement is only slightly affected. Moreover in the case of the CS5, it seems that the coverage performance is reduced by about 1% (compared to CS1) but the communication cost is significantly reduced. It is worth to note that if $r_c = 35m$ then the above communication schemes will result in almost the same area coverage performance (differences

[1]Each mobile must keep in its memory a communication matrix where it tracks with which mobiles was in communication during the previous step as well as what was its coverage value since the last time it communicates with another mobile.

[2]Consider the case when a cell (i,j) has been covered by a mobile m once (e.g. $P^m(i,j) = 0.5$), after the mobile m communicate that value to another mobile $m+1$, and the latter mobile communicate again with mobile m afterwards without covering the cell (i,j). If other rule is used (e.g. $P^m(i,j) = 1-(1-P^m(i,j))\cdot(1-P^{m+1}(i,j)) = 0.75$) a duplicated update might occurred.



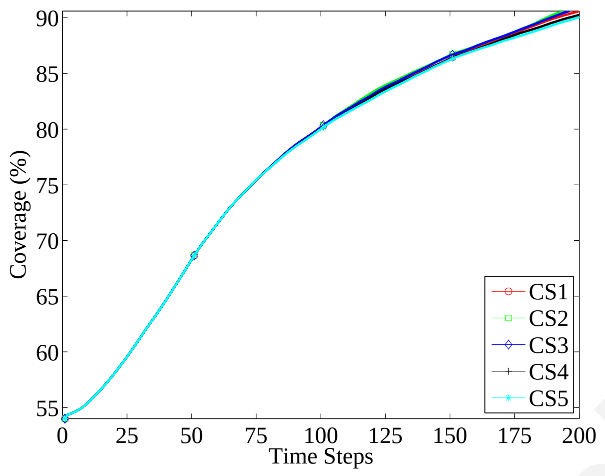
(a) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes



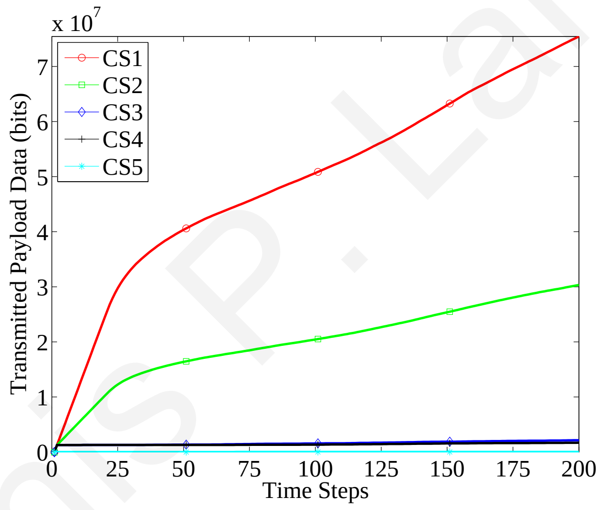
(b) Average total payload received by each mobile node

Figure 7.4: Comparison of different "cognitive map" merging communication schemes when $r_c = 19m$

between graphs in Fig. 7.5(a) are minimized), because as mentioned earlier if $r_c = 35m$ mobiles nodes are communicating with each other most of the times and thus by just exchanging their positions and target coordinates they always have almost accurate maps. The final simulation indicates that when *global inter-mobile communication is available ($r_c \geq 35m$) CS5 seriously minimizes the communication cost without serious loss in the area coverage performance.* On the other hand when *local communication is used ($r_c \leq 19m$) CS4 (or CS3) sufficiently minimizes the communication cost without affecting the area coverage performance compared to CS1.*



(a) Average coverage improvement over 100 sensor fields by a set of 5 mobiles nodes



(b) Average total payload received by each mobile node

Figure 7.5: Comparison of different "cognitive map" merging communication schemes when $r_c = 35$

7.4 Conclusion

The objective of this work is to develop a collaborative architecture for WSNs consisting of a large number of stationary nodes and a few mobile nodes. We investigate energy efficient collaboration mechanisms such that communication cost is sufficiently minimized without serious loss of the area coverage performance. Both global and local communication ranges are studied and different conclusions based on the communication ranges are derived.

Chapter 8

Distributed Collaborative Path Planning for Coverage and Event Source Detection in WSNs

8.1 Summary

This chapter investigates the collaborative area monitoring path planning problem in stationary sensor networks using multiple cooperative mobile sensor nodes to patrol the region of interest and locate events. We present the development of an architecture where a set of mobile sensors will collaborate with the stationary sensors in order to reliably detect and locate events. The main idea of this collaborative architecture is that the mobile sensors should sample the areas that are least covered (monitored) by the stationary sensors. Furthermore, when stationary sensors have a "suspicion" that an event may have occurred, they report it to a mobile sensor that can move closer to the suspected area and can confirm whether the event has occurred or not. An important component of the proposed architecture is that the mobile nodes autonomously decide their path based on local information. This approach is appropriate in the context of wireless sensor networks since it is not feasible to have an accurate global view of the state of the environment. We present extensive simulation results indicating the effectiveness of the proposed algorithm.

8.2 Introduction

In the previous chapters we present the development of a distributed collaborative framework for complete area coverage where a small set of mobile nodes collaborate with the stationary sensor nodes and with each other in order to improve the area coverage. However the collaborative framework developed does not consider the measurements that sensor nodes received when sampling the environment. In this chapter we incorporate this information to the path planning problem and also sufficiently model sensor measurements by considering the noise of the environment and the probability of false alarms.

This chapter considers the problem of monitoring a large area using wireless sensor networks in order to detect and locate events. In this context, we assume that an event emits a signal that is propagated in the environment. The sensors capture attenuated and noisy measurements of the signal and the objective is to *reliably* detect the presence of the event and estimate its position. By reliably we mean that we would like to minimize the probability of miss event (an event that remains undetected) subject to a constraint on the probability of false alarms (the sensors report an event due to noise). Note that in many applications false alarms are as bad (if not worse) than missed events. In addition to the incurred cost for sending response personnel to the area of the event, frequent false alarms may lead the users to ignore *all* alarms and as a result even detected events will go unnoticed.

To achieve reliable detection in a large area, it is necessary to deploy a huge number of sensors which with the current technology implies a prohibitive cost [130]. For example consider a lake to be monitored for events (an event can be a boat that spills a substance in the lake that changes the water turbidity). If the lake has an area of 20km$\times$20km, and we assume that each sensor has a reliable sensing range (detection range) r_d=10m, then the number of sensor nodes needed to monitor the entire lake is of the order of 10^6 which with today's technology implies a prohibitive cost.

Given that it is infeasible to reliably cover the entire area with stationary nodes, in this chapter we investigate an alternative way of monitoring the area using several stationary and some mobile sensor nodes that collaborate in order to improve the area coverage and to detect an event as fast as possible. The main idea is that the mobile nodes will collaborate with the stationary nodes (and with each other) in order to sample areas that are least covered by the stationary nodes and navigate through the sensor field autonomously searching for event sources.

This chapter investigates the use of signal processing techniques in the path planning of mobile agents for improving the area monitoring in the context of WSNs. The main contribution of this chapter is that it investigates a family of path planning algorithms and proposes a distributed algorithm that is fairly simple, it relies only on local information (i.e., information collected from the mobile's neighborhood) and can achieve very good performance. The strategy used by each mobile is based on receding horizon optimization where at every step, the mobile node tries to move towards the least covered area and at the same time it avoids areas covered by other nodes.

The chapter is organized as follows. Section 8.3 describes the model that has been adopted and the underlying assumptions. Section 8.4 presents a family of distributed path planning algorithms that can be utilized by each mobile sensor in order to navigate through the sensor field. Section 8.5 presents the dynamic target estimation and allocation strategy used for coverage, event detection and collaboration purposes. Section 8.6 presents several simulation results using various sensor fields with randomly deployed sensor nodes. The chapter concludes with Section 8.7.

8.3 Model Description and Problem Formulation

8.3.1 The Environment

The environment is represented as a rectangular area $\mathcal{A} = \mathcal{R}_x \times \mathcal{R}_y$. We consider a set $\mathcal{S}$ with $S = |\mathcal{S}|$ static sensor nodes that are randomly placed in the area $\mathcal{A}$, at positions $\mathbf{x}_i = (x_i, y_i)$, $i = 1, \cdots, S$. In addition, we assume that a set $\mathcal{M}$ of $M = |\mathcal{M}|$ mobile sensor nodes are available and their position after the k-th time step is $\mathbf{x}_i(k) = (x_i(k), y_i(k))$, $i = 1, \cdots, M$, $k = 0, 1, \cdots$. For notational convenience, we define the set of *all* sensor nodes $\mathcal{N} = \mathcal{S} \cup \mathcal{M}$ and re-index all mobile nodes as $m = S + 1, \cdots, S + M$. It is assumed that all sensors know their location through a combination of GPS and localization algorithms. Furthermore, it is assumed that all sensors can reach the fusion center (commonly referred to as sink in the WSN literature) using multihop communication.

In addition, we consider a set $\mathcal{E}$ with $E = |\mathcal{E}|$ stationary non-overlapping event sources[1] that are randomly placed in the environment at positions $\mathbf{e}_j = (x_j^e, y_j^e)$, $j = 1, \cdots, E$.

Next, we also define the neighborhood of a sensor s as the set of all sensors that are located at a distance less than or equal to r_c from the mobile. In other words, the neighborhood of sensor $s \in \mathcal{N}$ is the set of all sensors that are in the disc centered at $\mathbf{x}_s$ with radius r_c.

$$\mathcal{H}_{r_c}(s) = \{j \ : \ \|\mathbf{x}_s - \mathbf{x}_j\| \leq r_c, \ j \in \mathcal{N}, j \neq s\} \tag{8.1}$$

for all $s = 1, \cdots, S + M$. If r_c is the communication range of the sensor, then $\mathcal{H}_{r_c}(s)$ defines all sensors that are one hop away from that node. In general however, one can define larger neighborhoods that include sensors that are two or more hops away.

[1]Sources with non-overlapping footprints.

8.3.2 Sensor Model

We assume that each event source $j \in \mathcal{E}$ emits a constant signal V_j in the surrounding environment. As we move away from the source, the measured signal is inversely proportional to the distance from the source raised to some power $\alpha \in \mathbb{R}^+$ which depends on the environment. As a result, the t-th measurement of sensor $i \in \mathcal{N}$ is given by

$$z_{i,t} = \min\left\{V_{sat}, \sum_{j=1}^{E} \frac{V_j}{r_{ij}^{\alpha}}\right\} + w_{i,t} \tag{8.2}$$

where V_{sat} is the maximum measurement which can be recorded by a sensor, r_{ij} is the radial distance of sensor i from the event source j,

$$r_{ij} = \sqrt{(x_i - x_j^e)^2 + (y_i - y_j^e)^2}, \tag{8.3}$$

and $w_{i,t}$ is additive Gaussian noise with zero mean and variance σ_i^2. A sensor node reports that it has reliably detected an event if the measurement[2] it receives is greater than the detection threshold τ_d. This threshold is determined in a way such that the probability of false alarm is less than a given constraint p_{fa}. This calculation can be done as in [130] and references therein, but for the purposes of this chapter, it is assumed that this threshold is given. This threshold together with V_j define a disc around the source (footprint of the source) where, if sensor i is located inside this disc, then it will be alarmed (i.e., its measurement will be above the threshold τ_d) with high probability, at least 0.5. Given the model (8.2), the radius of the disc is given by

$$r_d = \sqrt[\alpha]{\frac{V_j}{\tau_d}}. \tag{8.4}$$

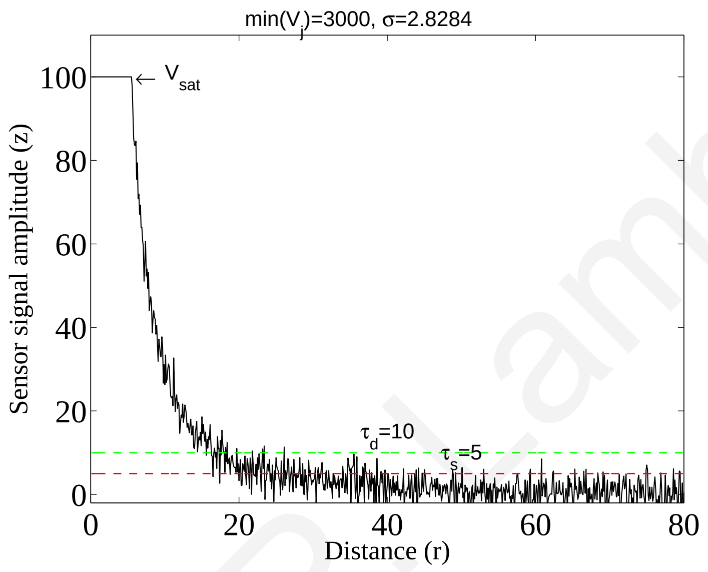
By symmetry, there exists a disc around every sensor with radius r_d where if a source exists it will cause the sensor to be alarmed with high probability (at least 0.5). This is referred to as the detection (sensing) range of the sensor and it is assumed known. For the purposes of this chapter, if the event occurs within this disc, then we say that it is reliably detected. Furthermore, we assume that an event is detected by the network if at least one sensor (stationary or mobile) detects the event but other fusion rules can also be used at the fusion center.

Similarly, we assume that we are given a "suspicion" threshold $\tau_s < \tau_d$ such that if the measurement of the sensor i, $\tau_s \leq z_i \leq \tau_d$, then sensor i does not report a detection, however, it may report that it "suspects" that there may be an event around its area. Note that τ_s defines a disc around the sensor with radius $r_s > r_d$, thus a node may report the suspicion if the event exists in the "donut" that is formed by the suspicion disc when the detection disc is removed. The event suspicion may be used in different ways. It can be reported to the sink which may fuse the information from several sensors or it can be given to a nearby mobile node which will collaborate with the stationary sensors in order to move closer to the suspected event area to confirm the

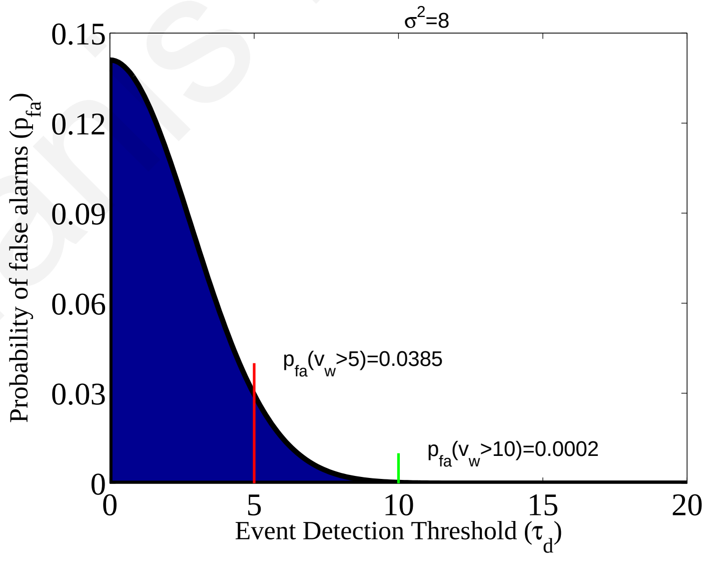
[2]Alternatively one could use the average measurement or simply assume smaller noise variance.

existence or not of the event. In this chapter, the suspicion will be used as in the latter example.

The received signal amplitude with respect to sensor – source distance for a sensor node is illustrated in figure 8.1(a). Figure 8.1(b) illustrates the probability of false alarms with respect to event detection threshold τ_d.



(a) Sensor signal amplitude z with respect to sensor – source distance r.



(b) Probability of false alarms $p_{fa}(v_w > \tau_d) = 1 - \mathbb{F}(\tau_d; 0, \sigma^2)$ with respect to event detection threshold τ_d. $\mathbb{F}$ represents the Normal (Gaussian) cumulative distribution function

Figure 8.1: Sensor signal amplitude and probability of false alarms for parameters: $V = 3000, V_{sat} = 100, \sigma^2 = 8, \tau_d = 10$ and $\tau_s = 5$.

8.3.3 Objectives

The aim of this chapter is to plan the path of the mobile nodes in order to achieve certain objectives. As already mentioned, the sensor network environment is constantly changing (sensors may fail or be carried away) thus it is unrealistic to expect that a central controller will have all necessary information to predetermine the paths that each mobile should follow, thus we will consider dynamic path planing algorithms that use locally available information to determine where to go next.

In this type of problems, one can define different objectives that may result in different strategies. A possible objective is to detect and locate events as fast as possible. For this objective, a candidate strategy for the mobile nodes is to quickly move towards large uncovered areas since, if there exists an undetected event source, it is most likely located in those areas. Another possible objective is to maximize the area coverage (minimize the average probability that an event source remains undetected). In this case, a good candidate strategy for the mobile is to navigate through areas not covered by other sensors (stationary or mobile). As will be shown in the sequel, it turns out that a combination of these two strategies can achieve very good results.

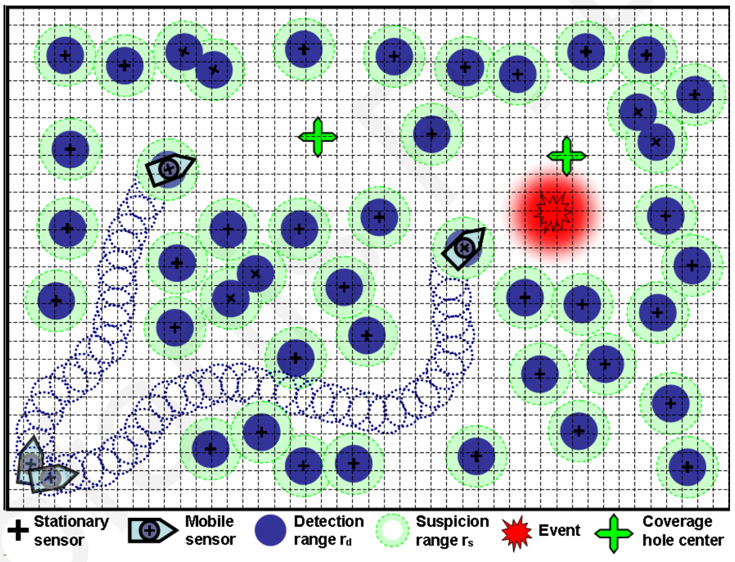


Figure 8.2: Environment Model.

To make the concept of area coverage more concrete, we divide the field area in small squares with side da. In other words, we transform the sensor field area $\mathcal{A}$ into a grid $\mathcal{G}$ of size $X \times Y$, where $X = \lceil R_x/da \rceil$ and $Y = \lceil R_y/da \rceil$ (see Fig. 8.2). Thus, we assume that any sensor $s \in \mathcal{N}$ is located in the cell $\bar{\mathbf{x}}_s = (i,j)$, $i = \lceil x_s/da \rceil$ and $j = \lceil y_s/da \rceil$ (i.e., $\bar{\mathbf{x}}_s$ is the discretized coordinate corresponding to $\mathbf{x}_s$). Furthermore, we assume that a sensor located in the cell $\bar{\mathbf{x}}_s$, depending on the detection range $\bar{r}_d = \lceil r_d/da \rceil$, covers a neighborhood of cells $\mathcal{D}_{\bar{r}_d}(\bar{\mathbf{x}}_s)$,

$$\mathcal{D}_{\bar{r}_d}(\bar{\mathbf{x}}_s) = \left\{(p,q) : \ (p-i)^2 + (q-j)^2 \leq l_d^2, \ \bar{\mathbf{x}}_s = (i,j)\right\}. \tag{8.5}$$

We associate with the grid $\mathcal{G}$, an $X \times Y$ matrix G_k, $k = 0, 1, \cdots$, where each element of

G_k captures our "confidence" that if an event occurs in the corresponding area of the field, it will be detected by the sensor network. If the (i,j)-th cell falls in the detection range of a static sensor, then the corresponding $G_k(i,j) = 1$, for all k (here we use the fact that a stationary sensor may perform a long run average of its measurements and thus the probability of detecting a source in its detection range goes to 1). Otherwise, initially (at $k = 0$) $G_k(i,j) = 0$ and as the mobile nodes move around, if they sample areas not covered by the static sensors, then our confidence increases and continues to increase as the mobiles take more samples. Furthermore, if a cell has not been sampled for some time, then it is possible that our confidence will be reduced. Thus at every step, we use the following updating rule for every element of matrix G_k.

$$G_{k+1}(i,j) = \begin{cases} 0.5 \cdot G_k(i,j) + 0.5, & \text{if } (i,j) \in D_{\bar{r}_d}(\bar{\mathbf{x}}_s) \\ f \cdot G_k(i,j), & \text{otherwise} \end{cases} \tag{8.6}$$

where $s \in \mathcal{N}$ and $0 \leq f \leq 1$ is the "forgetting" factor. This factor can be used to account for the physics involved with the phenomena of the events that are being monitored. For example, it can account for sources that are active only during a window of time of the observation interval or sources that turn on and off at various time instances. Consequently, *coverage* is defined as

$$C_k = \frac{1}{X \times Y} \times \sum_{\substack{1 \leq i \leq X \\ 1 \leq j \leq Y}} G_k(i,j). \tag{8.7}$$

8.3.4 Mobile Sensor Node Model

The state of the i-th mobile node at time k is denoted by $\upsilon_i(k)$ which is comprised of two components, $\upsilon_i(k) = [\mathbf{x}_i(k), \theta_i(k)]$. As already mentioned $\mathbf{x}_i(k)$ is the node's position and $\theta_i(k)$ is its orientation (heading direction). The mobile nodes move at some constant speed ψ and make path planning decisions at discrete time intervals, which means that each mobile node follows a straight line of length $\rho = \|\mathbf{x}_i(k+1) - \mathbf{x}_i(k)\|$ when moving from $\mathbf{x}_i(k)$ to $\mathbf{x}_i(k+1)$. Moreover, we point out that this model can also include maneuverability constraints of the mobile platform using some angle ϕ which constrains the maximum allowed difference between $\theta_i(k)$ and $\theta_i(k+1)$.

Finally, we describe the information required by each mobile in order to make path planning decisions. Each mobile uses a *coverage cognitive map*, an $X \times Y$ matrix P_k^m, $m \in \mathcal{M}$ where it keeps the state of the field. Ideally P_k^m should remain $P_k^m = G_k$ at all times k, since the matrix G_k represents the accurate global state of the field which is used for the computation of the field coverage C_k. Clearly, in a dynamic environment where several sensors may accidentally move, fail or more sensors are added, it is impossible to guarantee that $P_k^m = G_k$ at all times. However, we emphasize, that the proposed algorithm, that will run by a mobile located at some $\bar{\mathbf{x}}_m(k)$, computes its next position based mainly on local information, i.e., information in the submatrix of P_k^m that corresponds to the cells $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$, where $\bar{r}_c = \lceil r_c/da \rceil$ and thus, it is sufficient to have accurate information only for the $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ submatrix. This is easily attainable since the required information can be obtained from the mobile's neighbors in $\mathcal{H}_{r_c}(m)$.

8.4 Collaborative - Distributed Path Planning

In this section we present a family of distributed path planning algorithms that can be utilized by each mobile sensor in order to navigate through the sensor field and to achieve its objectives. These algorithms are based on a receding-horizon approach and are motivated by [17]. In this family of algorithms, the mobile's controller evaluates the cost of moving to a finite set of candidate positions and moves to the one that minimizes the overall cost as described next. Before we proceed, to simplify the notation, in this section, we dropped the index for each mobile, i.e., $\mathbf{x}(k)$ refers to the position of the i-th mobile sensor, $i \in \mathcal{M}$.

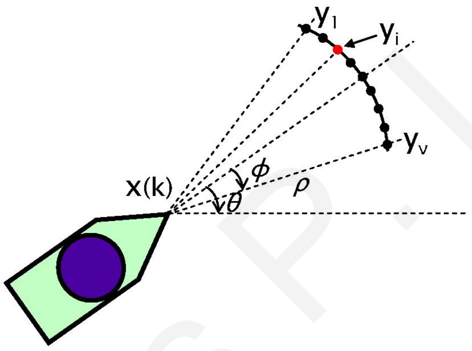


Figure 8.3: Evaluation of the mobile node's next step.

Suppose that during the kth step, the mobile node is at position $\mathbf{x}(k)$ and it is heading to a direction θ. The next candidate positions are the ν points $\mathbf{y}_1, \cdots, \mathbf{y}_\nu$ that are uniformly distributed on the arc that is ρ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$ as shown in Fig. 8.3. Note that the parameters ρ and ϕ can be used to also model the maneuverability constraints of the mobile platform. At the kth position, the mobile node evaluates a cost function $J(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k+1) = \mathbf{y}_{i^*}$ where i^* is the index that minimizes $J(\mathbf{y}_i)$,

$$J(y_{i^*}) = \min_{1 \le i \le \nu} \{J(\mathbf{y}_i)\}.$$

The cost function $J(\cdot)$ is of the form

$$J(\mathbf{y}_i) = \sum_{j \in \mathcal{F}} w_j J_j(\mathbf{y}_i) \tag{8.8}$$

where $\mathcal{F}$ is a set of indexes such that the functions J_j, $j \in \mathcal{F}$ are normalized cost functions with $0 \le J_j \le 1$ and are defined to achieve certain objectives. For the purposes of this chapter, $\mathcal{F} = \{t, c, s, r, m, b\}$ but other functions can also be included. The objective of J_c and J_s is to achieve collaboration between the mobile and its neighboring nodes that are very close to it using only local information. On the other hand, the objective of J_r and J_t is to use more "global" information in order to avoid local minima. J_m is a function for achieving collaboration between two or more mobile nodes and finally J_b is a function for avoiding getting out of the area boundaries. Furthermore, w_j, $j \in \mathcal{F}$ are positive weights that tradeoff the various objectives (for example, if it is desired that a mobile moves quickly to its target destination, then w_t is made large).

8.4.1 Path Cost Functions

In this section we present the details for the cost functions that we found to give the best performance among the algorithms that we have investigated. Also other functions that have been investigated are presented for completeness and evaluation purposes.

Neighboring Sensor Collaboration Cost Function using an Artificial Function

A main objective of the collaboration between the mobile and stationary nodes is for the mobile to avoid areas that have been covered by other nodes. The objective of this function to push the mobile away from areas covered by other sensors. The cost function $J_s(\mathbf{y})$ used involves a repulsion force that pushes the mobile away from its closest neighbor. The form of this function is given by

$$J_s(\mathbf{y}) = \max_{j \in \mathcal{H}_{r_c}(m)} \left\{ \exp\left(- \frac{\|\mathbf{y} - \mathbf{x}_j\|^2}{r_d^2} \right) \right\} \tag{8.9}$$

where $\mathcal{H}_{r_c}(m)$ is the set of all nodes in the communication range of the mobile m. The detection range r_d quantifies the region size around the mobile m to be repelled by its neighbors. A related function that we considered consists of the total force applied to the mobile, i.e., the resultant of all repulsion forces from *all* neighbors. However, we found that its performance was inferior to that of (8.9) and thus we do not consider it any further in the chapter.

Target Cost Function

Assuming that the mobile has a target destination point $\mathbf{x}_t$, the cost $J_t(\mathbf{y})$ is a function that pulls the mobile towards its target and is a function of the distance between the mobile and the target position. This function should take a smaller value as the mobile moves towards the target destination thus for the purposes of this chapter it is given by

$$J_t(\mathbf{y}) = \frac{\|\mathbf{y} - \mathbf{x}_t\|}{r_z} \tag{8.10}$$

here r_z is the maximum distance between the mobile node and its target and is used for normalization purposes. There are several ways that one can use to assign a target position to a mobile. For example, target points may be chosen by a central controller as part of the mobile's mission. During a subsequent section we will describe alternative ways of determining the target position for each mobile. Depending on the mode of the mobile's movement, its target may be either an area that is poorly covered (monitored) or the estimated location of a "suspected" source.

All cost functions used in the chapter can be easily computed by a mobile node using information in its cognitive map or by obtaining information from its neighbors. To compute $J_t(\cdot)$, one needs to determine a target position ($\mathbf{x}_t$) and this will be done in the next section.

Neighboring Sensors Collaboration Cost Function using the Cognitive Map

The cost function $J_c(\mathbf{y})$, similarly to J_s, is designed to push the mobile away from areas that have been covered by other sensors (stationary or mobile) using the relevant information from the cognitive map of the mobile node. This function takes a larger value if the candidate position is adequately covered by other sensors and a small value otherwise. Thus, for this chapter, the following cost is used

$$J_c(\mathbf{y}) = \frac{1}{\pi r_d^2} \sum_{\{i,j\} \in \mathcal{D}_{\bar{r}_d}(\bar{\mathbf{y}})} G(i,j) \tag{8.11}$$

where $\mathcal{D}_{\bar{r}_d}$ is given by (8.5) and recall that $\bar{r}_d$ is the discretized detection range of the sensor.

Cognitive Map Triangular Region Cost Function

This type of cost function has been proposed in [17] in the context of cooperative control of Unmanned Aerial Vehicles (UAVs) and its main objective is to give an estimate of the future cost (cost-to-go) so that the mobile will avoid local optimal points. This function gives to the path planning algorithm a more global view of the problem but it also requires some global (not just local) information. $J_r(\mathbf{y})$ represents the percentage of the covered cells in the cognitive map G_k that are included in a triangular region associated with the heading direction of the mobile sensor when going from $\mathbf{x}(k)$ to a point $\mathbf{y} = \mathbf{x}(k+1)$ (see Fig 8.4). This triangle has two important parameters, the height μ and the angle δ.

$$J_r(\mathbf{y}) = \frac{1}{\mu^2 \cdot \tan(\delta)} \sum_{\{i,j\} \in \mathcal{T}(\mathbf{y})} G(i,j) \tag{8.12}$$

where $\mathcal{T}(\mathbf{y})$ is the set of all cells (i,j) included in the triangular region associated with the heading direction and the parameters μ and δ (see also [132] where these parameters have been investigated in the context of wireless sensor networks). Even though this function has worked very well in the context of searching with UAVs, it does have some limitations in the context of sensor network coverage as will be demonstrated in the simulation results section.

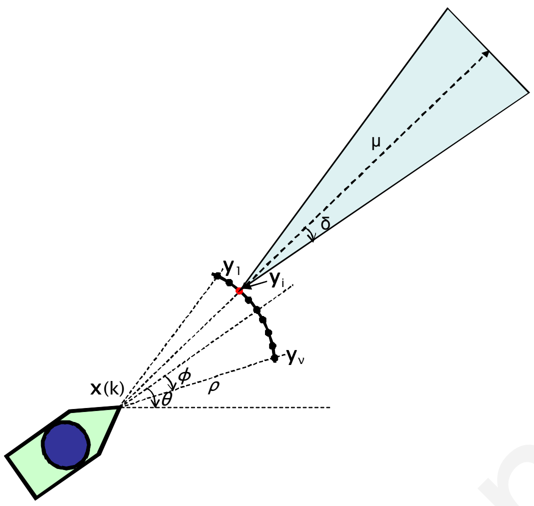


Figure 8.4: Triangular region used for the $J_r(\mathbf{y})$ cost function.

Mobile Sensors Collaboration Cost Function

In addition we investigated a function proposed in [17] to facilitate the collaboration between mobiles, $J_m(\mathbf{y})$ which penalizes each candidate position $\mathbf{y}$ that is close to other mobiles that are heading towards (or returning from) the same direction as the mobile tries to determine its next position. Specifically, when determining its next position, the mobile defines the set Λ that includes all other mobiles that are in its communication range and satisfy the following two conditions. 1) The mobiles that do not follow behind and 2) the mobiles that have a heading direction ξ such that $|\theta - \xi| \leq \varphi$ (the two mobiles are heading towards the same direction) or $|\theta - \xi| \geq 180^o - \varphi$ (the two mobiles are heading towards opposite directions), where φ is the maximum allowed difference in heading angle (see Fig. 8.5). The collaboration function is given by

$$J_m(\mathbf{y}) = \frac{1}{\beta\|\Lambda\|} \sum_{\lambda \in \Lambda} \beta \exp^{-\frac{r_\lambda}{2}} \tag{8.13}$$

where β is a positive design constant and r_λ is the distance between the candidate position $\mathbf{y}$ and the mobile λ. For more information refer to [17].

Boundaries Cost Function

To prevent mobiles from stepping outside the field, a boundary cost function $J_b(\mathbf{y})$ is introduced that penalizes all candidate positions $\mathbf{y}$ that are not included in the field area $\mathcal{A}$. For completeness, the function used is

$$J_b(\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \notin \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{8.14}$$

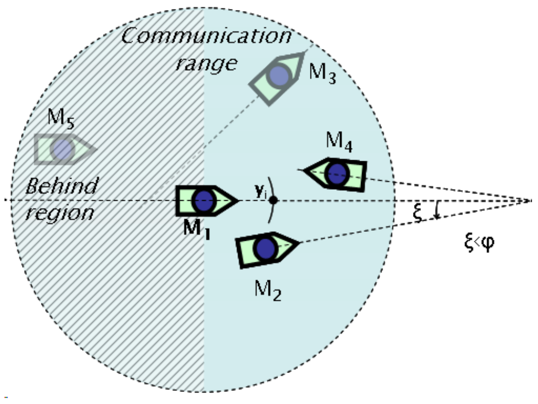


Figure 8.5: Illustration of the set Λ for M_1. Only nodes M_2 and M_4 are used in the calculation of $J_m(y_i)$.

8.5 Dynamic Target Estimation and Allocation

In addition to the possibility of prespecifying a target position for the mobile, in this chapter we investigate the possibility allowing the mobile to dynamically determine its target position $\mathbf{x}_t$; at every step k the mobile uses the collected information to determine its new target location. We point out that it is even possible for a mobile to have two target positions, a short term as well as a longer term target (i.e., include two similar terms in (8.8) with different weights).

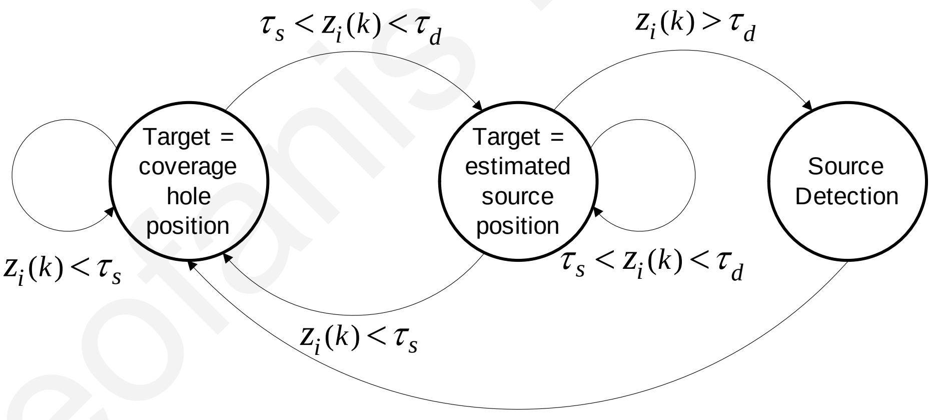

$\tau_s < z_i(k) < \tau_d$
$z_i(k) > \tau_d$
Target = coverage hole position
Target = estimated source position
Source Detection
$z_i(k) < \tau_s$
$\tau_s < z_i(k) < \tau_d$
$z_i(k) < \tau_s$

Figure 8.6: The target allocation strategy for the i-th mobile sensor node during the k-th step.

The dynamic target estimation is performed using two different algorithms depending on the state of the measurements obtained by the mobile and its neighbors as shown in Fig. 8.6. If the mobile does not get any "suspicion" messages from its neighbors (i.e., all obtained measurements are below the suspicion threshold τ_s), then the mobile is in a *coverage* mode and its target is the biggest coverage hole in some neighborhood around the mobile (the size and shape of this area can be a parameter of this problem). On the other hand, if the mobile receives at least a "suspicion" message then it goes into the *search* mode and the target becomes the estimated event source position. Finally, if an event source is detected by the mobile, we assume that it is neutralized[3] and that the mobile moves towards its next target. Next we present the specific algorithms used in each case.

8.5.1 Coverage Hole Estimation Scheme

In this subsection we present a computationally efficient algorithm for coverage hole detection. Using the coverage hole detection algorithm a central controller (e.g the sink) can estimate the coordinates of up to the M biggest coverage hole centers (which can become the target coordinates of the M mobiles). In other words, the aim of this algorithm is to determine where the M mobiles should be placed in order to maximize coverage (i.e., maximize (8.7)). We emphasize that this algorithm can run either by any central controller on the entire field to obtain up to M coverage holes, or by each mobile node itself, to estimate the coordinates of the biggest coverage hole center inside a neighborhood r_c at each moving step k. Since this algorithm may run frequently (as new information regarding the state of the field becomes available) it is required that it is computationally efficient.

Using the principle of divide and conquer we propose the Zoom Algorithm which is very efficient in computation complexity, time and memory. The idea is to divide the grid (i.e the matrix G_k) or any sub-grid (i.e a submatrix of P_k^m that corresponds to the cells $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$) in four equal segments, and choose the segment with the maximum number of empty cells i.e the segment with the maximum number of cells with $G(i,j) = 0$. Then, this procedure is repeated until either the segment size is equal to a single cell or until all segments have the same number of empty cells. In the first case, the hole center position will be the center of the cell. In the second case, the hole center position will be the lower right corner of the upper left segment (the center of the segment during the previous iteration). Fig. 6.2 in chapter 6 illustrates the idea of zooming for hole detection when this algorithm is used by each mobile node in a distributed fashion. The pseudocode of the algorithm, when it is used by a central controller is listed in 4.5 of chapter 4.

8.5.2 Source Position Estimation Scheme

As mentioned earlier, as each mobile node m navigates in the field, it continuously samples the environment and also queries its neighboring nodes about their positions and their sensor measurements z_j, $j \in \mathcal{H}_{r_c}(m)$. In the case when one or more sensor readings are between the τ_s and τ_d thresholds, the mobile node uses the measurements to estimate a likely position of the source which will then become its target location. For this estimation, a number of estimation algorithms can be used (e.g., see [133],

[3]This is a modeling assumption that may not be very practical. On one hand we may assume that the actual time between the step that the mobile detected the event and the next one is long enough to allow a response crew to respond. On the other hand, the mobile may be programmed to ignore (subtract) the signal from the known sources so it can continue its mission.

[134], [135]). For the purpose of this chapter non linear least squares estimation has been used. The event source location (target position) $\mathbf{x}_t = (x_t, y_t)$ is the solution to the minimization problem:

$$J = \sum_{i \in \Omega(k)} \left(z_i - \frac{V}{[(x_t - x_i)^2 + (y_t - y_i)^2]^{\frac{\alpha}{2}}} \right)^2 \tag{8.15}$$

where $\Omega(k)$ is a set of measurements that includes the measurements of the mobile's neighbors at the kth step together with any measurements obtained by the mobile up until step k. In this chapter, a uniform diffusion model [135] has been adopted and also the initial source concentration V is assumed to be known. We point out however that extension for the case where V is unknown is straightforward. As long as the mobile continues to get "suspicion" signals, it continues to search for the source by updating the estimated source position. As before, once the source is detected, it is assumed that it is neutralized and the mobile resumes its coverage function.

8.5.3 Distributed Target Allocation and Collaboration

The previous two subsections describe two different methods that can be used by the mobiles in order to autonomously decide their target location. Both methods utilize information that can be obtained by the mobile from its neighborhood. In the case of the coverage hole estimation, the information is included in a relevant submatrix of the cognitive map, while for the source position estimation the relevant information is the measurements of the neighboring nodes. A possible problem arises when two or more mobiles are close to each other. In this case, it is very likely that the information they will use to estimate the target position will be the same and as a result they will all estimate the same target location. Clearly, this is not a good collaboration strategy since there is no benefit if they all converge to the same point.

To avoid this problem (as well as possible collisions), we utilize the following two protocols depending on the state of the mobile node (i.e., searching for a source or coverage).

If a mobile node m is in searching mode and also in communication range with other mobiles, then it queries its neighboring mobiles for their current position and their target locations. Then, it computes the distance between its own target and the target of the neighboring mobiles $d^t_{m,j}$ for all neighboring mobiles j,

$$d^t_{m,j} = \|\mathbf{x}^t_m(k) - \mathbf{x}^t_j(k)\|. \tag{8.16}$$

If this distance is greater than a threshold value then it assumes that the two mobiles are heading towards different targets and thus it continues normally. If $d^t_{m,j}$ is less than the threshold value then it is very likely that the two mobiles are heading towards the same suspected point and thus only one should continue the search towards that target while the other should switch to the coverage mode. This decision is based on the distance of each mobile from its target. The mobile that is closest to its target continues the search while the other switches to the coverage mode. For the purposes of this chapter, the threshold distance used to decide whether two mobiles are heading towards the same target is set to $2r_d$.

Now if a mobile node m is in the coverage mode and is also in neighborhood of other mobiles, then, in order to avoid going towards the same point, it queries the other mobiles in its communication range for their current locations and their target points. Once a mobile has received the target points of all mobile neighbors, then it updates its cognitive map and assumes that these target points constitute covered areas. Then it proceeds normally with the coverage hole estimation algorithm (zoom algorithm). With this simple scheme, the mobiles avoid exploring the same areas. This scheme has some important benefits. It is distributed (no need for a central controller), it is simple, and utilizes only local information (the relevant information in the submatrix $\mathcal{D}_{r_c}(\bar{\mathbf{x}}_m(k))$, which corresponds to the neighborhood r_c of the cognitive map).

Finally, it is worthwhile to mention that when two mobiles come into communication range, they can also exchange their cognitive maps so that a mobile does not explore areas already explored by other mobile nodes.

Discussion

The above collaboration protocol is illustrated in Fig. 8.7, each mobile asynchronously computes its target at each step using local information received for its neighboring nodes. For example in the snapshot presented in Fig. 8.7 the mobile at the right is closer to its target than the other mobile and thus it engages its target. Note that this target may change in the next steps. The mobile at the left is currently engaged a target which is very close to the target of the mobile at the right. As this mobile is far away for its target, compared to the mobile at the right, during its next step it will estimate a different target (the one shown in the figure) which will be not very close to target of the neighboring mobile at the right. Therefore, at each step neighboring mobiles exchange their current positions and targets and utilize the collaboration protocol to collaborate and find spatially separated targets in an asynchronous [4] manner (synchronization is not needed).

8.5.4 Collision Avoidance and other Conflicts

In the proposed path planning algorithm collision avoidance is achieved efficiently by the neighboring sensor cost function J_s. As mention before J_s is a function that repels each mobile from its nearest neighbor and therefore when mobiles come very

[4] In the case of a synchronous system, (i.e. when the mobiles simultaneously find their targets) mobile nodes might keep moving towards the same target, given that their dynamic targets are not changing over time. To alleviate this extremely rare scenario each mobile should wait for a small random delay (exponential backoff period) before estimating its target.

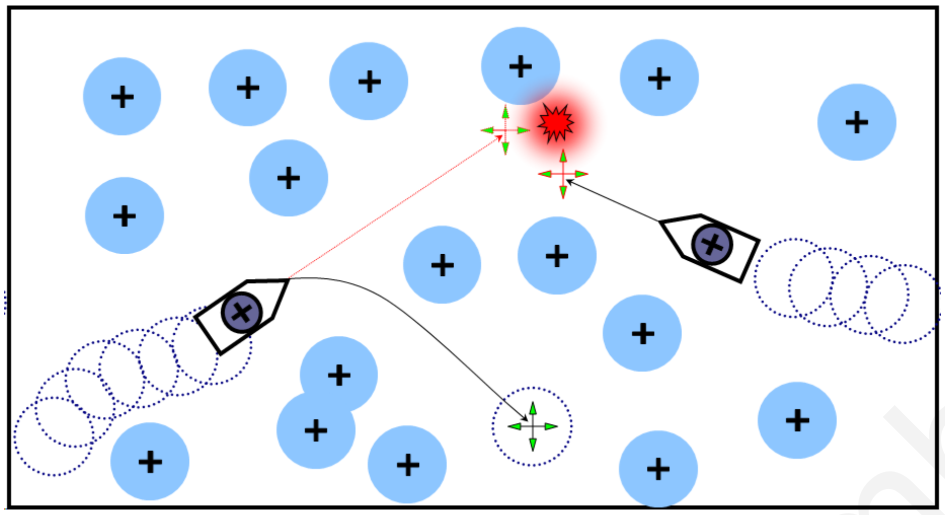


Figure 8.7: Distributed asynchronous collaboration via target allocation

close ($< r_d$)[5], they will repelled from each other and thus selected their next steps in a way that maximizes the distance between them. This function proactively prevents collisions as the distance ρ traveled by each mobile during the next step is very small compared to the distance r_d ($\rho < r_d$) where the mobiles start repel each other effectively. However, due to the collaboration protocols illustrated previously the cases where mobiles come very close to each other are extremely rare. Additionally one could employ the mobile sensors collaboration function J_m to prevented the congestion of mobiles in a small region. It is worth mentioning that the proposed collision avoidance scheme is based on algorithmic approaches, however to guarantee that the robots will never collide, some kind of hardware interrupt based on collision avoidance sensor measurements, should be implemented when mobile robots come seriously close to each other. Finally, possible deadlocks, livelocks or cases when the robots stuck in local minima or oscillate between the two closest points are almost impossible to occur due to the dynamic behavior of the path planning algorithm, the spatially large decision points available as well as the asynchronous nature of the collaboration protocol. Specifically, the mobiles at each step are moving towards a spatially different target (dynamic target) [6], the neighboring sensor cost function J_s is based only on the nearest neighbor avoidance and the selection of the next position of the mobile (see Fig. 8.3) does not consider the current mobile position and thus the mobile should move to a new position which is at distance ρ away from its current position and within an angle 2ϕ (therefore each current position is a “tabu” point). Therefore given the above facts local minimums as well as other conflicts are almost impossible to occur.

[5]When r_d is comparable to the physical radius of the mobile robot then this physical radius could be feed in J_s

[6]Even when two mobiles synchronously engage a target at a position which is equidistant from the mobiles, during the next step the distance of this target will not be the same given that the target is not changing in the next step!

8.6 Simulation Results

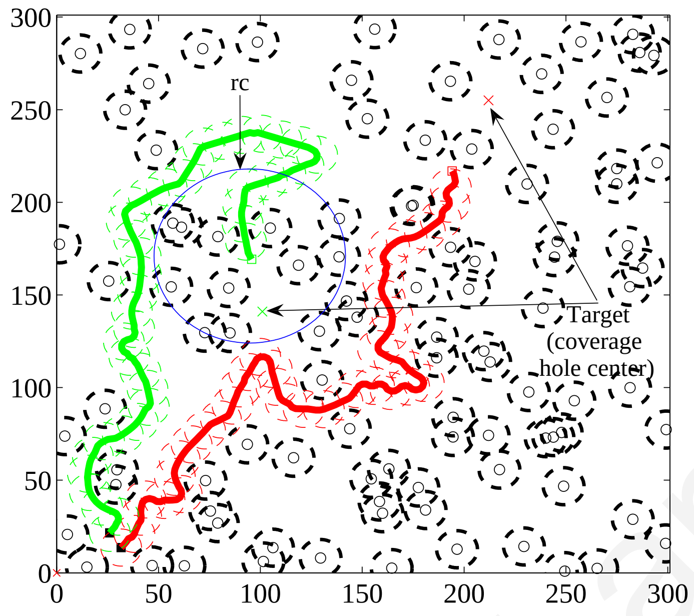
In this section we present some simulation results with some representative scenarios that show the movement of a set of mobile nodes and also compare the performance of different path planning algorithms (all from the family of algorithms presented in Section 8.4). Depending on which cost functions are used in (8.8) and the weights, one can obtain different algorithms. To distinguish between the different algorithms investigated, we use acronyms where each letter corresponds to the individual cost functions used, for example TS refers to an algorithm for which $w_t > 0$ and $w_s > 0$ while $w_c = w_r = w_m = 0$[7].

Unless otherwise stated, all experiments refer to a square $300m \times 300m$ field and a grid with $da = 1m$ is used. The mobile maneuverability parameters are set to $\rho = 2m$ and $\phi = 30°$ while for every decision $\nu = 10$ candidate next positions are considered. For the event propagation model, we assume that $V = 1500$, $V_{sat} = 100$, and the exponent $\alpha = 2$. Finally we assume that a detection threshold $\tau_d = 15$, thus the sensing radius of all sensors (stationary and mobile) is $r_d = 10m$ and the communication radius $r_c = 4.5 \cdot r_d = 45m$ (for the neighborhood of each sensor we only consider its one hop neighbors).

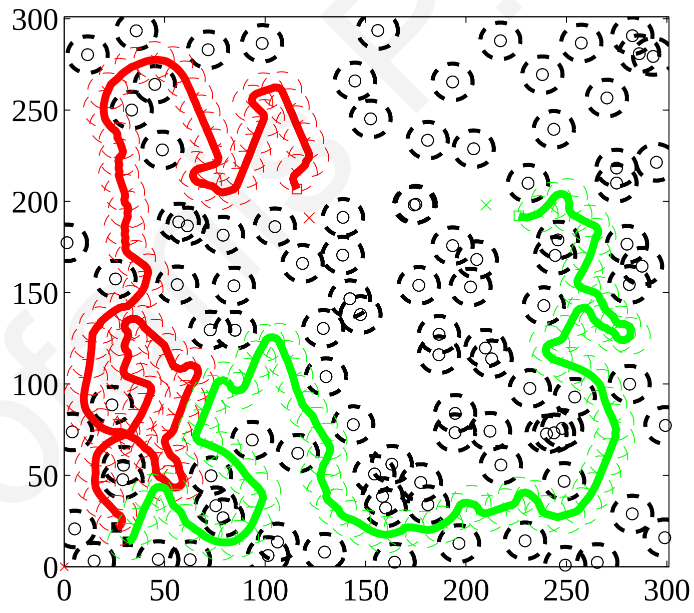
Next we present some representative scenarios and show the movement of a team of robots that uses the *Distributed TS* algorithm, a simple algorithm that performed very well against all other algorithms investigated. In this algorithm, every mobile makes autonomous decisions using only the J_t (with $r_z = r_c - r_d$) and J_s cost functions (i.e., $w_t = 0.8$, $w_s = 0.2$ and $w_c = w_r = w_m = 0$). For estimating the target positions, the mobile uses either the coverage hole detection algorithm (in coverage mode) or the source position estimation algorithm (in search mode) and the distributed target estimation scheme presented in the previous section. Finally, for the coverage hole detection algorithm only the cells in $D_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ are used. In other words, the coverage hole is estimated only in its neighborhood.

In the first simulation experiment we use a team of two mobile nodes and show the behavior of the Distributed TS algorithm in a field with 100 randomly deployed stationary sensors. In this simulation scenario there is no event source thus Fig. 8.8 shows how the two mobile nodes navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors. As seen from the paths followed, there is collaboration between mobile and stationary sensors in the sense that the mobiles have found two different paths that are least covered by the stationary sensors. Also notice how the two mobiles collaborate and select different targets at the beginning of their motion. Moreover note that one can adjust the mobile's parameters in order achieve different objectives. For example, Fig. 8.8(a) shows a path where the mobiles move quickly through the field to achieve faster detection. On the other hand, Fig. 8.8(b) shows a scenario where the mobiles try to achieve better coverage by covering a hole before they proceed. Finally, we point out that given enough time, all

[7]For all algorithms and all experiments to prevent any mobile from going outside the area we have used $w_b = 1$.



(a) Paths followed when the mobile's objective is fast detection ($w_t = 0.8,\ w_s = 0.2,\ r_c = 45m$).



(b) Paths followed when the mobile's objective is better coverage ($w_t = 0.2,\ w_s = 0.8,\ r_c = 25m$).

Figure 8.8: Dynamic path planning using a team of two mobile nodes.

algorithms will cover the entire field.

Fig. 8.9 shows the paths followed by two mobile nodes when a set of five non-overlapping static sources exist (each source is turned on at the beginning of the simulation time and stays on for the entire simulation with $V = 3000$). We assume 100

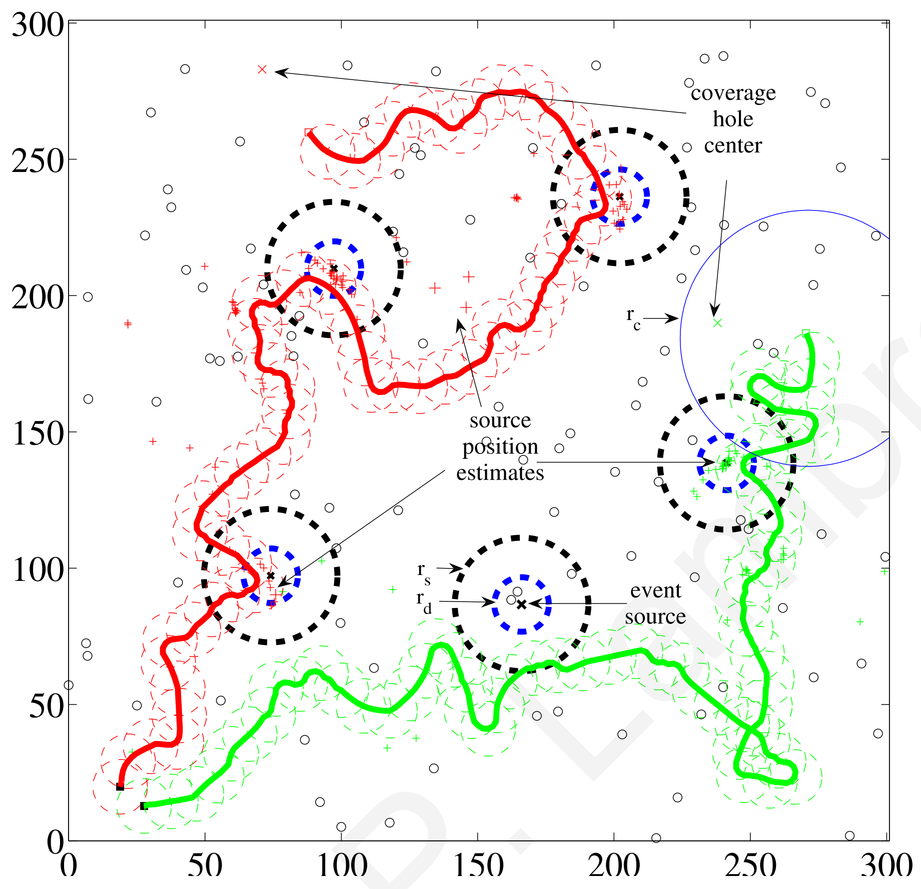
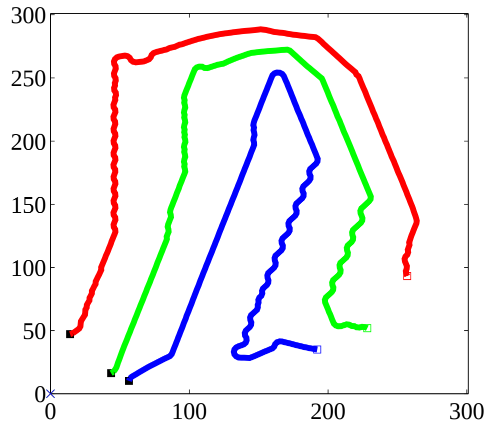


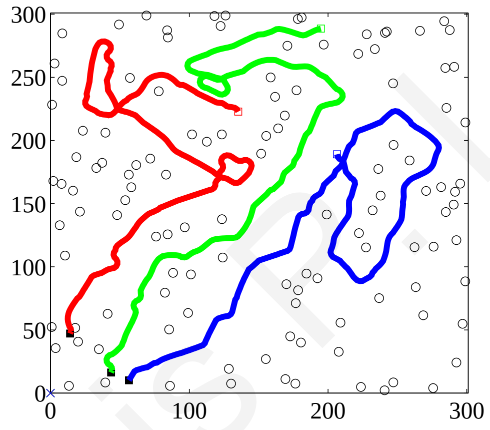
Figure 8.9: Dynamic distributed path planning using a team of two mobile nodes in the presence of event sources.

randomly deployed sensors in the field. The detection threshold of all sensors is $\tau_d = 30$ (thus $r_d = 10$m), and the suspicion threshold is $\tau_s = 5$ ($r_s = 24.5$m). Fig. 8.9 also shows the positions of the event sources. One source is reliably detected by the stationary sensors however for the remaining four there are no stationary sensors in a radius r_d around the event, thus these events would have remained undetected. Initially, both mobile nodes are navigating towards their current estimated coverage hole positions. Note that in some cases there are sensors within r_s from the event sources and these sensors are likely to report the "suspicion" to the passing mobile node. Once a mobile node gets a suspicion message from a static node in its communication range (or its sensor measurement is inside the "suspicion" region, $\tau_s \leq z_i \leq \tau_d$), then it switches its target to the estimated location of the event.

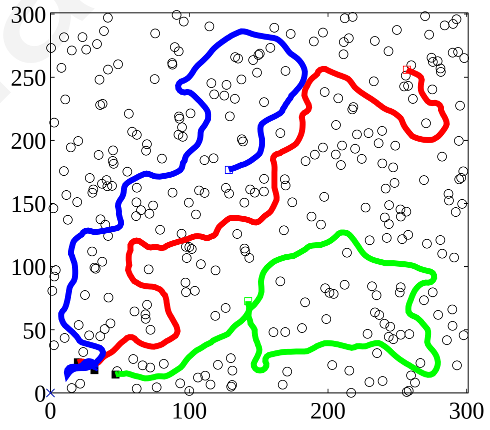
The next simulation experiment demonstrates the behavior of the Distributed TS algorithm (with fixed parameters as described above) for sensors fields with different densities (empty, sparse and dense fields). Fig. 8.10 shows the paths followed by three mobile nodes after 300 moving steps. From the figure it is evident that the Distributed TS algorithm is able to easily adapt to different sensor node densities without getting trapped in local minima. Mobile nodes always keep navigating in the sensor field, passing/sampling through uncovered regions and improving coverage. Fig 8.10(a) shows that in the case of an empty field (no stationary sensors are available) mobile nodes collaborate and navigate similarly to standard search algorithms.



(a) Paths followed in an empty sensor field (0 stationary sensors)



(b) Paths followed in a sparse sensor field (100 stationary sensors)



(c) Paths followed in a dense sensor field (300 stationary sensors)

Figure 8.10: Paths followed after 300 moving steps by a set of three mobile sensor nodes using the distributed path planning algorithm for different sensor field densities

In the next simulation experiment (Fig. 8.11) we investigate the value for the suspicion threshold τ_s. Note that there exists a tradeoff in its actual value. If this threshold is set too high, then the mobile will get in the searching mode rarely (clearly $\tau_s < \tau_d$). On the other hand, if this threshold is set too low, then the mobile will be running after frequent false alarms. In this experiment we evaluate the number of detected sources over 20 sensor fields with 100 stationary sensors. In each field 15 non-overlapping event sources are randomly placed. As shown in Fig. 8.11(a) only a small number of the sources is detected by the stationary sensors (at time zero, about 6.5 sources on average are detected). A group of two mobile sensor nodes using the Distributed TS algorithm is employed. We measure the average number of detected event sources as well as the average coverage improvement for 1000 moving steps. Moreover the following values for other parameters are used: noise variance is $\sigma^2 = 10$, $\tau_d = 15$, $\nu = 5$, $r_c = 5 \cdot r_d$, and $r_z = r_c - r_d$.

Fig. 8.11 shows that if the suspicion threshold is set too low ($\tau_s = 1$), then the mobile does run after frequent false alarms and as a result its performance with respect to either the number of detected sources or the area coverage is not very good. As shown in the Fig. 8.11 the best value for this experiment is $\tau_s = 5$ as this value succeeds coverage close to the maximum which means that it minimizes the uncertainty (or the probability of miss events) and at the same time achieves the maximum rate of detected event sources.

In the next simulation results we compare the following path planning algorithms:

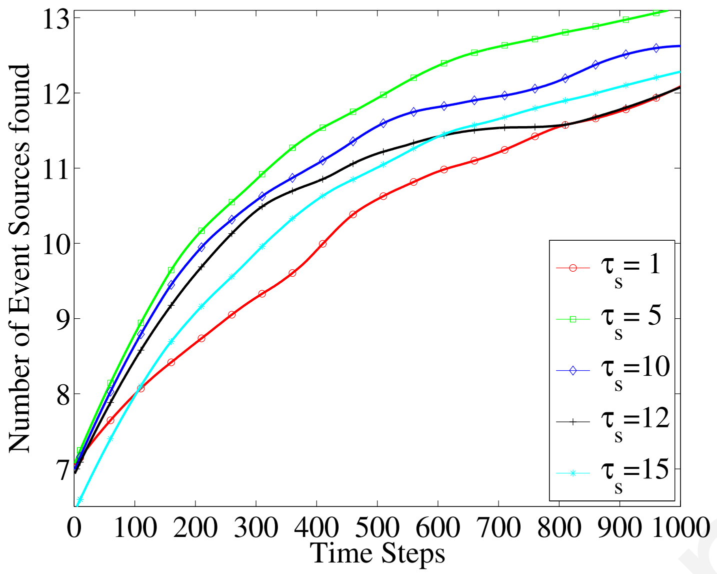
1. RCM: This algorithm has been developed in [17], [75] for cooperative search missions by UAVs. The RCM algorithm uses the cost functions J_r, J_c, and J_m with the following weights $w_r = 0.5$, $w_c = 0.2$, $w_m = 0.3$ and with triangle parameters $\delta = 15°$ and $\mu = 40$. Note that since this algorithm does not use the J_t function, it can only navigate in the field to reduce uncertainty (maximize coverage) and cannot move towards a target.

2. TCM: In this algorithm a central controller decides the next step of each mobile node. Once a mobile node approaches its target destination a new target (coverage hole point) is assign to the mobile using a centralized target assignment scheme where the controller computes the biggest coverage hole in the entire field which is not already assigned to other mobile nodes. The TCM algorithm uses the following cost functions J_t, J_c, and J_m with the following weights $w_t = 0.5$, $w_c = 0.2$, $w_m = 0.3$ and with parameters $r_z = \sqrt{2\mathcal{A}}$, where $\mathcal{A}$ is the sensor field area.

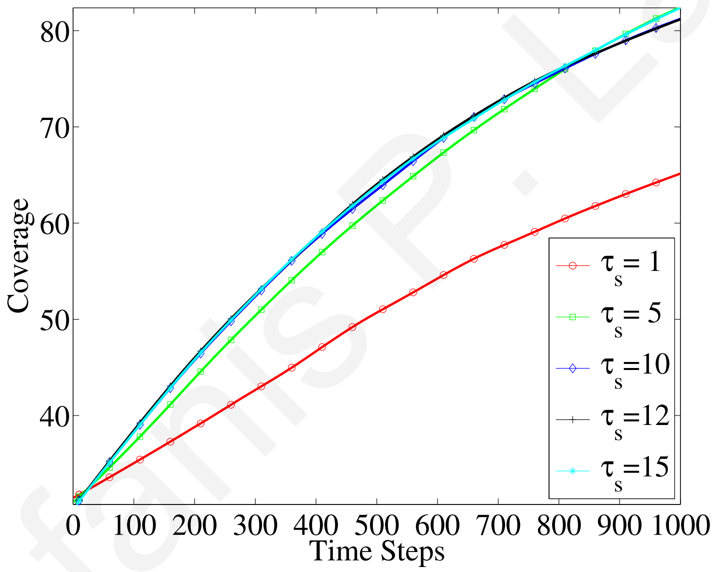
3. TSM: This algorithm is similar to the TCM algorithm (uses a central controller to solve the global target assignment problem). The TSM algorithm uses the following cost functions J_t, J_s, and, J_m with the following weights $w_t = 0.5$, $w_s = 0.2$, and $w_m = 0.3$ and with parameters $r_z = \sqrt{2\mathcal{A}}$, where $\mathcal{A}$ is the sensor field area.

4. Distributed TS: As described earlier.

Furthermore the following parameters are used: $r_d = 10$ ($\tau_d = 15$), $\tau_s = 5$, $r_c = 3 \cdot r_d$, $\nu = 5$ and $\sigma^2 = 10$.



(a) Average number of non overlapping event-sources found over 20 sensor fields

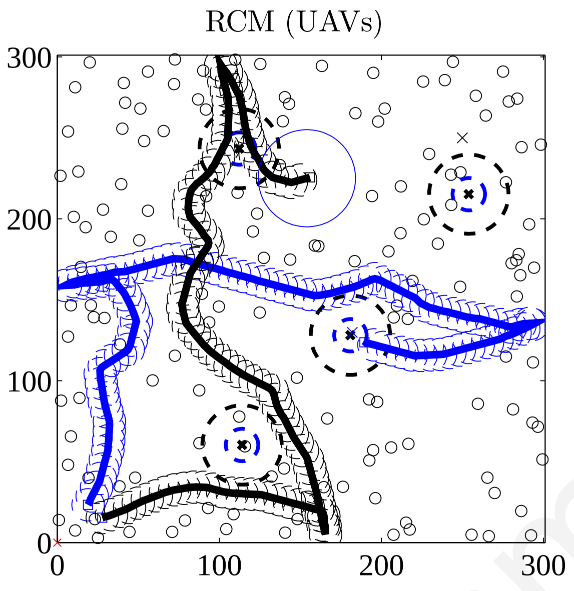


(b) Average coverage improvement over 20 sensor fields

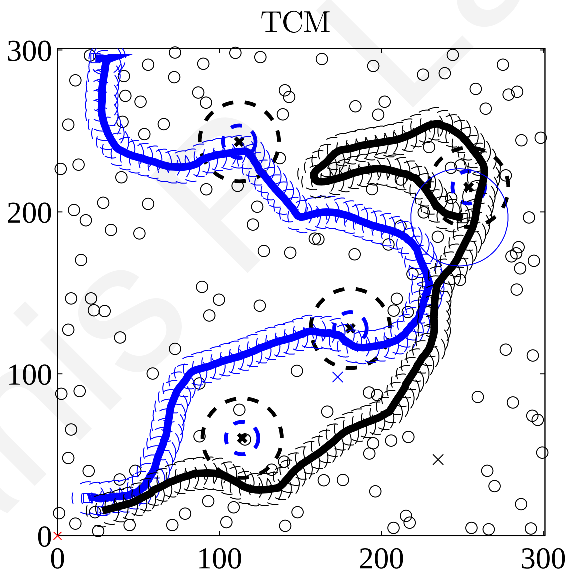
Figure 8.11: Evaluation of the suspicion threshold τ_s optimum value.

Fig. 8.12 shows the paths followed by two mobile nodes for 500 moving steps when the above algorithms are employed. We use a randomly deployed sensor field with 100 stationary sensor nodes and 4 non overlapping event sources. As shown in Fig. 8.12(d) the Distributed TS algorithm achieves better collaboration between the mobiles and detects all the event sources. Better collaboration is achieved because the paths, followed by the mobile sensors using the distributed TS algorithm, have the minimum overlap (almost zero) compared to the other algorithms.

Next we compare the average performance of each algorithm using Monte Carlo simulation. We assume 20 sensor fields with 100 randomly deployed static sensors and 15 static non-overlapping sources (also placed at random points). Fig. 8.13 is an average over the 20 randomly generated sensor fields and shows that the static sensor



(a) RCM

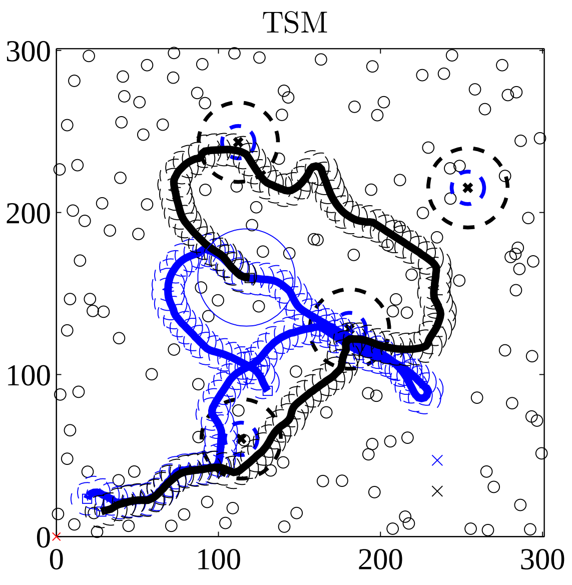


(b) TCM

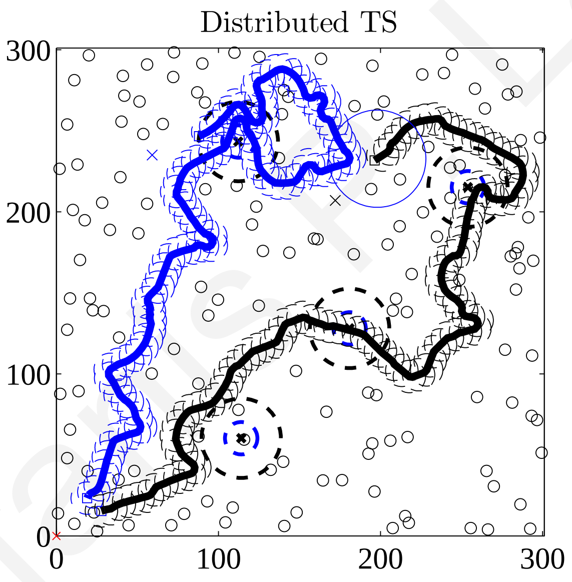
Figure 8.12: Paths followed for 500 moving steps using different path planning algorithms

network would have detected around 6 − 7 event-sources on average and the average coverage of the stationary field would be about 30%. Next, a set of two mobile nodes is used for 1000 moving steps. Fig. 8.13 shows that the Distributed TS algorithm outperforms the other algorithms both in the average number of detected event-sources (see (Fig. 8.13(a)), and in the average coverage improvement (Fig. 8.13(b)) and its computation is negligible compared to the RCM algorithm (Fig. 8.13(c)) mainly because there is no need to compute the triangle needed in J_r.

This performances indicates that the Distributed TS algorithm is able to achieve better collaboration between the mobile nodes and its computation efficiency shows



(c) TSM

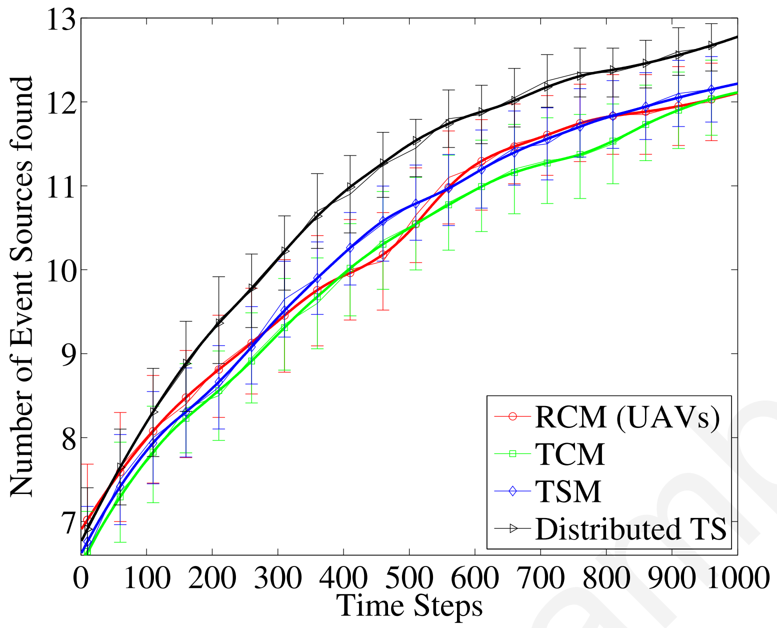


(d) Distributed TS with distributed target assignment

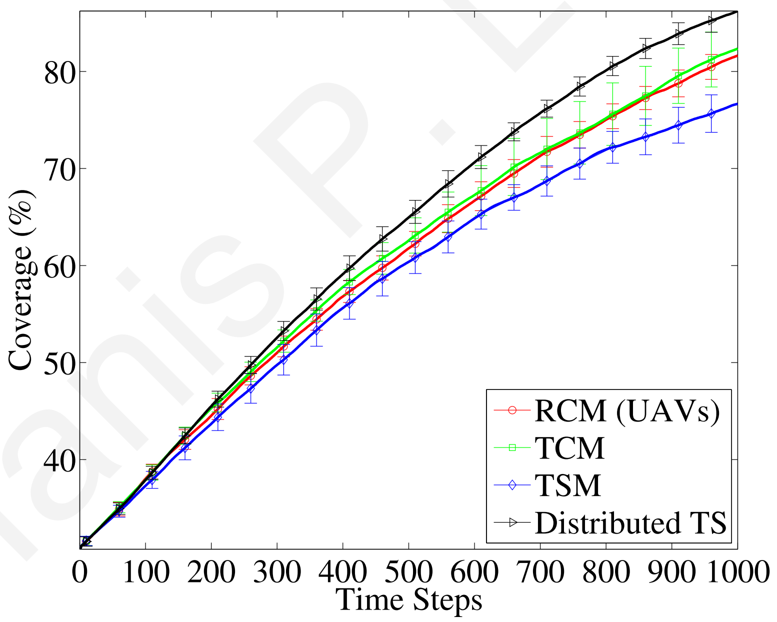
Figure 8.12: Paths followed for 500 moving steps using different path planning algorithms(cont.)

that it is a good candidate to be implemented even onto a tiny microcontroller of a mobile sensor node [3].

The centralized coverage hole assignment scheme eliminates the detection and the coverage performance of the TCM and TSM algorithms as the mobile nodes are not able to adapt their paths on their way towards their targets and they can easily get trapped by (or move on) other mobiles paths although the are navigating towards the best coverage hole of the sensor field. This behavior is clearly illustrated in figure 8.14 where we compare the centralized (TSM algorithm) and the distributed (TS algorithm)



(a) Average number of non overlapping event-sources found over 20 sensor fields

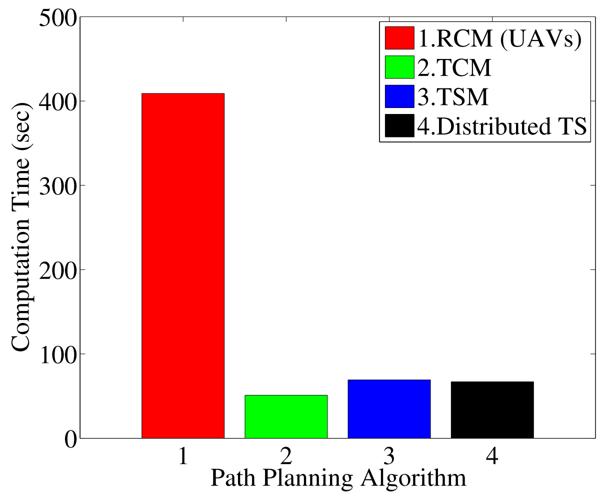


(b) Average coverage improvement over 20 sensor fields

Figure 8.13: Comparison of different path planning algorithms

coverage hole estimation scheme.

Fig 8.14(a) shows the scenario where the mobile with the red path has been trapped by its neighbor path (black path) in the case when the global target assignment scheme is used. Using the same scenario (sensor field), Fig 8.14(b) shows the path that the 3 mobiles follow when the distributed-dynamic target assignment scheme is used. Using this decision process each mobile select its optimum target inside its communication range which is not assigned to any other mobile as they share their current targets positions when they come into communication range. Moreover these target-coverage



(c) Average computation times

Figure 8.13: Comparison of different path planning algorithms(cont.)

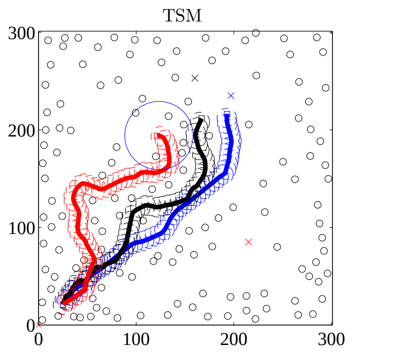
hole positions are never reached as in the next step a new target will be selected in order to adapt in the dynamic environment. As shown in Fig 8.14(c) the local target assignment scheme improves more the area coverage compared to the global target assignment scheme.

Finally, as mentioned earlier, fast event detection and area coverage may be two slightly conflicting objectives. Depending on the objective, there may be one or more optimal paths, however, finding them is not easy. Given a path, an easier problem is to determine whether it achieves close to optimal performance. For the coverage objective, this is easily done by observing the coverage overlap between the static and mobile sensors. In that respect, the paths found by the Distributed TS algorithm have performance close to the optimal.

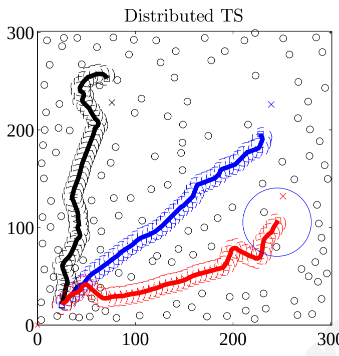
8.7 Conclusion

In this chapter we propose a collaborate event detection architecture for WSNs consisting of a large number of stationary nodes and a few mobile nodes. The benefit of this architecture is that the mobile nodes collaborate with the stationary nodes so that they sample the areas least covered by the stationary nodes. In this way, events that would have remained undetected can now be detected.

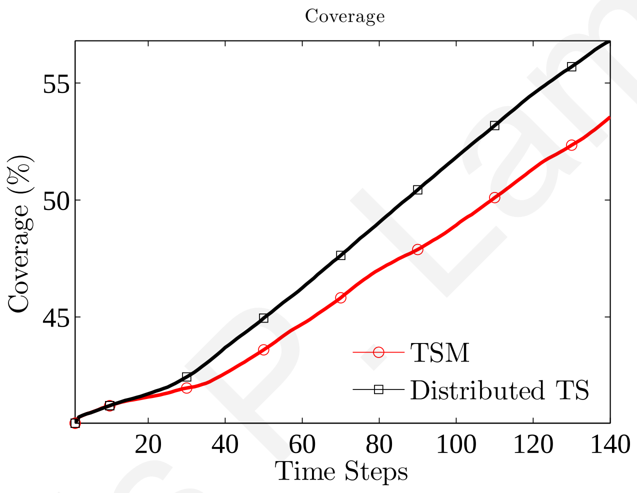
For the proposed architecture we have investigated a family of path planning algorithms that are based on the receding horizon approach. At every step, the mobile controller estimates the cost of moving to a finite set of future positions and moves to the one that achieves the minimum. This cost is a linear combination of certain functions each designed to achieve certain objectives. Five such functions have been investigated in this chapter, but more can also be included (e.g., functions that represent obstacles). Among the functions investigated, two had a more local perspective



(a) Centralized coverage hole assignment



(b) Distributed coverage hole assignment



(c) Coverage

Figure 8.14: Comparison between centralized and distributed coverage hole assignment

and were designed to avoid stepping to areas covered by immediate neighbors (J_c and J_s). The other two were used to give a more global pictures (J_r and J_t) and one was explicitly used to facilitate the collaboration between the mobiles (J_m). Our investigation yielded the following conclusions with respect to these functions when applied in the context of randomly deployed sensor networks.

- J_c significantly restricts the movement of mobiles and some times it creates fictitious barriers that may "trap" other mobiles and as a result the simpler J_s is able to achieve a better collaboration between the mobile and its neighbors and yield better performance.
- Even though in the context of UAVs J_r could achieve very good performance, in the context of randomly deployed sensor networks, its performance was limited and was often outperformed by J_t. One limitation of the J_r function (other than the complexity) is that for reasonably large triangles, due to the random field deployment, the number of sensors that fell in the triangle was fairly constant (proportional to the size and field density) providing no significantly new information.
- When the coverage hole detection algorithm is used to determine the targets of the mobiles, it is usually more beneficial (achieves better collaboration between

a group of mobiles) if targets are determined more frequently and closer to the mobile as opposed to more "globally". If a far away target is given to the mobile and is not updated frequently, then it cannot utilize newly discovered information that can help it achieve a better performance.

- When a target is used, the distributed target assignment scheme is more effective in facilitating the collaboration between the mobiles compared to the J_m function.

In the next chapter, the optimal path that minimizes the expected event detection time in the context of mixed sensor networks is investigated. The optimal path problem is found to be NP-complete but provides evidence that searching for an event "locally" leads in fast event detection in our context. The proposed path planning strategy is optimized further based on that observation.

Chapter 9

Area Coverage and Event Detection in Monitoring Applications using Mixed Sensor Networks

9.1 Summary

This chapter considers the problem of improving the monitoring capability of a sparse stationary sensor network by using mobile sensor nodes and addresses the trade off between area coverage and fast event detection. In this context, this chapter provides a theoretical study for deciding the optimum searching path for one mobile node that minimizes the expected event detection time when several large uncovered regions exist in the sensor field. Finding the optimal searching path is an *NP*-hard problem, which implies that we can not find the optimal solution of an arbitrary problem instance in reasonable time, due to large number of uncovered regions and multiple mobile nodes. Thus using the principles of the derived solution, the proposed path planning strategy is improved further. Both analytical and simulation results indicate that a "local" search is better than a "global" search in the context of mixed sensor networks.

9.2 Introduction

In the previous chapters we present the development of a distributed collaborative path planning strategy for area coverage and fast event detection. Simulation outcomes in chapters 7 and 8 indicate that is better to search - cover local uncovered regions first instead of searching the most bigger uncovered regions in the sensor field first (i.e. is better to determine dynamic targets closer to the mobiles). In this chapter, we investigate optimum path planning strategy for a single mobile node when several uncovered regions exist in order to gain some insight that can be used to further optimize the performance of the proposed distributed path planning strategy.

Theoretical work on searching for targets in unknown location was initiated by B. Koopman [73] during the World War II to find enemy marine vessels for the U. S. Navy. Search theory as we know it today is based on work by Koopman [73] and later work by L. Stone [74] who especially study the moving target problem. However, in [73, 74], there is significant focus on how to allocate search effort across the environment instead of finding the best search path to follow.

Depending on the type of wireless sensor network used, there is a fundamental trade off that involves the area coverage and the delay to detect an event. In monitoring applications with stationary sensor nodes, an event is either detected or not (it is not detected if it occurs in a coverage hole) and this is done immediately (assuming a connected network where the message propagation time is negligible). On the other hand, in applications with mobile nodes, it is possible to search and cover the entire field, however, this will take time, thus an event may remain undetected for a significant amount of time. An important question that arises is how to search the area such that events are detected in minimum time. Intuitively, one may argue that in order to minimize the detection time, one needs to first search the areas where it is most likely to find the event. However, because of the time that may be needed by the mobile to get to the most likely area, it may be faster to first search less likely areas that are closer to the mobile node. To address this trade off, in this chapter we adopt the mixed sensor network framework proposed in the previous chapters 6,7 and 8 and consider a scenario with several coverage holes.

The contribution of this chapter is that it determines the optimal strategy of the mobile node such that the average detection time is minimized. The resulting strategy is an optimal trade off between the probabilities of finding an event at a location and the distance of the mobile from the location. The resulting optimal strategy confirms that it is better to search areas that are less likely to hide an event but are located closer to the mobile node rather than heading towards the *most* likely area. This strategy has a counter-intuitive implication that a “local” search is better than a “global” search. This implication is also confirmed in the context of monitoring applications using mixed sensor networks [136, 137]. We have also proved that the optimal search path strategy is an *NP*-complete problem and therefore we proposed a heuristic centralized path planning strategy that provides computationally tractable and close to optimal solution to the problem. This heuristic centralized approach has been also compared with the proposed distributed collaborative path planning approach which remains valid for the general case where uncovered regions created by random deployment of static sensor nodes. Results indicate that the selection of the searching neighborhood radius r_z is of critical importance.

The remaining of the chapter is organized as follows. Section 9.3 describes the model that has been adopted and the underlying assumptions. Section 9.4 derives the optimal search strategy of the mobile node with respect to the detection time. Section 9.5 presents a path planning algorithm that can be utilized by a mobile node in order to navigate through the sensor field. Section 9.6 presents the simulation results. The chapter concludes with Section 9.7.

9.3 Model Description

In this chapter, we address the problem of detecting a stationary event (target) using a mixed sensor network consisting of both mobile and static nodes. In this section we present the modeling assumptions and define some concepts and objectives that will be used in the sequel. We consider a mixed sensor network made of a large number of sensor nodes deployed in a large region $\mathcal{A}$ as shown in Fig. 9.1.

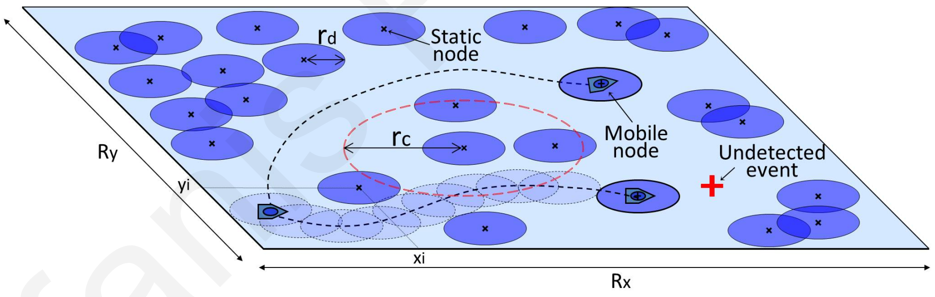


Figure 9.1: Mixed sensor network model.

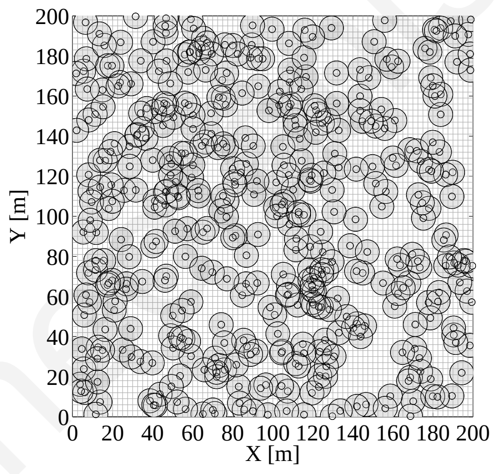
When a large region is to be monitored by a sensor network, it is desirable to deploy randomly a large number of inexpensive, low cost sensor nodes to the given area (as manual deployment might be infeasible, impractical and costly). However, the problem is that a dense deployment is not always feasible due to high cost (even though each sensor could be cheap, deploying a large number of them is still expensive) thus a WSN with static sensors may not be able to guarantee full coverage. It is also noted that as time passes, more coverage holes may be created due to the random failure of some sensors (due to ageing).

We assume that the region under monitored is a rectangular area $\mathcal{A} = \mathcal{R}_x \times \mathcal{R}_y$ and a set $\mathcal{S}$ with $S = |\mathcal{S}|$ static sensor nodes that are randomly placed in the area $\mathcal{A}$, at positions $\mathbf{x}_i = (x_i, y_i)$, $i = 1, \cdots, S$. In addition, we assume that a set $\mathcal{M}$ of $M = |\mathcal{M}|$ mobile sensor nodes are available and their position after the k-th time step is $\mathbf{x}_i(k) = (x_i(k), y_i(k))$, $i = 1, \cdots, M$, $k = 0, 1, \cdots$. We also consider a stationary event that occurs at random position $\mathbf{e} = (x^e, y^e)$ in the environment.

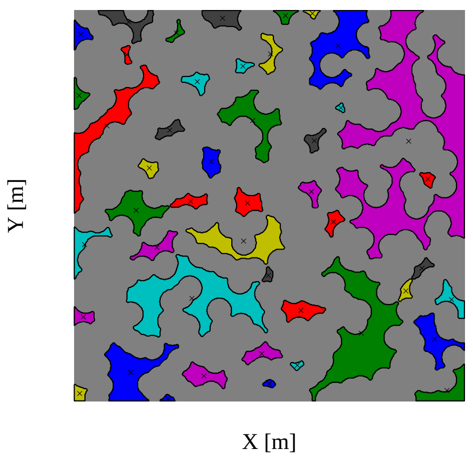
We assume that all static and mobile nodes have a common (known) sensing range r_d with the sensing area of πr_d^2 and and a common communication range $r_c > r_d$ (see Fig. 9.1). For notational convenience, we define the set of *all* sensor nodes $\mathcal{N} = \mathcal{S} \cup \mathcal{M}$ and in this set the mobile nodes are re-indexed as $m = S + 1, \cdots, N$, where $N = S + M$. It is assumed that all sensors know their location through a combination of GPS and localization algorithms. Furthermore, it is assumed that all sensors can reach the gateway (commonly referred to as sink in the WSN literature) using multihop communication. The neighborhood of a sensor s is the set of all sensors nodes that are one hop away, i.e., the nodes that are located at a distance less than or equal to r_c from s. This set is denoted by

$$\mathcal{H}_{r_c}(s) = \{j \ : \ \|\mathbf{x}_s - \mathbf{x}_j\| \leq r_c, \ j \in \mathcal{N}, j \neq s\} \tag{9.1}$$

The sensor field can be partitioned into two regions in terms of coverage: covered and uncovered. The covered region (see Fig. 9.2(a)) means that any point in the region is within the sensing area (coverage) of at least one sensor. The uncovered region (see Fig. 9.2(b)) is the complement of the covered region. Every sensor has a sensing area and the union of the sensing areas of all sensors is called the sensing area coverage. If an event occurs within the sensing range of a sensor node, it can be detected immediately. If an event occurs in an uncovered region then it is possible to be detected by mobile sensors with some delay. The aim of this chapter is to plan the path of the mobile nodes in order to search the uncovered regions such that events are detected in minimum time.



(a) The grid map of a sensor field



(b) The coverage holes in the sensor field detected using image processing techniques

Figure 9.2: Covered and uncovered regions in a WSN of 400 randomly distributed stationary sensors with sensing radius $r_d = 5m$.

To make the concept of area coverage more concrete, the entire sensor field is discretized into an $X \times Y$ grid as shown in Fig. 9.2(a). The current state of the sensor field is represented by a $X \times Y$ matrix G_k, $k = 0, 1, \cdots$, which corresponds to the "confidence" in detecting an event. If the (i, j)-th cell falls in the detection range of a functioning static sensor, then the corresponding $G_k(i, j) = 1$, for all k and we are confident that no event will occur in the area of the corresponding grid cell without being detected. If the matrix element has the value 0, then we have no way of knowing if an event has occurred in the corresponding area. This matrix represents the accurate state of the sensor field and is updated as the mobiles move around in the field. Thus at every step, we use the following updating rule for every element of matrix G_k.

$$G_{k+1}(i, j) = \begin{cases} 1, & \text{if}(i, j) \in \mathcal{D}_{\bar{r}_d}(\bar{\mathbf{x}}_s) \\ f.G_k(i, j), & \text{otherwise} \end{cases} \tag{9.2}$$

where $\bar{\mathbf{x}}_s$ are the coordinates of sensor s in the Grid G_k and $\mathcal{D}_{\bar{r}_d}(\bar{\mathbf{x}}_s)$ is the set of Grid cells covered by sensor $s \in \mathcal{N}$ with detection range r_d. Also, $0 \le f \le 1$ is the "forgetting" factor. This factor can be used to account for the time characteristics of the events that are being monitored. For example, it can account for events that are active only during a window of time of the observation interval or events that turn on and off at various time instances.

The *area coverage* is defined as

$$C_k = \frac{1}{X \times Y} \times \sum_{i=1}^{X} \sum_{j=1}^{Y} G_k\left(i, j\right) \tag{9.3}$$

If $f = 1$ then C_k represents the area coverage over a time interval $[0, k]$ and it is an appropriate quality metric for applications that require coverage of all locations within some time interval.

9.4 Optimal Search Strategy

To make the analysis more concrete, in this section consider the problem of finding the optimal path that minimizes the expected detection time of an event that has occurred in either of h uncovered locations (coverage holes). Let T be the generally random time when an event has been detected, then the objective of this chapter is to determine the paths of the mobile sensors in order to solve (P).

$$(P) := \min\{\mathbb{E}\left[T\right]\}$$

where $\mathbb{E}\left[\cdot\right]$ denotes the expectation operator.

9.4.1 NP-completeness

In principle, finding the optimal path when h locations are possible and when M mobiles are available is a problem that is more complex compared to the traveling salesman problem (TSP) or the multiple TSP (mTSP) [138] which are widely known to be NP-complete problems. In the TSP the objective is to find the minimum distance tour through a set of N cities, visiting each city exactly once and returning to the starting city. The TSP is an NP-hard problem in combinatorial optimization, the optimal tour can be found using exhaustive search on $(N-1)!$ paths for N cities which is computationally intractable (takes $O(n!)$ running time) even for a small number of cities. Dynamic programming yields a much faster solution, though not a polynomial one (takes $O(n^2 2^n)$ running time).

In the following, it is shown that the problem defined above is NP-complete. This implies that we can not hope to find an optimal solution of an arbitrary instance of the problem in reasonable time (intractable) but more efficient search approaches are needed. Furthermore, it is desirable to determine solution strategies that are distributed in nature, thus each mobile can determine its path by itself utilizing only information that is locally available since such strategies will significantly reduce the communication overhead and can easily adapt to changes in the environment (node failures)

Theorem 9.4.1. *The optimal search path problem (OSPP) is* NP*-complete*

Proof. We first show that $OSPP \in NP$. Given an instance of the problem, we use as a certificate the sequence of h coverage holes searched in the path. The verification algorithm checks that this sequence contains all holes (each hole exactly once), sums up the costs (euclidian distances for inter-hole traveling and hole searching), and checks whether the sum is the minimum. This process can certainly be done in polynomial time.

To prove that $OSPP$ is NP-hard, we describe a reduction [139] from an arbitrary instance of a well-known NP-hard problem, namely the Euclidean path TSP ($EpTSP$) [140], to a special instance of $OSPP$. Given an $ETSP$ instance, (N, d_{ij}), where N is the number of cities and d_{ij} denotes the matrix with inter-city distances, we choose the following parameters of $OSPP$ such that the achieved optimal solution corresponds to that of the $EpTSP$:

1. Number of mobile nodes M
2. Start and finish depots for all mobile nodes,
3. Stationary node deployment
4. Areas A_h of the coverage holes to be searched
5. Sensor node detection range r_d

We choose that $M = 1$ and this single mobile node is initially located in an arbitrary coverage hole (note that there is not a requirement of returning to the starting city in $EpTSP$ although such a requirement does not change the computational complexity of the ordinary TSP [140]), the stationary node deployment is such that enables a definite number of h coverage holes and the area of each A_i, $i = 1, \cdots, h$ is set to $A_i = 0$ and finally $r_d = 0$. Thus the optimal path of the mobile node is the path that visits all isolated uncovered points (city locations) with the minimum distance cost. Due to these choices, the optimal solution of this specially designed instance of the $OSPP$ will coincide with the optimal solution of the $EpTSP$. This completes the proof. $\square$

9.4.2 Optimal Solution for the single mobile - two hole problem

To simplify the analysis, next we consider the smallest instance of the problem which can provide some insight to the nature of the general solution. Consider the scenario where a single mobile node is available and an event has occurred in either of two locations ($h = 2$ coverage holes). The objective of the mobile is to decide which location to search first such that the detection delay $\mathbb{E}[T]$ is minimized. In sequel, we extend the analysis when h coverage holes exist. Before we proceed with the derivation of the optimal strategy, the following definitions are made.

D1 There are only two uncovered regions (coverage holes) with areas A_b and A_s ($A_b > A_s$) and centroids (geometric centers), $\mathbf{C_b}$ and $\mathbf{C_s}$ respectively. It is also assumed that there is no overlap between the two holes.

D2 An event can occur at any point of the uncovered region with equal probability.

D3 The mobile node is initially placed at position $\mathbf{O}$ at distance $d_b = \|\mathbf{C_b} - \mathbf{O}\|$ from hole A_b and at distance $d_s = \|\mathbf{C_s} - \mathbf{O}\|$ from hole A_s. The distance between the two coverage holes is indicated by $d_{sb} = \|\mathbf{C_s} - \mathbf{C_b}\|$.

D4 The mobile node detects an event within its detection range r_d with probability 1 while it detects events outside its detection range with probability 0.

D5 The speed of the mobile is v.

D6 An exhaustive search of a coverage hole with area A is performed in $A/(2r_d v)$ time units.

Using definitions D1 and D2 one can obtain the probability of an event occurring in either of the two holes

$$P(A_b) = \frac{A_b}{A_b + A_s}, \quad \text{and} \quad P(A_s) = \frac{A_s}{A_b + A_s}$$

Also, from definitions D2 and D5, given that the event has occurred in a hole of area A, then the time to detect the event (assuming exhaustive search) is uniformly distributed in the interval $[0, A/(2r_d v)]$. Therefore the average time to detect the event is simply $A/(4r_d v)$. Note that the uniform distribution assumption was made to simplify the analysis but the main results would also hold for any other event distribution.

Fig. 9.3 illustrates the geometry of the problem. An event that will occur in either A_b or A_s will remain undetected unless a mobile node finds it. A mobile node can reach

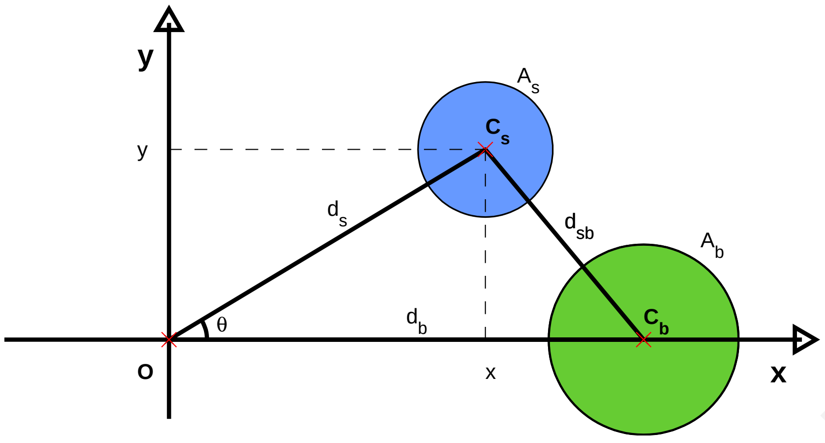


Figure 9.3: Problem geometry

either of the holes and search it sequentially, e.g., by following a path on concentric circles (spiral-in or spiral-out path) or making "S" shaped moves (parallel sweeps) until the entire area is covered. Next, consider the scenario where, given that an event has occurred, a mobile node initially located at **O** should conduct an exhaustive search of the two coverage holes in order to find it. In this scenario, the mobile has two options; it can either go and search A_b first and then to A_s or it can follow a reverse path, i.e., first search A_s and then A_b. In the following analysis, our objective is to determine the strategy that the mobile should follow in order to minimize the time to detect the event. The $\mathbb{E}[T]$ is given by

$$\mathbb{E}[T] = \mathbb{E}[T|A_s] \cdot P(A_s) + \mathbb{E}[T|A_b] \cdot P(A_b) \tag{9.4}$$

where $\mathbb{E}[T|A_j]$ denotes the expected T given that the event is located in A_j, $j \in \{b, s\}$. When the mobile follows the path from $\mathbf{O} \rightarrow \mathbf{C_s} \rightarrow \mathbf{C_b}$, the expected event detection time $\mathbb{E}[T_{sb}]$ is given by

$$\begin{aligned} \mathbb{E}[T_{sb}] &= \left(\frac{d_s}{v} + \frac{1}{2}\frac{A_s}{2r_d v}\right)\frac{A_s}{A_s + A_b} + \\ &= \left(\frac{d_s}{v} + \frac{A_s}{2r_d v} + \frac{d_{sb}}{v} + \frac{1}{2}\frac{A_b}{2r_d v}\right)\frac{A_b}{A_s + A_b} \end{aligned} \tag{9.5}$$

On the other hand, if the mobile follows the path from $\mathbf{O} \rightarrow \mathbf{C_b} \rightarrow \mathbf{C_s}$ the expected event detection time $\mathbb{E}[T_{bs}]$ is given by

$$\begin{aligned} \mathbb{E}[T_{bs}] &= \left(\frac{d_b}{v} + \frac{1}{2}\frac{A_b}{2r_d v}\right)\frac{A_b}{A_s + A_b} + \\ &= \left(\frac{d_b}{v} + \frac{A_b}{2r_d v} + \frac{d_{sb}}{v} + \frac{1}{2}\frac{A_s}{2r_d v}\right)\frac{A_s}{A_s + A_b} \end{aligned} \tag{9.6}$$

In order for the mobile to decide which path to follow, it needs to compute $\mathbb{E}[T_{sb}]$ and $\mathbb{E}[T_{bs}]$ and follow the path that minimizes the expected detection time. This decision problem (i.e. which is the minimum expected detection time between $\mathbb{E}[T_{sb}]$

and $\mathbb{E}[T_{bs}]$) can be reduced by comparing the terms of (9.5) and (9.6) and ignoring all common terms in the inequality $\mathbb{E}[T_{sb}] \lesseqgtr \mathbb{E}[T_{bs}]$. Consequently, we have

$$\mathbb{E}[T_{sb}] \lesseqgtr \mathbb{E}[T_{bs}] \Leftrightarrow d_{sb}(A_b - A_s) \lesseqgtr (d_b - d_s)(A_b + A_s) \tag{9.7}$$

Next, we consider the following cases:

C1 $\{A_b = A_s = A\}$: The decision problem $\mathbb{E}[T_{sb}] \lesseqgtr \mathbb{E}[T_{bs}]$ reduces to $d_s \lesseqgtr d_b$, i.e., the mobile should go to its nearest coverage hole first. The proof follows easily by substituting $A_b = A_s = A$ in eq. (9.7).

C2 $\{d_b = d_s = d\}$: The decision problem $\mathbb{E}[T_{sb}] \lesseqgtr \mathbb{E}[T_{bs}]$ reduces to $A_b \lesseqgtr A_s$, i.e., the mobile should go to the biggest hole first. The proof again follows easily by substituting $d_b = d_s = d$ in eq. (9.7).

C3 $\{A_b > A_s \textit{ and } d_b < d_s\}$: The decision is to always go to the biggest hole which is also located nearer to the mobile. The proof follows by comparing the terms of eq. (9.7). Using the conditions of this case, the right hand side is always less than the left hand side, therefore $\mathbb{E}[T_{sb}] > \mathbb{E}[T_{bs}]$.

C4 $\{A_b > A_s \textit{ and } d_b > d_s\}$: The decision depends on the relative position (d_{sb}) and area ratio ($\varrho = A_b/A_s$) of the bigger hole with respect to the smaller one. Specifically, if the smaller hole is located inside an "egg shaped" area then the decision is to search the smaller hole first, otherwise, it is better to search the larger hole first. The proof follows by solving eq. (9.8) defined by the cosines rule of the triangle in Fig. 9.3. Using the cosines rule we know that

$$d_{sb}{}^2 = d_s{}^2 + d_b{}^2 - 2 d_s d_b \cos(\theta). \tag{9.8}$$

Also, using some algebra, one can rewrite eq. (9.7) according to the condition that the mobile should visit the smaller hole first. Thus we have

$$d_{sb} \leq \frac{\varrho + 1}{\varrho - 1}(d_b - d_s) \tag{9.9}$$

where $\varrho = A_b/A_s > 1$. Substituting eq. (9.9) in eq. (9.8), we get a single equation (eq. 9.10) with one unknown, d_s which denotes the decision boundary that determines which hole will be visited first.

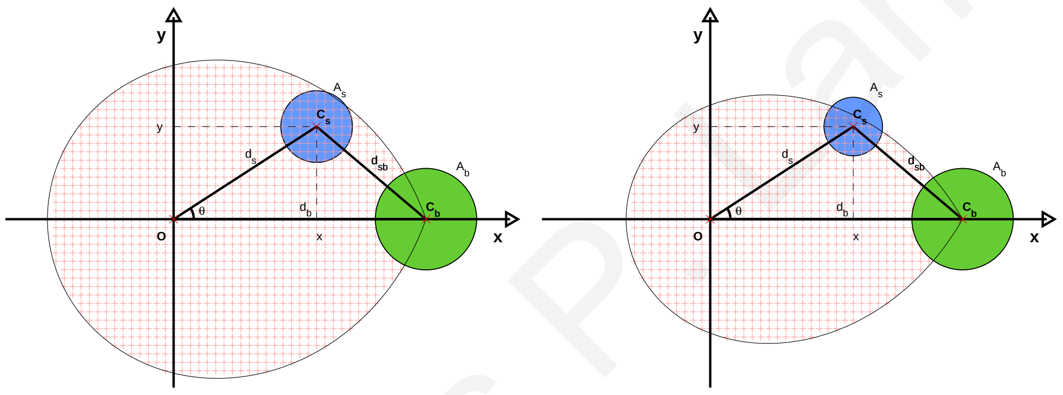
$$\left(\frac{\varrho + 1}{\varrho - 1}(d_b - d_s)\right)^2 = d_s{}^2 + d_b{}^2 - 2 d_s d_b \cos(\theta) \tag{9.10}$$

Therefore, the mobile should search the smaller hole first if its centroid is located within the egg-shaped region defined by the solution of eq. 9.10 in the polar coordinate system.

The accepted solution of the eq. (9.10) is the following (eq. (9.11)) where $r = d_s$

$$\begin{aligned} r &= \frac{d_b\left((\varrho+1)^2-(\varrho-1)^2 cos(\theta)-\sqrt{\left(\left((\varrho+1)^2-(\varrho-1)^2 cos(\theta)\right)^2-(4\varrho)^2\right)}\right)}{4\varrho} \\ \theta &= [0, 2\pi) \end{aligned} \tag{9.11}$$

This holds true when $\mathbf{O} = (\mathbf{0}, \mathbf{0})$. One can use eq. (9.11) to draw the region in polar coordinate system. This egg-shaped region is illustrated in Fig. 9.4 and indicates that if the centroid of the small hole is located inside this shape the expected detection time is minimized when following the $\mathbf{O} \rightarrow \mathbf{C_s} \rightarrow \mathbf{C_b}$ path. It is also important to consider that for the case when $\varrho = 1$ (case C1 $\{A_b = A_s = A\}$) the egg-shaped region becomes a circular region ($r = d_b$ in eq. (9.11)) which means that its better to follow the $\mathbf{O} \rightarrow \mathbf{C_s} \rightarrow \mathbf{C_b}$ only if $d_s < d_b$.



(a) The Egg-shaped region for $\varrho = 2$ (b) The Egg-shaped region for $\varrho = 3$

Figure 9.4: The Egg-shaped region for $\varrho = 2$, $\varrho = 3$ and $\mathbf{O} = (\mathbf{0}, \mathbf{0})$.

Finally, in the more general case, when $\mathbf{O} = (\mathbf{x_0}, \mathbf{y_0}) = (\mathbf{r_o}, \theta_\mathbf{o})$ the analytical equation can be found using two translations of the polar coordinate system. The first translation will be a rotation of the polar axis through an angle of θ_o, thus we have

$$\begin{aligned} r' &= r(\theta - \theta_o) \\ \theta' &= \theta = [0, 2\pi) \end{aligned} \tag{9.12}$$

The second translation will be a move from $(0, 0)$ to (r_o, θ_0) of the polar axis, thus now we have

$$\begin{aligned} \mathrm{r}'' &= \sqrt{r'^2 + r_o^2 + 2r'r_o \cos(\theta - \theta_o)} \\ \theta'' &= arctan\left(\frac{r' \sin(\theta) + r_o \sin(\theta_o)}{r' \cos(\theta) + r_o \cos(\theta_o)}\right) \end{aligned} \tag{9.13}$$

Fig. 9.5 illustrates the formed egg-shaped region in the case when $\varrho = 2$ and $\mathbf{O} = (\mathbf{r_o}, \theta_\mathbf{o})$.

Concluding the analysis above, one can easily find the exact optimal solution to the $OSPP$ when $h = 2$ by evaluating eq. 9.7 given the locations and areas of the two coverage holes in the WSN field.

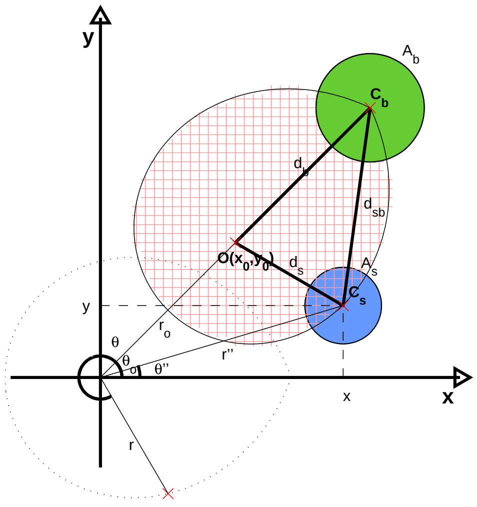


Figure 9.5: The Egg-shaped region for $\varrho = 2$ and $\mathbf{O} = (\mathbf{r_o}, \theta_\mathbf{o})$ where if $\mathbf{C_s}$ is located inside that shape then by following $\mathbf{O} \to \mathbf{C_s} \to \mathbf{C_b}$ path, the expected event detection time is minimized

9.4.3 Optimal Solution for the single mobile - h hole problem

Now consider the scenario where there are $h = 3$ coverage holes (see Fig. 9.6). In

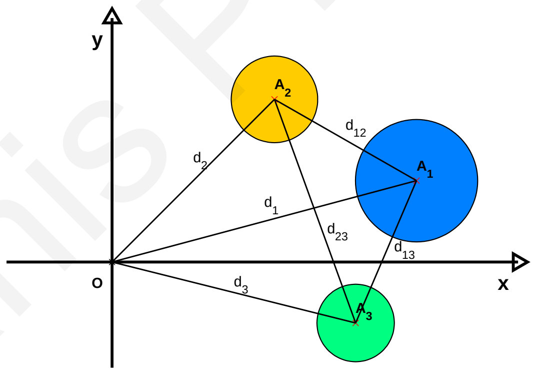


Figure 9.6: Problem geometry with three coverage holes

such case, one can use the following technique to reduce the problem to the $h = 2$ case as follows: For each i, $i = 1, \cdots, 3$ consider that the mobile is at A_i and has already search the A_i hole, then it decides where to go next using eq. (9.5) and eq. (9.6). Thus the expected event detection time for $i = 1$ can be given by

$$\begin{aligned}\min\left(\mathbb{E}\left[T_{123}\right], \mathbb{E}\left[T_{132}\right]\right) = \left(\tfrac{d_1}{v} + \tfrac{1}{2}\tfrac{A_1}{2r_d v}\right)\tfrac{A_1}{A_1+A_2+A_3} + \\ \left(\min\left(\mathbb{E}\left[T_{23}\right], \mathbb{E}\left[T_{32}\right]\right) + \tfrac{d_1}{v} + \tfrac{A_1}{2r_d v}\right)\tfrac{A_2+A_3}{A_1+A_2+A_3}\end{aligned} \tag{9.14}$$

Finally the optimal path should be found by comparing the cases of all i, $i = 1, \cdots, 3$ to find the path that minimizes the expected detection time. Thought the proposed solution needs $h!-1$ comparisons-evaluations to find the optimal path when h uncovered holes exist, it easy to tackle the equations using the proposed technique. Note that the $OSPP$ problem can not easy formulated in Dynamic Programming as the TSP

problem because the search cost between two holes is not a constant value but depends on the previous holes that have been already searched.

Next, we present a centralized algorithm for finding the optimal tour for a single mobile node when h non-overlapping holes exist using exhaustive search. In this algorithm we generate all possible permutations (paths) and compute the expected detection time associated with each path. The optimal solution is the path that results in the minimum average detection delay. The details of the algorithm are listed in Fig. 9.7.

9.4.4 Discussion

Finding the optimal solution to the problem by using technics as branch and bound or dynamic programming is still not computationally tractable as the algorithms are intended to run on tiny microcontrollers of sensor nodes. Therefore, knowing that $OSPP$ is *NP*-complete, we can not hope to solve all problem instances to optimality in reasonable time (as h is usually large) but we must adopt a heuristic solution method that provides a close to optimal solution.

The solution to the single mobile, two-hole problem indicates that rather than first searching the areas that are most likely to hide an event, often it is optimal (with respect to the detection delay) to search areas that are located closer to the current position of the mobile even if it is less likely to find an event in them. In addition, from the derived egg-shaped region (see fig. 9.4(b)), there is always a region *around* the mobile where if a small hole is located, then it should be searched first even if a larger hole is present. This implies that a “local” search (i.e., a search closer to the current location of the mobile) is better than a more “global” search (i.e., a search in a larger uncovered area far away from the mobile).

This motivates a heuristic centralized approach to solve the single mobile - several hole problem as follows: Given that the mobile has *global information* regarding the coverage holes of the sensor field (i.e. knows the number, the centroid and area of each hole), it can decide whether to go and search the *biggest* uncover region in the field *or* the *nearest* uncovered region. The decision is based on eq. 9.9 (i.e. decides based on egg-shape region). Once the mobile has searched the decided coverage hole decides the next hole to search based on its current position and the remaining holes of the field. More information about this heuristic approach is presented in section 9.6.

However, in the context of large and randomly deployed WSNs, it is infeasible to have a central controller to solve the problem and thus the proposed solution must be implementable in a distributed fashion and based on local-accurate information. In addition, it is needed to support multiple mobile nodes and to be dynamic because coverage holes might change their areas and centroids as some stationary or mobile sensors failed and/or multiple mobiles are searching the WSN field.

This indicates that the proposed path planning algorithm presented in the previous

Optimal Search Path Algorithm using Exhaustive Search

```
function OSP ($\mathbf{O}, r_d, v, h, \mathbf{C}, \mathbf{A}$)
    Input:  $\mathbf{O}$: initial position of mobile sensor
            $r_d$: detection range of mobile sensor
            $v$: velocity of mobile sensor
            $h$: Number of uncovered regions
            $\mathbf{C}$: Centroids of uncovered regions
            $\mathbf{A}$: Areas of uncovered regions
    Output: Optimal Path (ordered set of uncovered regions)
    /* Find all h! possible permutations (Johnson-Trotter algorithm)*/
1: $\mathbf{P}(i,j) = \mathbf{Permutations}(h)$; $i = 1,...,h!$,  $j = 1,...,h$
2: $ET_{\mathbf{P}(i,:)}(i) = 0$; $i = 1,...,h!$
    /* Create distance matrix*/
3: for $i = 1 : h$
4:     for $j = 1 : h$
5:         $d(i,j) = \|\mathbf{C}(i) - \mathbf{C}(j)\|$;
6:         if $i == j$
7:             $d(i,j) = \|\mathbf{C}(i) - \mathbf{O}\|$;
8:         end
9:     end
10: end
    /* Find Optimal Path */
11: for each path $\mathbf{P}(i,:)$, $i = 1,...,h!$
12:     for each hole $\mathbf{P}(i,j)$, $j = 1,...,h$
13:         $h_f = \mathbf{P}(i,1)$; /* first hole in $\mathbf{P}(i,:)$ */
14:         if $h_f == \mathbf{P}(i,j)$
15:             $t(j) = d\big(\mathbf{P}(i,j), \mathbf{P}(i,j)\big)/v + (1/2)\big(\mathbf{A}(\mathbf{P}(i,j))/(2r_d v)\big)$;
16:         else
17:             $t(j) = t(j-1) + (1/2)\big(\mathbf{A}(\mathbf{P}(i,j-1))/(2r_d v)\big) +$
 :                 $d\big(\mathbf{P}(i,j-1), \mathbf{P}(i,j)\big)/v + (1/2)\big(\mathbf{A}(\mathbf{P}(i,j))/(2r_d v)\big)$;
18:         end
19:         $Et(j) = t(j)\big(\mathbf{A}(\mathbf{P}(i,j))/\sum(\mathbf{A})\big)$;
20:     end
21:     $ET_{\mathbf{P}(i,:)}(i) = \sum(Et)$;
22: end
23: $i^* = \arg\min_{i=1,...,h!}\{ET_{\mathbf{P}(i,:)}(i)\}$
24: Optimal Path= $\mathbf{P}(i^*,:)$
```

Figure 9.7: Pseudo code for finding the optimal search path using exhaustive search

chapters, though does not explicitly solve the $OSPP$, it provides a good heuristic solution and also tackles the more general case where several overlapping holes exist and multiple mobile nodes are searching the WSN field. In this approach, each mobile node searches for coverage holes in a small circular region of radius r_z around the itself instead of the derived egg-shaped region. Therefore it is meaningful to study how the radius r_z of the search neighborhood affects the event detection performance of the propose algorithm.

In the next section, we present briefly the distributed path planning algorithm that can be used by each of the mobiles to search the event with the minimum detection delay.

9.5 Mobile Distributed Path Planning Algorithm

In this section we present the path planning algorithm that can be used by mobile node(s) in order to search an area for the event. The objective of each mobile is to collaborate with the sensor nodes (stationary or mobile) in WSN field so it will search areas not covered by the other nodes. Furthermore, the approach is dynamic in the sense that each mobile determines the coverage holes at every step in order to capture possible changes in the environment, e.g., failures of stationary nodes. Before we proceed, let us present the information structure that is needed by mobile node(s) in order to run the path-planning algorithm. Each mobile uses an $X \times Y$ matrix P_k^m, $m \in \mathcal{M}$ where it keeps the state of the field. Ideally P_k^m should remain $P_k^m = G_k$ at all times k, since the matrix G_k represents the accurate global state of the field which is used for the computation of the area coverage C_k. Clearly, in a dynamic environment where several sensors move, fail or more sensors are added, it is impossible to guarantee that $P_k^m = G_k$ at all times. However, we emphasize, that the proposed algorithm, that will run by a mobile located at some position $\mathbf{x}_m(k)$, computes its path based *only* on local information, i.e., information in the submatrix of P_k^m that corresponds to the cells $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$, and thus, it is sufficient to have accurate information only for the $\mathcal{D}_{\bar{r}_c}(\bar{\mathbf{x}}_m(k))$ submatrix. This is easily attainable since the required information can be obtained from the one-hop neighbors.

9.5.1 Path Planning

The path planning algorithm is based on Receding-Horizon approach where at each step the mobile's controller evaluates the cost of moving to a finite set of candidate positions and moves to the one that minimizes an overall cost. Suppose that during the kth step, the mobile node is at position $\mathbf{x}(k)$ and is heading to a direction θ. The next candidate positions are the $\nu \in \{2n+1, \forall n \in \mathbb{Z}^+\}$ points $\mathbf{y}_1, \cdots, \mathbf{y}_\nu$ that are uniformly distributed on the arc that is ρ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$. The mobile node evaluates a cost function $J(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \cdots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k+1) = \mathbf{y}_{i^*} = \mathbf{x}(k) + \rho.e^{\mathrm{i}(\theta+\varphi_{i^*})}$ where i is the imaginary unit and i^* is the index that minimizes $J(\mathbf{y}_i)$.

$$J(y_{i^*}) = \min_{1 \le i \le \nu} \{J(\mathbf{y}_i)\} \tag{9.15}$$

In this model, θ is the direction that the mobile is heading, ρ is the distance that the mobile can cover in one time step, ϕ is the maximum angle that the mobile can turn in a single step, and ν is the number of candidate positions that is being evaluated for the next step.

The objective function that each mobile is trying to minimize is of the form

$$J(\mathbf{y}) = \sum_j w_j J_j(\mathbf{y}) \tag{9.16}$$

where $J_j(\cdot)$ is a specific objective and w_j's are non-negative constant weights such that $\sum_j w_j = 1$.

As indicated by the analysis of the previous section, the optimal policy is to head towards the bigger hole in the area but in the way, the mobile should search smaller holes that exist in the egg-shaped area around it. Thus, the objective (9.16) should be such that the mobile will approximate that behavior. After an extensive investigation [136], three specific normalized functions have been selected: $J_t(\cdot)$ which penalizes positions that are away from large coverage holes and $J_s(\cdot)$ and $J_c(\cdot)$ which penalize positions that are close to regions been covered by other sensors (stationary or mobile). How $J_t(\cdot)$, $J_s(\cdot)$ and $J_c(\cdot)$ are computed is presented next.

9.5.2 Path Cost Functions

The objective of $J_s(\cdot)$ function is to push the mobile away from areas covered by other sensors, thus the $J_s(\mathbf{y})$ used involves a repulsion force that pushes the mobile away from its closest neighbor. The form of this function is given by

$$J_s(\mathbf{y}) = \max_{j \in \mathcal{H}_{r_c}(m)} \left\{ \exp\left(- \frac{\|\mathbf{y} - \mathbf{x}_j\|^2}{r_d^2} \right) \right\} \tag{9.17}$$

where $\mathcal{H}_{r_c}(m)$ is the set of all nodes in the communication range r_c of the mobile m. The detection range r_d quantifies the size of the region around the mobile m to be repelled by its neighbors.

The cost function $J_c(\mathbf{y})$, similarly to J_s, is designed to push the mobile away from areas that have been covered by other sensors (stationary or mobile) using the relevant information from the cognitive map of the mobile node. This function takes a larger value if the candidate position is adequately covered by other sensors and a small value otherwise. This cost function is given by

$$J_c(\mathbf{y}) = \frac{1}{\pi r_d^2} \sum_{\{i,j\} \in \mathcal{D}_{\bar{r}_d}(\bar{\mathbf{y}})} P_k(i,j) \tag{9.18}$$

where $\mathcal{D}_{\bar{r}_d}(\bar{\mathbf{y}})$ is the set of cells existing in a discretized disk of the mobile's P_k matrix, centered at the position $\bar{\mathbf{y}} = \lceil \mathbf{y} \rceil$ with radius of $\bar{r}_d = \lceil r_d \rceil$. This objective enables the mobile nodes to search large coverage holes efficiently when stationary sensors are absent and the hole is covered only by the mobile.

Even if the mobile should search holes in its immediate vicinity, it should eventually make some progress towards approaching the larger coverage hole. This is achieved by the target[1] function. In this function, each mobile has a destination point $\mathbf{x}_t$ (e.g., the centroid of the larger coverage hole), and the target cost $J_t(\mathbf{y})$ is a function that pulls the mobile towards its target. $J_t(\mathbf{y})$ is a function of the distance between the mobile and the target position and should take smaller values as the mobile moves towards the target destination thus for the purposes of this paper it is given by

$$J_t(\mathbf{y}) = \frac{\|\mathbf{y} - \mathbf{x}_t\|}{r_z}. \tag{9.19}$$

An important parameter of the path planning algorithm is the radius of the search area where a target position can be found r_z. The importance of the parameter will be discussed in the next section.

To compute $J_t(\cdot)$, one needs to determine a target position $\mathbf{x}_t$ and in our case the target position is assumed the centroid of the largest coverage hole in the search area defined by r_z. This is achieved by the Zoom algorithm [9] which is an efficient algorithm that can run at every step k. The idea of the algorithm is to divide the submatrix of P_k^m that corresponds to the cells $\mathcal{D}_{\bar{r}_z}(\bar{\mathbf{x}}_m(k))$ in four equal segments, and choose the segment with the maximum number of empty cells i.e the segment with the maximum number of cells with $P(i,j) = 0$ and repeats until either the segment size is equal to a single cell or until all segments have the same number of empty cells. In the first case, the hole center position will be the center of the cell. In the second case, the hole center position will be the center of the segment during the previous iteration. The algorithm is based on the divide-and-conquer principle and as such it is very efficient and can run repeatedly even on simple microprocessors. We point out that both cost functions used in (11.3) can be easily computed by a mobile node using information in its cognitive map or by obtaining information from its one-hop neighbors.

For completeness, note that another cost function that prevents mobiles from stepping outside the field exist. This boundary cost function $J_b(\mathbf{y})$ penalizes all candidate positions $\mathbf{y}$ that are not included in the field area $\mathcal{A}$ and is given by

$$J_b(\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \notin \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{9.20}$$

All cost functions used in (9.16) can be easily computed by a mobile node using information in its cognitive map or by obtaining information from its one-hop neighbors.

[1]Not to be confused with the targets (or events) that the mobiles are searching for.

9.5.3 Search Neighborhood r_z

An important consideration for the above path planning algorithm is the size of the neighborhood r_z where the mobile needs to search for its target (largest coverage hole). Given the analysis of the previous section, the mobile should not always head towards *the biggest* hole since this may not result in fast event detection. Rather, it should head towards big enough holes that are located close to the mobile. Therefore, the objective of the mobile should be to look for a big enough (not necessarily the biggest) hole that is located relatively close to it (search for a "local" big enough hole rather than search for the biggest "global" one). In a sense, this is equivalent to searching for the biggest hole in a small enough neighborhood around the mobile. Also note that a smaller r_z is advantageous since it implies that less information is needed for the coverage hole estimation.

Therefore, the proposed algorithm searches for the biggest hole in a radius r_z around the mobile. An important question is how big should r_z be. If r_z is too small, then there is a risk that the search strategy of the mobile will be "myopic" always searching in smaller holes never reaching the larger holes. On the other hand, if r_z is too big, then the mobile will give more priority to larger holes that are located far away ignoring large enough holes that are located close to it. Thus, an optimal r_z exists, however it is not easy to compute. This question is also addressed in Section 9.6.

9.5.4 Distributed Collaboration between Mobile Nodes

When multiple mobile nodes are used, it is desirable that they collaborate so they can better achieve their tasks. In this sense, it is desirable that each mobile searches different areas to avoid duplication of work. Due to the localized nature of the proposed algorithm (it uses only the information in r_z), if two mobiles are located sufficiently far apart, then they are guaranteed to search different coverage holes which is advantageous since duplication of work is avoided. A possible problem arises when the two mobiles come sufficiently close to each other. In this case, it is very likely that the information they will use to estimate the next target position will be the same and as a result they will all estimate the same target location. Clearly, this is not a good collaboration strategy since there is no benefit if they *all* converge to the same point. To avoid this problem we utilize a collaboration protocol that enables mobile nodes to exchange some information in order to avoid converging to the same point.

When two mobiles come into communication range r_c for the first time (they are out of communication range at step $k-1$ but they are in communication range at k) they exchange their cognitive map P_k^m, thus they now what areas each one has searched so far. From this point onward, at every step, the mobiles exchange their current location as well as their computed target location (i.e., the centroid of the biggest coverage hole in their respective r_z's).

After mobile node i has exchanged collaboration messages with its neighboring mobiles it has all the necessary information to execute the collaboration protocol. Thus at first, it merges its cognitive map P_k^i with the cognitive maps P_k^j, received from its neighbors, so that it does not explore areas already explored by other mobile nodes. Afterwards, the mobile node i utilizes the current locations $\mathbf{x}^j(k)$ and dynamic target coordinates $\mathbf{x}_t^j(k)$ received from its neighboring mobiles $j \neq i$ in order to avoid going towards the same point. Once the mobile i has received all target points from its neighbors, it forms the matrix $D_{\bar{r}_z}(\bar{\mathbf{x}}_i(k))$ (which is a copy of the set of P_k^i cells that corresponds to the distance r_z from the current position of mobile i) and updates the $D_{\bar{r}_z}(\bar{\mathbf{x}}_i(k))$ matrix by assuming that these targets points constitute covered areas. Finally, it executes the zoom algorithm [9] where the input is the $D_{\bar{r}_z}(\bar{\mathbf{x}}_i(k))$ matrix and the output is the dynamic target point $\mathbf{x}_t^i(k)$ of the mobile node i which is definitely different that the target points of its neighboring mobiles.

With this simple scheme, the mobiles avoid exploring the same areas. This scheme has some important benefits. It is distributed (no need for a central controller), it is simple, and utilizes only local information (the relevant information in the submatrix $\mathcal{D}_{\bar{r}_z}(\bar{\mathbf{x}}_i(k))$, which corresponds to the neighborhood r_z of the cognitive map P_k^i). This approach is also very "intuitive" based on the discussion of Section 9.4.4. If there are two mobiles and two coverage holes, then one will decide to go to its closer hole while the other will consider part of that hole covered thus it will probably decide to go to the second hole unless the first hole is significantly larger than the second.

9.6 Simulation Results

In this section we present a representative scenario of the movement of a mobile node to illustrate the behavior of the proposed path planning algorithm. We assume a sensor field with 300 randomly deployed stationary sensors in a $200m \times 200m$ area. The detection radius of all sensors is $r_d = 5m$ and the communication range $r_c = r_z + r_d = 30m$. The weights are set to $w_t = w_s = 0.49$, $w_c = 0.02$ and the mobile maneuverability parameters are set to $\rho = 1m$ and $\phi = 35^\circ$ while for every decision $\nu = 10$ candidate next positions are considered. In this simulation scenario a static event that is not detected initially by the stationary sensor field exists and it is indicated by the $+$ sign.

Fig. 9.8 shows how the mobile node navigates through the sensor field, sampling points that are not adequately covered by the stationary sensors. As seen from the path followed, there is collaboration between the mobile and stationary sensors in the sense that the mobile has found a path that is least covered by the stationary sensors and also the undetected event is now been detected.

Fig. 9.9 also shows representative scenario of the movement of a team of five mobile nodes. In this simulation we use the same set of parameters as described previously except the number of randomly deployed stationary sensors which is now 200. Fig. 9.9 shows how the five mobile nodes navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors. As seen from the paths followed, there is collaboration between mobile and stationary sensors in the

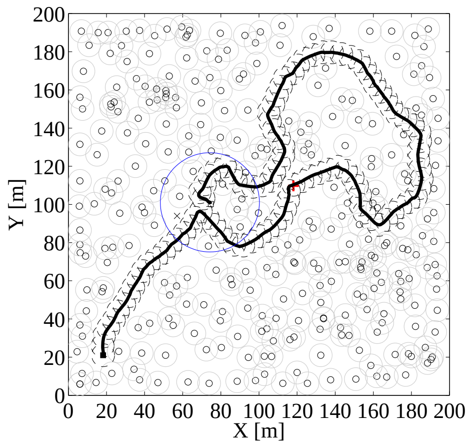


Figure 9.8: Dynamic path planning using $M = 1$ mobile node.

sense that the mobiles have found five different paths that are least covered by the stationary sensors. Notice that the five mobiles collaborate and select different targets at the beginning of their motion and that the undetected event has now been detected by a mobile node.

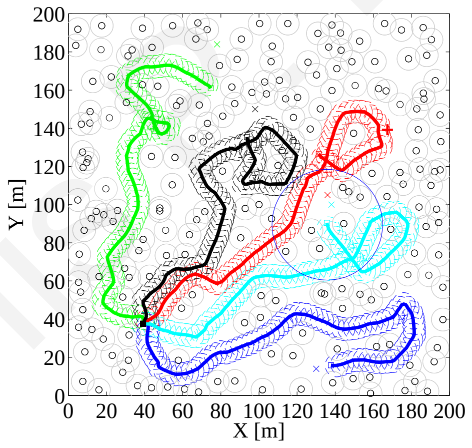
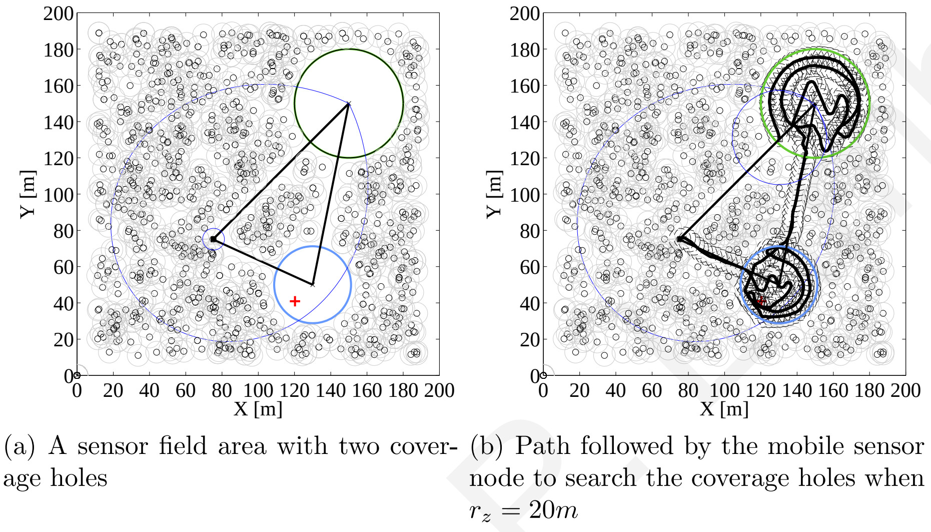


Figure 9.9: Dynamic path planning using a team of $M = 5$ mobile nodes.

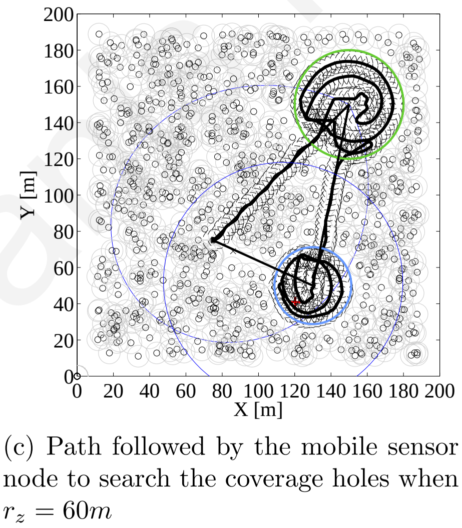
Next, we provide a simulation experiment to illustrate the behavior of the proposed path planning algorithm with two different r_z values for the scenario presented in section 9.4. As previously mentioned, r_z is the searching range where the mobile node founds its dynamic target (finds the coverage hole center). Fig. 9.10(a) shows a sensor field area where two coverage holes exist. Such WSN field deployment could appear when stationary sensors destroyed in the particular regions due to due to malicious action or environmental effects (e.g. shadowy regions prevent solar battery charging). The smaller hole is closer to the mobile node and the big hole has an area $A_b = 2 \times A_s$. The egg-shaped area is also shown in the figure. Clearly, the mobile node should follow a path from its current location towards the small hole, search the small hole and afterwards continue towards the big hole and search the big hole. Fig. 9.10(b) shows the path followed by the mobile node when $r_z = 20m$ and Fig. 9.10(c) shows the path followed by the mobile node when $r_z = 60m$.

It turns out that when the mobile node is searching for events "locally" the expected event detection time could be minimized with high probability (i.e. the mobile in Fig. 9.10(b) follows the optimal search path according to the analysis provided in section 9.4). This behavior of the proposed path planning algorithm is very important as it allows the algorithm to make local decisions for enhancing the system's global performance as it is infeasible or costly to continuously collect information from a large WSN, moreover enables the path planning algorithm to be adaptive and requiring limited amount of information and computation.



(a) A sensor field area with two coverage holes

(b) Path followed by the mobile sensor node to search the coverage holes when $r_z = 20m$



(c) Path followed by the mobile sensor node to search the coverage holes when $r_z = 60m$

Figure 9.10: Paths followed by the mobile sensor node to search the coverage holes of the sensor field when two different r_z values are used in the proposed path planning algorithm

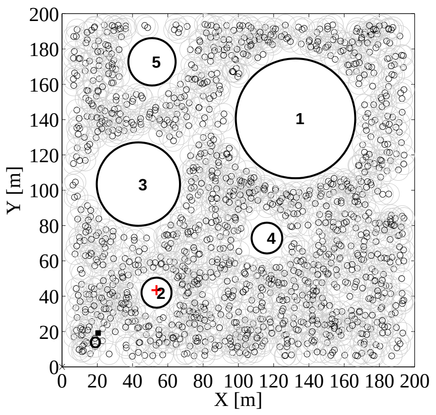
In the next simulation results we provide a comparison between *two different heuristic path planning methods* for the scenario presented in section 9.4 where more than two ($h > 2$) circular and non overlapping coverage holes exist in the sensor field. We consider the following two path planning algorithms:

- *CPP*: This Centralized Path Planning algorithm is based on a heuristic derived by analytical results presented in section 9.4. In this algorithm, the mobile has *global information* regarding the coverage holes of the sensor field (i.e. knows the number, the centroid and area of each hole) and utilizes the following heuristic decision. At the beginning or once a coverage hole has been searched, the mobile decides whether to go and search the biggest uncover region in the field or the nearest uncovered region. This decision is based on based on eq. 9.9 (i.e. decides based on egg-shape region). Once the mobile has searched the decided coverage hole decides the next hole to search based on its current position and the remaining holes of the field. Once the hole to search has been decided, the mobile navigates on a straight towards that hole. After the mobile enters the hole, it utilizes a coverage path planning algorithm to search the hole. Various coverage path planning methods for covering well defined areas have been proposed in the literature [141, 59], however for the purposes of this simulation we use the method presented in section 9.5.1 with two cost functions J_t and J_c. (see eq. (9.18,9.19)) and the target is defined by zoom algorithm where the search radius is set equal to the radius of the coverage hole been searched.

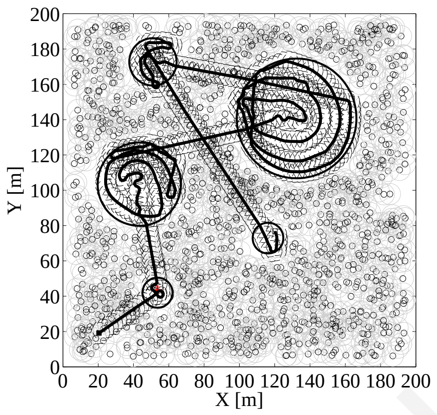
- *DPP*: This Distributed Path Planning is the proposed algorithm described earlier in section 9.5. This is an adaptive and on-line algorithm in the sense that the mobile uses only *accurate local information* in order to decide where and how to search the uncovered region of the sensor field (i.e. does not know the number, centroids or the uncovered regions). The mobile navigates using the method presented in section 9.5.1 with three cost functions J_t, J_s and J_c. The target of the J_t is defined at each step k using the zoom algorithm with a fixed search radius r_z. When this algorithm is used by the mobile to search holes, in the case when only few circular and non overlapping coverage holes exist in the sensor field, the mobile seems to navigate towards the nearest-largest hole in a greedy way. Note that "nearest" is defined by r_z radius. Therefore, using this algorithm, sometimes the mobile might missed searching a hole completely (because a near by hole is bigger) and sometimes the mobile might miss to cover a hole at all (because the remaining field is completely covered and the r_z is very small to catch the uncovered hole). To conclude, note that the searching neighborhood r_z radius is an important parameter of this algorithm and this algorithm works very well for the general case (when sensors are randomly deployed, see fig 9.2) and also supports easily collaboration between mobile nodes, when multiple mobiles are available.

Fig. 9.11 shows how the mobile node navigates to search the coverage holes of the sensor field using the proposed *CPP* and *DPP* path planing methods. Also for this particular scenario we find the optimal tour using the exhaustive search algorithm presented in Fig. 9.7. The optimal path indicates that the mobile should follow the path from $\mathbf{O} \rightarrow \mathbf{2} \rightarrow \mathbf{3} \rightarrow \mathbf{1} \rightarrow \mathbf{5} \rightarrow \mathbf{4}$ in order to search for the event with the minimum expected detection delay.

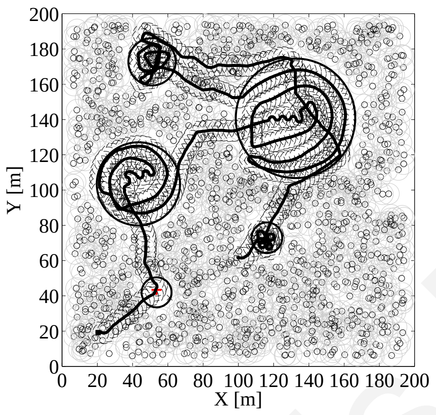
To further investigate the performance of the previous two path planning algorithms we compare the average probability of not detecting the event until time k and average event detection time accomplished over 100 sensor fields after 1000 moving steps of the mobile node. Each field contains 5 circular coverage holes that are randomly placed in



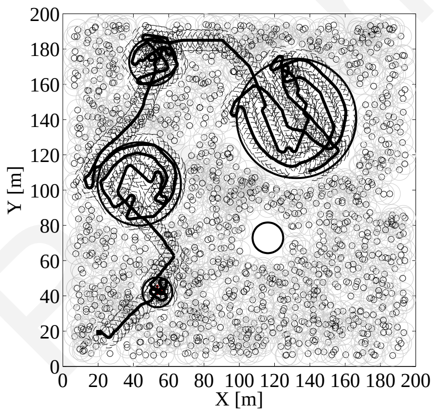
(a) The optimal path that the mobile should follow is the path from **O** $\rightarrow$ **2** $\rightarrow$ **3** $\rightarrow$ **1** $\rightarrow$ **5** $\rightarrow$ **4**



(b) Path followed by the mobile to search the coverage holes when CPP algorithm is used



(c) Path followed by the mobile to search the coverage holes when DPP algorithm is used with $r_z = 50m$

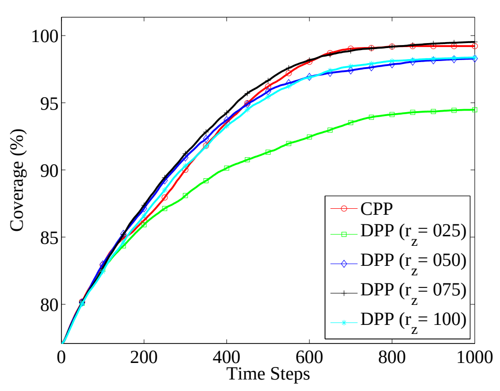


(d) Path followed by the mobile to search the coverage holes when DPP algorithm is used with $r_z = 20m$

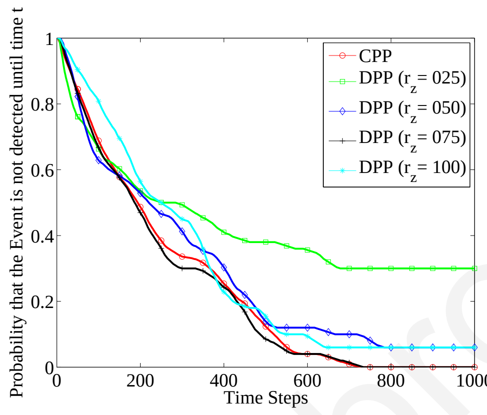
Figure 9.11: Paths followed by the mobile sensor node to search the four coverage holes of the sensor field using two different path planning algorithms

the sensor field and each hole has a random radius in the interval $[2r_d, 2r_d\sqrt{\varrho}]$, where $\varrho = 10$ is the area ratio of the possible biggest hole with respect to the possible smaller hole. Finally, each sensor field contains an undetected event which is randomly placed in one of the coverage holes. The purpose of this Monte Carlo simulation is to find the algorithm (*CPP* or *DPP*) that requires the minimum time to detect the event as well as to find the near optimum radius r_z of the searching neighborhood of the proposed path planning algorithm *DPP*.

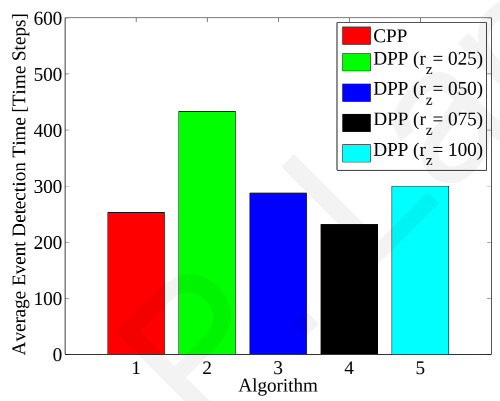
Results, depicted in Fig. 9.12, indicate that two of the algorithms, CPP and DPP with $r_z = 75m$, search all coverage holes completely (see Fig. 9.12(a)) and the average time to detect the event is minimized in comparison with the other algorithms (see Fig. 9.12(b),9.12(c)). Clearly the heuristic used in CPP algorithm results in fast event detection, but it can be used only when global knowledge of the sensor field deployment is available. However, DPP with $r_z = 75m$ algorithm has similar performance and uses less information and computation. It seems that $r_z = 75m$ is the near optimum r_z to



(a) Average area coverage.



(b) Average probability of not detecting the event.



(c) Average event detection time.

Figure 9.12: The average coverage, average probability of not detecting the event and average event detection time accomplished over 100 sensor fields by the mobile node after 1000 moving steps when *CPP* algorithm and *DPP* algorithm with different r_z values are used

search holes in such particular sensor field deployments (non overlapping holes that appear randomly in $200m \times 200m$ area). This r_z value is affected also from other parameters of the sensor field deployment such as the number of holes, the radius of the biggest hole etc.

If r_z radius is set below the optimum value i.e $r_z << 75m$, the performance is reduced. Not only in terms of fast event detection but also coverage. This is expected as the mobile seems to miss some holes that are placed far away from it and r_z searching radius is too small to find those holes. This is also shown in Fig. 9.12(a), where area coverage is not around 100% as well as in Fig. 9.12(b) where the probability on detection never approaches 0.

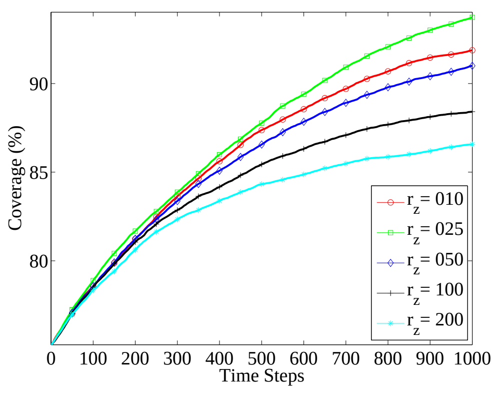
A counter intuitive results came out for the case where r_z is above optimum (e.g $r_z = 100m$). One would expect that as the mobile is able to catch all coverage holes of the sensor field, the performance should be better either in terms of coverage or in terms of fast event detection. However is this not how the algorithm works. In the case when r_z is very big (i.e $r_z >> 75m$), the mobile identifies the biggest coverage hole in the field and starts moving towards there. This is not good strategy as the mobile spends much time in moving between holes instead of search them, for instance, consider what happens when the first biggest hole is far away for the mobile and the second bigger hole is next to it. But even if the mobile goes to the biggest hole, the coverage should approximates almost 100% as moving steps increase to 1000, however Fig. 9.12(a) shows that this algorithm also avoids to provide complete coverage (i.e. some holes or parts of them never searched). This seems to holds true because as the mobile searches the biggest hole this biggest hole becomes smaller and smaller and once is smaller than any other hole in the field, the target (provided by zoom algorithm) changes and thus the mobile navigates towards the other hole (the current biggest hole) and does not complete the search of the first biggest hole.

To conclude with, if global knowledge of the sensor field in available and this knowledge does not changes over time, CPP algorithm provides a very good heuristic to search the coverage holes such that the expected time to detect an event is minimized. However this heuristic can not provide solution when the sensor field has several overlapping (connected) holes or in the general case when static sensors are placed randomly and/or the static deployment is very sparse. For all previous cases, DPP algorithm with a searching neighborhood of r_z radius that is very small compared to the sensor field area provides a very good heuristic both for coverage over time as well as for fast event detection. This is also proved in the next simulation results. This heuristic is valid because in a sparse sensor network (as the one shown in Fig. 9.2) the algorithm searches the largest coverage holes that are located near the mobile. Also for the case investigated in this simulation (non overlapping holes) there is also an optimum r_z radius that also provides a solution to the problem.

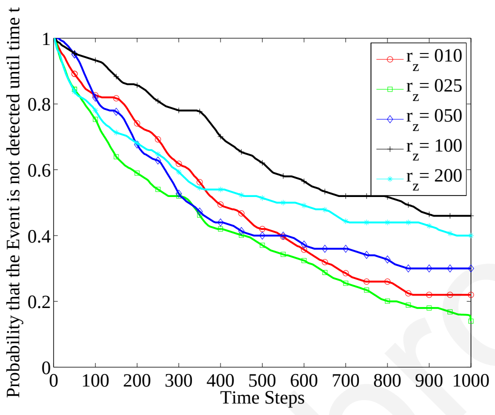
Therefore, if the mobile uses the heuristic to search first the largest holes that are located near to it (inside an r_z radius), there is an optimal r_z range that can provide fast event detection. If this optimum r_z is found (e.g. through monte carlo simulations), then the DPP algorithm provides even faster event detection compared to CPP algorithm. Finally, this heuristic can provide the solution using less information compared to the CPP algorithm and also the path can be found on-line and thus adapt to possible sensor field deployment changes.

In the next simulation, we investigate the average performance in terms of event detection time and area coverage of the proposed path planning for different r_z values using Monte Carlo simulation. We consider 100 sensor fields with 400 randomly distributed stationary sensors, each sensor field is deployed in a $200m \times 200m$ area and in each sensor field one undetected event exists. We present the average area coverage, the average probability of not detecting the event until time t and average event detection time accomplished over 100 sensor fields after 1000 moving steps by $M = 1$ mobile node at Fig. 9.13 and by $M = 5$ mobile nodes at Fig. 9.14.

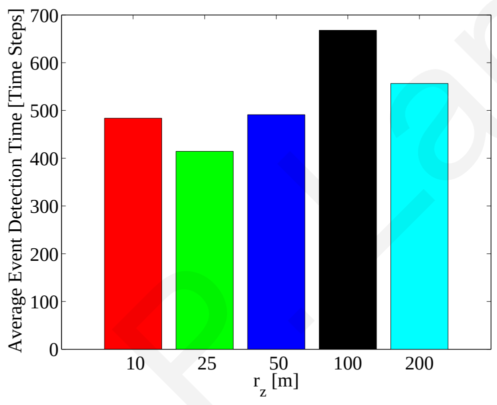
These simulation experiments indicate that when each mobile node is searching for an event in sparse stationary sensor fields, searching the event in "local" holes rather than searching it in a more "global" holes results in fast detection of the event. It is also shown that the neighborhood r_z should not be very big (compared to WSN area)



(a) Average area coverage.



(b) Average probability of not detecting the event.



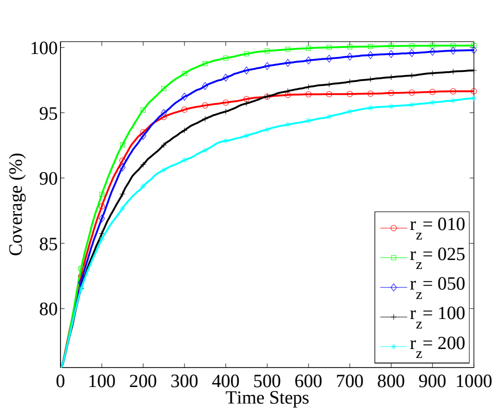
(c) Average event detection time.

Figure 9.13: The average coverage, average probability of not detecting the event and average event detection time accomplished over 100 sensor fields with 400 randomly distributed stationary sensors by $M = 1$ mobile node after 1000 moving steps for different r_z values.

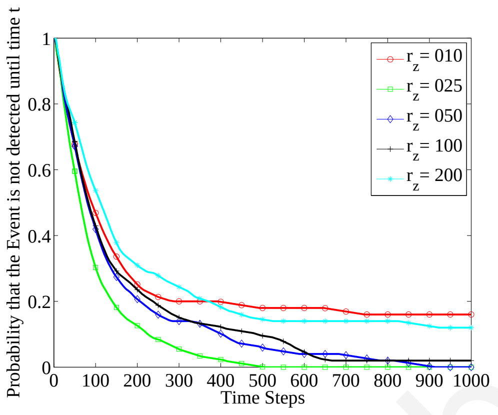
to avoid always heading towards the biggest hole but it should also be big enough to prevent the mobile for visiting very small holes compared to the detection range r_d of the mobile node. Also note that this trade off, also maximizes the area coverage (minimize the average probability that an event remains undetected).

Searching for the event “locally” also promotes the collaboration, when multiple mobile nodes exist, as it requires less information to be exchanged between sensor nodes, enables a distributed and simple method for dynamic target assignment and each mobile decides its next step more accurately as it utilizes updated information in its neighborhood.

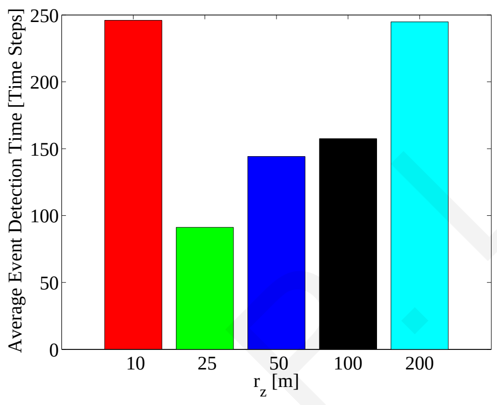
As indicated in simulation experiments of Fig. 9.13 and Fig. 9.14 the optimal r_z radius does not depend much on the number of mobile sensor nodes that move in the field and also the close to optimal value (r_z=25m) that is approximated by the simulation outcomes is much more less compared to the sensor field diagonal $\sqrt{2A}$. Of course, when we choose different parameters regarding the stationary sensor field deployment and/or the path planning algorithm a different optimal r_z value should



(a) Average area coverage.



(b) Average probability of not detecting the event.



(c) Average event detection time.

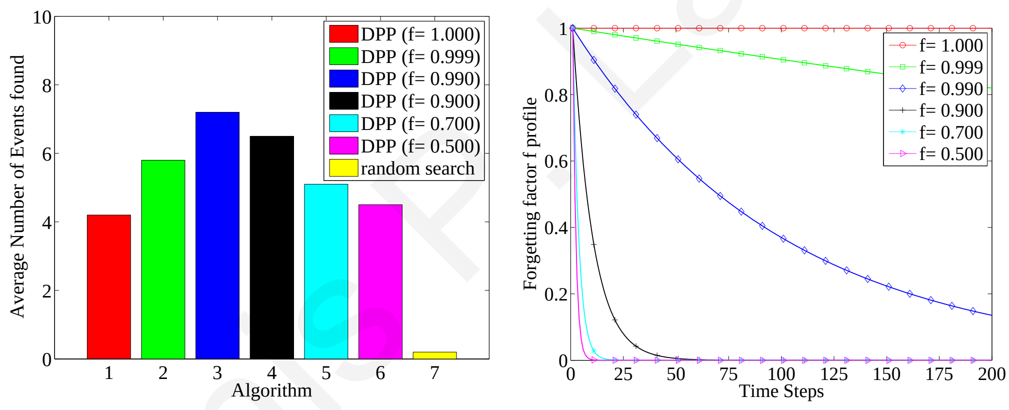
Figure 9.14: The average coverage, average probability of not detecting the event and average event detection time accomplished over 100 sensor fields with 400 randomly distributed stationary sensors by $M = 5$ mobile nodes after 1000 moving steps for different r_z values.

found. Nevertheless, using Monte Carlo simulation one can set and adjust the different parameters involved in the path planning based on the stationary sensor field set-up as well as the hardware constrains of the mobile nodes. However we point out that in the next chapter we further investigate the searching neighborhood r_z radius and in particular we derive an approximation method that estimates the r_z radius without extensive Monte Carlo simulations.

The next simulation investigates the performance of our approach in more dynamic environments where the events occur not only randomly in space but in time as well. Specifically, we consider scenarios where the events are activated at random times and have finite duration. In such scenarios an event may stay undetected even when it occurs in areas searched by a mobile because it may become active *after* the mobile has searched the area or it may become inactive *before* the mobile searches the area. Our approach can address such scenarios using the forgetting factor parameter f in eq. (9.2).

We consider 100 sensor fields with 200 randomly distributed stationary sensors, each sensor field is deployed in a $200m \times 200m$ area and contains 10 undetected events. We assume that the events are activated according to a Poisson process with rate $\lambda = 1/300$ (i.e. 300 time steps is the expectation of inter-arrival time of each event) and they are uniformly distributed in the areas not covered by the static sensor nodes. Each event remains active for a time interval that it exponentially distributed with rate $\mu = 1/100$ (i.e. 100 time steps is the expectation of lifetime of each event).

In the following simulation experiment we investigate the performance of our approach in terms of the average number of detected events as a function of the forgetting factor f values and also compare our approach with random search using Monte Carlo simulation. In the random search, the next position of a mobile node is $\rho = 1m$ away from the previous one and at random heading direction $0 \leq \theta \leq 2\pi$. The results are depicted in Fig. 9.15 where a mobile node has moved for 3000 moving steps. Instead of fixed values one can use a function of time k for the forgetting factor, however this will require extensive memory and computation (each element of matrix P_k must have a time-stamp).



(a) Number of Events found (b) forgetting factor f profiles.

Figure 9.15: Average number of detected events as a function of the forgetting factor f values when a single mobile is moving for 3000 moving steps with r_z=25m.

Under more extensive simulations that we are not tabulated due to space limitations, we reach the following conclusions regarding the forgetting factor f: The forgetting factor f and the searching neighborhood r_z are two parameters that are affect each other. For instance if $f \rightarrow 1$ and r_z is too small to catch this degradation in the mobile's P_k the mobile seems to be trapped after searching once the entire field. Such trapped behavior is also experience when the coverage cost function J_c is used because sometimes it blocks the mobile to run over a recently searched path when the mobile tries to go towards areas not search for long time. When multiple mobiles are used this problematic behavior becomes more significant.

Clearly, when the environment is dynamic the mobiles have to continuously search the environment for events, given more moving steps or using more mobile nodes more events can be found. The forgetting factor f must be set in conjunction to r_z range such that the mobile could identify which regions have not been searched recently. Given that the r_z is big enough to catch the degradation in the mobile's P_k, the forgetting factor must set accordingly such that the mobile forgets "smoothly" the regions already searched. If $f = 1$ the mobile will avoid search again the field and thus once the field is searched it will trapped or move randomly in some sense. If the mobile forgets rapidly i.e. after a few moving steps the mobile can not explore the entire field instead is searching the field locally. We should mention here that the algorithm outperforms random search in the number of undetected events found in such dynamic environments. To conclude with, we mention that depending on the events characteristics the mobiles should forget "smoothly" such that the entire field can be search. If the number of mobiles is big enough such that they can communicate among them in a multi-hop manner and thus having almost the same P_k our approach can address such dynamic environment successfully.

Finally, the last simulation experiment evaluates the robustness of the distributed path planning algorithm with respect to the positioning uncertainty on the mobile and static sensor nodes. That is, when a mobile node requests the positions of its neighboring static and mobile sensors, the positions received are not the accurate positions but approximate due to inaccuracies in the localization system of sensor nodes. In particular, if the sensor node (mobile or static) accurate position is $\mathbf{x}(k)$, its approximate position is given by $\hat{\mathbf{x}}(k) = \mathbf{x}(k) + \boldsymbol{\delta}$ where $\boldsymbol{\delta}$ is a random vector that represents the effects of positioning uncertainty (noise). We assume $\boldsymbol{\delta}$ is additive white Gaussian noise with zero mean and variance σ^2.

Fig. 9.16 presents the average area coverage achieved by the distributed path planning algorithm for different σ values after 300 time steps using 5 mobile nodes with $r_d = 5m$ and $\rho = 2.5m$. As indicated in simulation experiments of Fig. 9.16, small

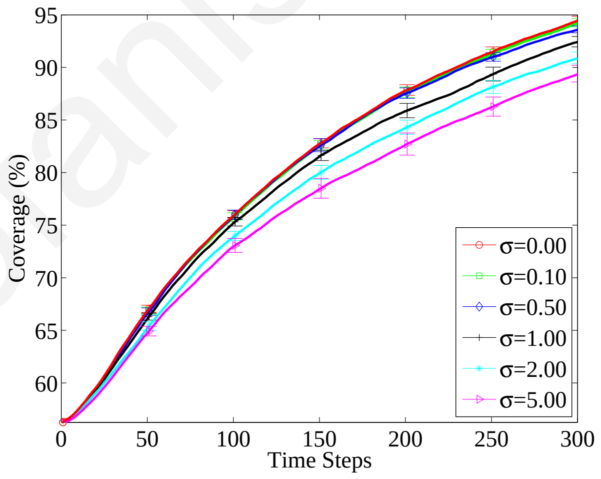


Figure 9.16: The average coverage accomplished over 100 sensor fields with 300 randomly distributed stationary sensors by $M = 5$ mobile nodes after 300 moving steps for different Gaussian noise standard deviation σ values (positioning uncertainties)

positioning uncertainties do not significantly affect the area coverage performance of the distributed path planning algorithm, however as the Gaussian noise standard deviation σ is becoming bigger and comparable to ρ (the distance that each mobile covers in one time step) or even r_d, the area coverage performance is significantly affected.

9.7 Conclusion

The objective of this work is to provide an analysis for deciding the optimum searching strategy that minimizes the expected detection time of an event in the context of mixed sensor networks. We have prove analytically that searching for an event “locally” under certain circumstances can lead to fast event detection. Finding the optimal search path strategy is proved to be an *NP*-complete problem and therefore we proposed a heuristic centralized path planning strategy that provides computationally tractable and close to optimal solution to the problem. The heuristic centralized approach has been also compared with the proposed distributed collaborative path planning approach which remains valid for the general case where multiple mobile nodes are searching for an event in a sensor field area and the shape as well as the number and distribution of uncovered regions are govern by randomness. Extensive simulation results indicate that the selection of the searching neighborhood radius r_z is of critical importance. Therefore, in the next chapter we further investigate the searching neighborhood r_z radius and in particular to study how this parameter can be found or approximated analytically in conjunction with other parameters used in the path planning algorithm as well as the stationary field random deployment parameters.

Chapter 10

On the Optimal Search Neighborhood in Mixed Wireless Sensor Networks

10.1 Summary

This chapter considers the problem of finding the path that improves the area coverage over time in a sparse stationary sensor network with mobile sensor nodes. For a simplified scenario, the chapter derives the optimal path strategy and extrapolates some of the properties of the scenario to a more general instance of the problem. Furthermore, the chapter proposes a surrogate metric that can be used in order to determine the optimal searching neighborhood and presents extensive simulation results which indicated that this approach can achieve very good results.

10.2 Introduction

In the previous chapter we study the optimum searching path for one mobile node that minimizes the expected event detection time for a simplified scenario of the problem and study the event detection capabilities of the proposed path planning algorithm for the general problem. Obtained results indicated that the selection of the searching neighborhood radius is a crucial issue. In this chapter we further investigate the searching neighborhood r_z radius and we propose a surrogate metric that can be used in order to determine the optimal size of the search neighborhood as a function of other parameters used in the proposed path planning algorithm as well as the stationary field random deployment parameters. The main objective is to determine the optimal path that the mobile node (or a group of nodes) should follow in order to that maximize the area coverage over time. As it will be shown in sequel, this objective is identical to the objective of minimizing the expected event detection delay.

In principle, assuming there are h coverage holes, at known locations, one could determine the optimal path of the mobile. However, this problem is *NP* complete since it can be reduced to the Traveling Salesman Problem (TSP or the multi-salesmen problem if there is a group of mobiles). Therefore, finding an optimal solution for medium or large size problems is not computationally feasible. Furthermore, in general sensor networks are deployed in harsh environments thus the exact state of the network (operating condition of each node) is not known a priori. Thus, one of the benefits of deploying mobiles nodes is that the state of the sensors can also be investigated.

Our approach in addressing this problem is to use a dynamic search strategy where the mobile determines the biggest coverage hole in an area (neighborhood) around it which constitutes its target location (i.e., the area that needs to be sampled next) [136, 142]. An interesting question that needs to be addressed is the size of the neighborhood that the mobile needs to consider when determining its target. Clearly, that neighborhood cannot be very small since this will lead to myopic strategies where the mobile will search for very small holes ignoring much bigger holes that are a little further away. On the other hand, what we show in this chapter is that the neighborhood should not be very big either which is a rather counter-intuitive result. This result indicates that the mobile should look for a "medium" size coverage hole located in the mobile's immediate neighborhood and ignore the possibly larger holes that are located further away. This strategy is justified because the mobile will waste valuable time traveling towards a bigger hole when it can sample the smaller coverage holes that are located much closer to it.

As indicated above, the searched neighborhood should not be very small but it should not be very big either, therefore, there must be an optimal size. Formulating the problem to determine the optimal neighborhood size is not straightforward[1], thus we resort to a surrogate metric that can lead us to the optimal neighborhood size. The surrogate metric used is the *variance of vacancy* used in coverage processes [30, 143]. In this context, the computation of the variance of vacancy is a function of the size of the area that is used for the computation. The main idea in this chapter is to associate the neighborhood where a mobile is going to search for the biggest coverage hole with the area that maximizes the variance of vacancy. The justification behind this approach is that the mobile needs to consider as much new information as possible when it will decide where it will go next. Thus, the mobile uses the neighborhood that maximizes variance of vacancy. As will be presented in the sequel, this choice achieves very good results.

[1]Potentially one could run multiple simulations off-line, under different neighborhood sizes and pick the one with the best coverage but this is excessively time consuming and does not provide any important insights about the nature of the problem.

The contributions of this chapter are the following. In the context of mixed wireless sensor networks, it shows that it is not optimal to first search the largest coverage hole in the entire field; rather searching a big enough hole close to the current mobile location can yield faster coverage. Furthermore, the chapter proposes a surrogate metric that can be used in order to determine the optimal size of the search neighborhood. Even though the proposed search approach cannot guarantee an optimal solution, the obtained solutions are satisfactory considering that the original problem is *NP*-complete.

The remaining of this chapter is organized as follows. Section 10.3 presents the modeling assumptions as well as the required definitions. Section 10.4 determines the optimal path of a mobile when only two coverage holes exist and shows that it is optimal to search smaller holes located closer to the mobile rather than bigger holes far away. Section 10.5 introduces some basic results relating to the coverage processes and presents the surrogate metric used in this chapter. Section 10.6 present some simulation results and finally the chapter ends with the Conclusions.

10.3 Sensor Network Model and Objectives

In this chapter, we adopt the mixed sensor network framework proposed in the previous chapter and we also use exactly the same notation, modeling assumptions and information structure needed by the mobile nodes to run the path-planning algorithm as in the sections 9.5 and 9.3 of previous chapter.

Consider a mixed wireless sensor network that consists of large set of static sensor nodes and a small set of mobile nodes deployed in a large square area $\mathcal{A}$. Let the set $\mathcal{N}$ to define the set of all sensor nodes in the sensor network and let all nodes have a common detection range r_d where, if an event occurs it will be detected with probability one.

Next, we define the *coverage over time* $\mathcal{C}$ which will serve as an objective function to be maximized by the mobile sensors. At any instant t, let $I(\mathbf{x}, t)$ be an indicator function that takes the value 1 if point $\mathbf{x} \in \mathcal{A}$ is covered once by at least one sensor during the time interval $[0, t)$ and 0 otherwise. In other words, if we define $\mathbf{x}_i + U_i$, to be the set of points covered by sensor $\mathbf{x}_i$ such that $\{\mathbf{x}_i + \mathbf{x} : \mathbf{x} \in U_i\}$ [2] then $I(\mathbf{x}) = 1$ if $\mathbf{x} \in \bigcup_{i=1}^{N} \{\mathbf{x}_i + U_i\}$. Thus, the coverage achieved by the network at t is given by

$$C(t) = \frac{1}{A} \int_{\mathcal{A}} I(\mathbf{x}, t) d\mathbf{x}.$$

[2] U_i is the area covered (shape) by sensor $\mathbf{x}_i$, $i \in \mathcal{N}$, over the time interval $[0, t)$. e.g. it defines a disk of radius r_d for a static sensor or defines the area covered by the track of width r_d of a mobile sensor.

The coverage $C(t)$ defines the ratio of covered area by the mixed sensor network to the area of interest in the time interval $[0,t)$. In other words, it defines the probability $P(t)$ that a static event will be detected by at least one sensor node somewhere in the time interval $[0,t)$, where $t \leq T$ and T defines the time that is needed by mobile nodes to achieve full coverage of the uncovered regions.

As mobile nodes move, they cover new areas, thus the coverage over time is a reasonable objective function that needs to be maximized by the mobiles.

$$\mathcal{C}(T) = \int_0^T C(t)dt \tag{10.1}$$

Now, consider the expected time $\mathbb{E}[\tau]$ of initial detection denoted to be the time at which the stationary event falls in the sensing area of a mobile sensor for first time, where τ is a random variable that denotes the time that the event is first detected. The probability that a stationary event remains undetected in the time interval $[0,t)$ can be given by

$$P(\tau \geq t) = 1 - P(\tau < t) = 1 - C(t) \tag{10.2}$$

Consequently, the expectation of first detection time $\mathbb{E}[\tau]$ is given by

$$\mathbb{E}[\tau] = \int_0^T P(\tau \geq t)dt = \int_0^T (1 - C(t))\, dt \tag{10.3}$$

where T is the time that is needed for full coverage, i.e.$C(t \geq T) = 1$.

Therefore, by comparing eq.(10.1) and eq.(10.3) we get

$$\mathcal{C}(T) = 1 - \mathbb{E}[\tau]$$

Consequently, one can come to the conclusion that $\mathcal{C}(T)$ and $\mathbb{E}[\tau]$ are two complementary objectives. Hence, a proper motion planning strategy should either maximize $\mathcal{C}(T)$ or minimize $\mathbb{E}[\tau]$.

10.4 The Two Hole Problem

As already mentioned, the objective of the mobile nodes is to sample all coverage holes such that $\mathcal{C}(T)$ is maximized (for some T). Assuming there are h coverage holes, then the problem of determining the optimal path of the mobile is NP-complete (it can be reduced to the TSP). In this section we investigate what happens if there are only two coverage holes in order to gain some insight that can be used in other heuristic approaches for efficiently solving the problem.

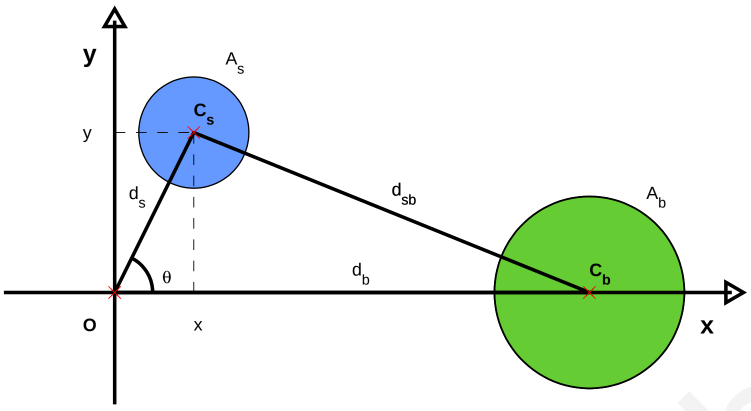


Figure 10.1: Problem geometry

Assume that the field has only two coverage holes with areas A_b and A_s ($A_b \geq A_s$) and centroids, $\mathbf{C_b}$ and $\mathbf{C_s}$ respectively (see Fig. 10.1). For simplicity, it is also assumed that there is no overlap between the two holes. A mobile node is initially placed at position $\mathbf{O}$ at distance $d_b = \|\mathbf{C_b} - \mathbf{O}\|$ from hole A_b and at distance $d_s = \|\mathbf{C_s} - \mathbf{O}\|$ from hole A_s. The distance between the two coverage holes is indicated by $d_{sb} = \|\mathbf{C_s} - \mathbf{C_b}\|$. The objective of the mobile is to maximize $\mathcal{C}(T)$ in eq.(10.1) where T is some time instant such that in all of the paths considered, the mobiles achieve full coverage.

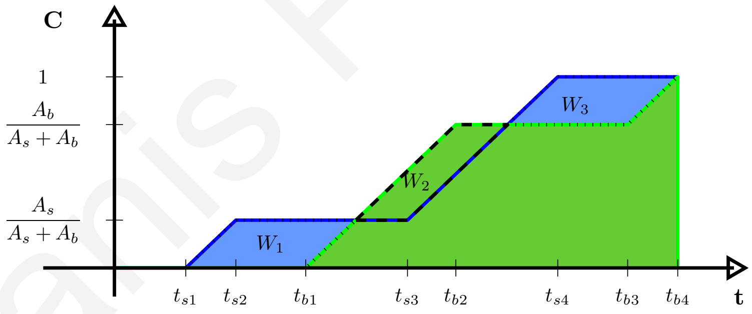


Figure 10.2: Coverage over time

Given that there are only two holes, the mobile has only two options[3]. First go to A_b, search A_b and then go to A_s or first go to A_s, search A_s and then go to A_b. Fig. 10.2 shows $C(t)$ under the two different paths thus $\mathcal{C}(T)$ for each path is the area under the corresponding curve from 0 until $T \geq t_{b4}$. In this figure, we assume that when the mobile travels over covered regions $\dot{C}(t) = 0$ while when it searches in coverage holes the coverage improvement is constant at rate $\dot{C}(t) = 2r_d v/(A_s + A_b)$ where v is the constant mobile speed[4].

[3]We do not consider paths where the mobile can wander around in covered regions.

[4]Fig. 10.2 implies that the mobile does not start sampling until it reaches the centroid of the hole and that the shape and size of the hole are such that constant $\dot{C}(t)$ is always applicable, however, we point out that these simplifications do not significantly affect the final result.

When the mobile follows the path from $\mathbf{O} \to \mathbf{C_s} \to \mathbf{C_b}$, $\mathcal{C}_{sb}(T)$ is given by

$$\begin{aligned}\mathcal{C}_{sb}(T) \;=\;& \frac{1}{2}\frac{A_s}{2r_d v}\frac{A_s}{A_s+A_b}+\frac{d_{sb}}{v}\frac{A_s}{A_s+A_b}+\\ & \frac{A_b}{2r_d v}\frac{A_s}{A_s+A_b}+\frac{1}{2}\frac{A_b}{2r_d v}\frac{A_b}{A_s+A_b}+\frac{d_b-d_s}{v}\end{aligned} \tag{10.4}$$

Similarly, if the mobile follows the path from $\mathbf{O} \to \mathbf{C_b} \to \mathbf{C_s}$, $\mathcal{C}_{bs}(T)$ is given by

$$\begin{aligned}\mathcal{C}_{bs}(T) \;=\;& \frac{1}{2}\frac{A_b}{2r_d v}\frac{A_b}{A_s+A_b}+\frac{d_{sb}}{v}\frac{A_b}{A_s+A_b}+\\ & \frac{A_s}{2r_d v}\frac{A_b}{A_s+A_b}+\frac{1}{2}\frac{A_s}{2r_d v}\frac{A_s}{A_s+A_b}\end{aligned} \tag{10.5}$$

Comparing (10.4) and (10.5) or simply observing Fig. 10.2, the decision of the mobile is to follow the path that maximizes $\mathcal{C}(T)$ which is equivalent to comparing the three areas W_1, W_2 and W_3 in Fig. 10.2. Thus

$$\mathcal{C}_{sb}(T) \lesseqgtr \mathcal{C}_{bs}(T) \Leftrightarrow W_1 + W_3 \lesseqgtr W_2 \tag{10.6}$$

which in turn, after some algebraic manipulations, is equivalent to

$$\mathcal{C}_{sb}(T) \lesseqgtr \mathcal{C}_{bs}(T) \Leftrightarrow d_{sb}\left(A_b - A_s\right) \gtrless \left(d_b - d_s\right)\left(A_b + A_s\right) \tag{10.7}$$

This result is complementary to result presented in eq.(9.7) of the previous chapter where we consider the path that minimizes the expected event detection time. (see also the previous section).

Next, we consider the following cases:

C1 $\{A_b = A_s = A\}$: The decision problem $\mathcal{C}_{sb}(T) \lesseqgtr \mathcal{C}_{bs}(T)$ reduces to $d_s \gtrless d_b$, i.e., the mobile should go to its nearest coverage hole first. The proof follows easily by substituting $A_b = A_s = A$ in (10.7).

C2 $\{d_b = d_s = d\}$: The decision problem $\mathcal{C}_{sb}(T) \lesseqgtr \mathcal{C}_{bs}(T)$ reduces to $A_b \gtrless A_s$, i.e., the mobile should go to the biggest hole A_b first (since by assumption $A_b \geq A_s$). The proof again follows easily by substituting $d_b = d_s = d$ in (10.7).

C3 $\{A_b > A_s \textit{ and } d_b < d_s\}$: The decision is to always go to the biggest hole which is also located nearer to the mobile. The proof follows by comparing the terms of (10.7).

C4 $\{A_b > A_s \textit{ and } d_b > d_s\}$: The decision depends on the relative position (d_{sb}) and area ratio ($\varrho = A_b/A_s$) of the bigger hole with respect to the smaller one. Specifically, if the smaller hole is located inside an "egg shaped" area then the decision is to search the smaller hole first, otherwise, it is better to search the larger hole first. The proof follows by solving (10.8) defined by the cosines rule of the triangle in Fig. 10.1. Using the cosines rule we know that

$$ {d_{sb}}^2 = {d_s}^2 + {d_b}^2 - 2 d_s d_b \cos(\theta). \tag{10.8} $$

Also, using some algebra, one can rewrite (10.7) such that the mobile should visit the smaller hole first. Thus, the mobile should first visit the smallest hole if

$$ d_{sb} \leq \frac{\varrho + 1}{\varrho - 1}(d_b - d_s) \tag{10.9} $$

where $\varrho = A_b/A_s > 1$. Substituting (10.9) in (10.8), we get a single equation (10.10) with one unknown, d_s which denotes the decision boundary that determines which hole will be visited first.

$$ \left(\frac{\varrho + 1}{\varrho - 1}(d_b - d_s)\right)^2 = {d_s}^2 + {d_b}^2 - 2 d_s d_b \cos(\theta) \tag{10.10} $$

Therefore, the mobile should search the smaller hole first if its centroid is located within the egg-shaped region defined by the solution of (10.10). In polar coordinates the solution of (10.10) is eq. (10.11) where $r = d_s$

$$ \begin{aligned} r &= \frac{d_b\left((\varrho+1)^2-(\varrho-1)^2 cos(\theta)-\sqrt{\left((\varrho+1)^2-(\varrho-1)^2 cos(\theta)\right)^2-(4\varrho)^2}\right)}{4\varrho} \\ \theta &= [0, 2\pi) \end{aligned} \tag{10.11} $$

This holds true when $\mathbf{O} = (\mathbf{0}, \mathbf{0})$. One can use (10.11) to draw the region in polar coordinate system. This egg-shaped region is illustrated in Fig. 10.3 and indicates that if the centroid of the small hole is located inside this shape the coverage improvement rate is maximized when following the $\mathbf{O} \rightarrow \mathbf{C_s} \rightarrow \mathbf{C_b}$ path.

Concluding, the analysis above demonstrates that a mobile should not go immediately to the largest hole in the field but it should first search smaller holes that are closer to the mobile (areas in the egg shaped region). However, note that the precise size of the egg, depends on the relative size of the two coverage holes (ϱ). If for example the smaller hole is significantly smaller than the larger one, then the egg will be significantly narrower, implying that the smaller one should be visited first only if it is exactly in the straight path to the big hole. Furthermore, in many scenarios it may be difficult to clearly identify two holes (some holes may be connected) and as already mentioned there may be more than two holes which makes it impractical to determine the optimal path of the mobile. Thus, the implementation of such an algorithm is rather difficult, however, the insight from the analysis is clear: "Large enough holes close to the mobile should be searched first, before moving towards the biggest holes of the field". The simplest way to implement this "insight" is by searching for the biggest coverage hole in a neighborhood around the mobile (the proposed path planning algorithm presented in the previous chapter 9). If this neighborhood is too small, then the mobile may waste time searching insignificant holes missing much larger holes. On the

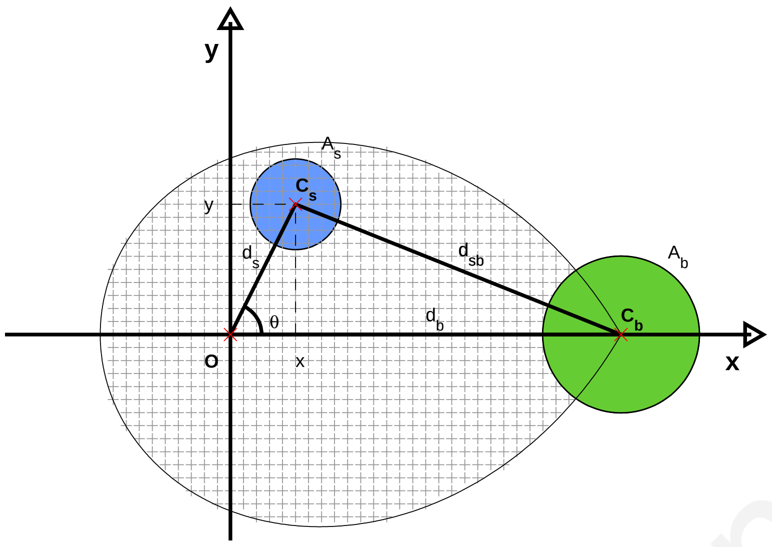


Figure 10.3: The Egg-shaped region for $\varrho = 3$ and $\mathbf{O} = (\mathbf{0}, \mathbf{0})$. If $\mathbf{C_s}$ is located inside the shaded region then a mobile should follow the path $\mathbf{O} \rightarrow \mathbf{C_s} \rightarrow \mathbf{C_b}$ to maximize coverage over time.

other hand, if the neighborhood is too big, then the mobile will move straight towards much larger holes avoiding significant holes that are located close to it. Therefore, there is an optimal neighborhood size. In the next section we investigate a surrogate function that we can use to perform this optimization.

10.5 Vacancy

In this section we use the tools from coverage processes [30, 143] in order to analyse the coverage holes that are generated from the random deployment of sensors in $\mathcal{A}$. Consider a two-dimensional point process where a collection of N random points is thrown in a square area $\mathcal{A}$ according to the probability density $f(\mathbf{x}) = \frac{1}{A}$. Let the countable collection of randomly distributed points be $\mathcal{P} \equiv \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. Assume that there exists a disc around each point of radius r (in our case $r = r_d$ the detection range) thus all points in the union of all N discs are considered as covered while all non-covered points are considered as vacant. Vacancy is the collection of all vacant points within an arbitrary area $\mathcal{R} \subset \mathcal{A}$ which constitutes a random variable with mean and variance that are defined in the sequel [144]. Let $\{U_1, U_2, ...U_N\}$ be a countable collection of non-empty sets (the coverage area each each sensor). For the purposes of this chapter, U_i is a disc of radius r_d and area $a = \pi r_d^2$. Below we summarize some of the results presented in [144]. Let $\overline{I(\mathbf{x})}$ be the indicator function of uncovered points such that $\overline{I(\mathbf{x})} = 1 - I(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathcal{A}$ is not covered by any disk of radius r_d or $\overline{I(\mathbf{x})} = 0$ otherwise. The vacancy within and arbitrary area $\mathcal{R} \subset \mathcal{A}$, $V_{\mathcal{R}} = V(\mathcal{R})$ is given by

$$V_{\mathcal{R}} = V(\mathcal{R}) \equiv \int_{\mathcal{R}} \overline{I(\mathbf{x})} d\mathbf{x} \tag{10.12}$$

and the mean of vacancy (expected uncovered area) is

$$\begin{aligned} E(V_{\mathcal{R}}) &= \int_{\mathcal{R}} E\{\overline{I(\mathbf{x})}\} d\mathbf{x} = \int_{\mathcal{R}} P(\mathbf{x} \textit{ not covered}) \, d\mathbf{x} \\ &= \int_{\mathcal{R}} \Big(1 - \frac{a}{A}\Big)^N d\mathbf{x} = R\Big(1 - \frac{a}{A}\Big)^N \end{aligned} \tag{10.13}$$

where $p = \frac{a}{A}$ is the probability that a point $\mathbf{x} \in \mathcal{A}$ is covered by a disk of area $a = \pi r_d^2$ and $(1-p)^N$ is the probability that the point $\mathbf{x}$ is not covered by any of the N disks (sensor positions are independent). Also R is the area of $\mathcal{R}$.

The variance of vacancy is

$$Var(V_{\mathcal{R}}) = E\left(V_{\mathcal{R}}^2\right) - \left(E(V_{\mathcal{R}})\right)^2 \tag{10.14}$$

where the mean square of vacancy is

$$\begin{aligned} E(V_{\mathcal{R}}^2) &= \int\int_{\mathcal{R}^2} E\{\overline{I(\mathbf{x})I(\mathbf{y})}\} d\mathbf{x} d\mathbf{y} \\ &= \int\int_{\mathcal{R}^2} P(\mathbf{x}, \mathbf{y} \textit{ both not covered}) d\mathbf{x} d\mathbf{y} \end{aligned} \tag{10.15}$$

Thus, $Var(V_{\mathcal{R}})$ can be computed by numerically performing an integration of the probability $P(\mathbf{x}, \mathbf{y}$ *both not covered*). This technique is explicated in [30, 145].

Let the density $\lambda \equiv \frac{N}{A}$ of points per unit area of $\mathcal{A}$ converges to a constant value as $\mathcal{A}$ increases. Hence, for N large and $\frac{a}{A}$ small

$$\Big(1 - \frac{a}{A}\Big)^N \approx \left(1 - \frac{\lambda a}{N}\right)^N \approx e^{-\lambda a}$$

Thus by (10.13) the mean of vacancy in a region $\mathcal{R} \subseteq \mathcal{A}$ is

$$E(V_{\mathcal{R}}) \approx R e^{-\lambda a} \tag{10.16}$$

An approximation of the variance of vacancy in a subregion $\mathcal{R} \subseteq \mathcal{A}$ with area R is derived in [144] and is given by

$$Var(V_{\mathcal{R}}) \approx R a e^{-2\lambda a} \bigg(8 \int_0^1 x \Big(e^{\lambda \frac{a}{\pi} \mathrm{B}(x,1)} - 1\Big) dx - R a \lambda^2\bigg) \tag{10.17}$$

where $\mathrm{B}(x, r)$ is the intersection area of two disks with radius r and which are centered $2x$ apart. This area is given by

$$\mathrm{B}(x, r) = \begin{cases} 4r^2 \int_{x/r}^1 \sqrt{1 - y^2} dy & \text{if } 0 \le x \le r \\ 0 & \text{if } x > r \end{cases} \tag{10.18}$$

Hence $\mathrm{B}(x, 1) = 2 \arccos(x) - 2x\sqrt{1 - x^2}$. Even though (10.17) cannot be computed analytically, it can be computed numerically[5].

Let

$$Q(\lambda, r_d) = \int_0^1 x \left(e^{\lambda r_d^2 \mathrm{B}(x,1)} - 1 \right) dx \tag{10.19}$$

independent of R, then the $Var(V_{\mathcal{R}})$ can be written as

$$Var(V_{\mathcal{R}}) \approx R\pi r_d^2 e^{-2\pi r_d^2 \lambda} \left(8Q(\lambda, r_d) - R\lambda^2 \pi r_d^2 \right) \tag{10.20}$$

which is a second order polynomial in R with a maximum at

$$R^* = \frac{4Q(\lambda, r_d)}{\pi \lambda^2 r_d^2} \tag{10.21}$$

10.5.1 An Approximation of Optimal Search Neighborhood

Next, we use the optimal area size R^* in order to determine the optimal neighborhood size r_z that the mobile node should use in order to determine the biggest coverage hole to visit next. Recall that the conjecture is that the neighborhood size should be large enough such that the new information considered by the mobile in making this decision is maximized. Assuming that at time t the mobile is at position $\mathbf{x}(t)$, then it should search for the biggest hole in a a circular area $\mathcal{R}_1$ with radius r_z. During the next step, the mobile will move to a new location $\mathbf{x}(t+1) = \mathbf{x}(t) + \boldsymbol{\rho}$, $\boldsymbol{\rho} \in \mathbb{R}^2$, where the region that the mobile will search for a coverage hole will be $\mathcal{R}_2$. Thus, the *new* information that the mobile will consider from one step to the next is $\Delta\mathcal{R} = \mathcal{R}_2 \setminus \mathcal{R}_1$ (i.e $\mathcal{R}_1^c \cap \mathcal{R}_2$). The objective then is to choose the size of the areas $\mathcal{R}_1$ and $\mathcal{R}_2$ (the radius r_z [137, 146]) such that the variance of vacancy in $\Delta\mathcal{R}$ is maximized. As the variance of vacancy in $\Delta\mathcal{R}$ is maximized between two consecutive steps, the mobile can exploit, on average, the "maximum" difference in vacancy at each step t in order to take, on average, the optimal local decision. Given the result of (10.21), the optimal radius r_z^* is the solution to the equation

$$\Delta R = \frac{4Q(\lambda, r_d)}{\pi \lambda^2 r_d^2} \tag{10.22}$$

where ΔR is the area of $\Delta\mathcal{R}$.

Lemma 10.5.1. *The solution to (10.22) is approximated by*

$$r_z^* \approx \frac{64Q^2(\lambda, r_d) + \pi^2(\rho\lambda r_d)^4}{32\pi\rho\lambda^2 r_d^2 Q(\lambda, r_d)} \tag{10.23}$$

[5]Note that in order to avoid the edge effects, the above results assume that the square region $\mathcal{A}$ is a quadratic torus, i.e., when a disk protrudes out of one side of the region it re-enters from the opposite side. Also, note that the approximation in (10.17) holds true under the assumption that aN converges to a constant value α ($0 < \alpha < \infty$) as $N \to \infty$ and $a \to 0$. The proofs are provided in [144] (see Case B)

where $\rho = \|\boldsymbol{\rho}\|$ is the distance traveled by the mobile in one step.

Proof. Assuming the mobile sensor has moved a distance ρ, the area of $\Delta\mathcal{R}$ is given by

$$\begin{aligned}\Delta R &= \pi {r_z}^2 - \mathrm{B}(\rho/2, r_z)\\ &= \pi r_z^2 - 2r_z^2 \arccos(\tfrac{\rho}{2r_z}) + \tfrac{\rho}{2}\sqrt{4r_z^2-\rho^2}\end{aligned}$$

Thus, substituting in (10.22), r_z^* is the solution of the

$$\pi r_z^2 - 2r_z^2 \arccos(\frac{\rho}{2r_z}) + \frac{\rho}{2}\sqrt{4r_z^2-\rho^2} = \frac{4Q(\lambda, r_d)}{\pi\lambda^2 r_d^2}$$

This equation is difficult to be solved due to the arccos term. Using a Taylor series expansion, one can approximate

$$\arccos(\frac{\rho}{2r_z}) \approx \frac{\pi}{2} - \frac{\rho}{2r_z}$$

Therefore, ΔR is approximated by

$$\Delta R \approx \frac{\rho}{2}\left(2r_z + \sqrt{4r_z^2-\rho^2}\right)$$

which is substituted in (10.22) and as a result, r_z^* is the solution to

$$\frac{\rho}{2}\left(2r_z + \sqrt{4r_z^2-\rho^2}\right) - \frac{4Q(\lambda, r_d)}{\pi\lambda^2 r_d^2} = 0$$

which, after some algebraic manipulations reduces to the lemma result. □

10.6 Simulation Results

In this section we present some numerical results that support the conjecture of this chapter, i.e., that the optimal search area is given by Lemma 10.5.1. The precise algorithm used for navigation by the mobile is presented in [136, 137] with the following parameters: The mobile evaluates $\nu = 10$ candidate next positions which are uniformly distributed on an arc with radius $\rho = 2.5m$ and extends $\phi = 35°$ above and below of the current direction of the mobile. Unless otherwise stated, all experiments refer to a square sensor field of area $A = 40000m^2$. A set of $S = 200$ static sensors are deployed and their coordinates are generated according to a uniform distribution. The detection radius of all sensors is $r_d = 5m$ and the communication range $r_c = r_z + r_d$. The radius r_z defines the radius of the search area where the mobile is searching for its target (largest coverage hole center). All simulations performed in MATLAB and the outcomes are the averages of 100 independent random deployments.

In the first simulation experiment we investigate the effect of the sensor detection range r_d on the optimal neighborhood size. Using Lemma 10.5.1, the optimal neighborhood size for different r_d is presented in Table 10.1. As shown in Table 10.1 as

r_d	S	ρ	$r_z{}^*$	$Var(V_{\Delta\mathcal{R}})$
2	200	2.5	20.3	35.9
5	200	2.5	21.9	847
8	200	2.5	25.7	2232.1
10	200	2.5	30.1	2417.6

Table 10.1: The optimal search neighborhood r_z^* for different r_d values

the detection radius r_d increases, the r_z^* radius, where $Var(V_{\Delta\mathcal{R}})$ is maximized, also increases but remains small compared to the field size (e.g. 200m). This is reasonable because as the sensing radius of each sensor increases (and given that the number of sensors is fixed $S = 200$) it is possible to generate deployments where with higher variation in the achieved coverage.

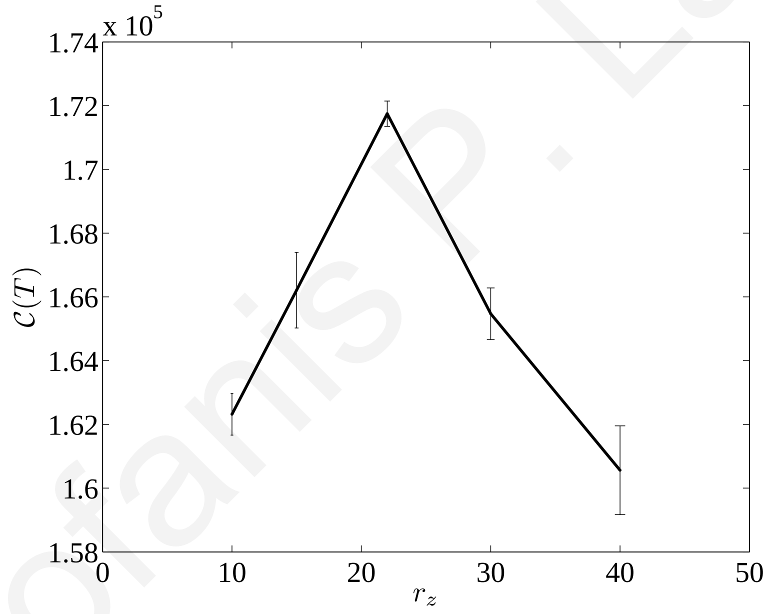


Figure 10.4: The average dynamic coverage accomplished over 100 sensor fields by a mobile node after 2000 moving steps for different r_z values when $r_d = 5m$

Fig. 10.4 presents the average dynamic coverage $\mathcal{C}(k)$ achieved by the path planning algorithms presented in [136, 146] after $k = 2000$ time steps accomplished over 100 sensor fields by one mobile node when $r_d = 5m$. The figure indicates that coverage is maximized when $r_z = 22m$ which is what was also predicted by Lemma 10.5.1 (see Table 10.1).

In the next simulation experiment we investigate how the optimal r_z value is affected by the density $\lambda \equiv \frac{S}{A}$ of the static sensors. First, using Lemma 10.5.1 we compute the optimal r_z^* as shown in 10.2.

r_d	S	ρ	${r_z}^*$	$Var(V_{\Delta\mathcal{R}})$
5	100	2.5	41.9	1141.3
5	200	2.5	21.9	847
5	300	2.5	15.4	630.4
5	400	2.5	12.1	470.7

Table 10.2: The optimal search neighborhood r_z^* for different S values

Fig. 10.4 shows that for $S = 200$ the optimal $r_z^* = 22m$ which is agreement with the results of Table 10.2. Furthermore, Fig. 10.5 presents the coverage achieved by the path planning algorithm when $S = 300$ sensors are deployed. The maximum coverage is achieved when $r_z = 15m$ which is again consistent with the Lemma 10.5.1 prediction as indicated in Table 10.2.

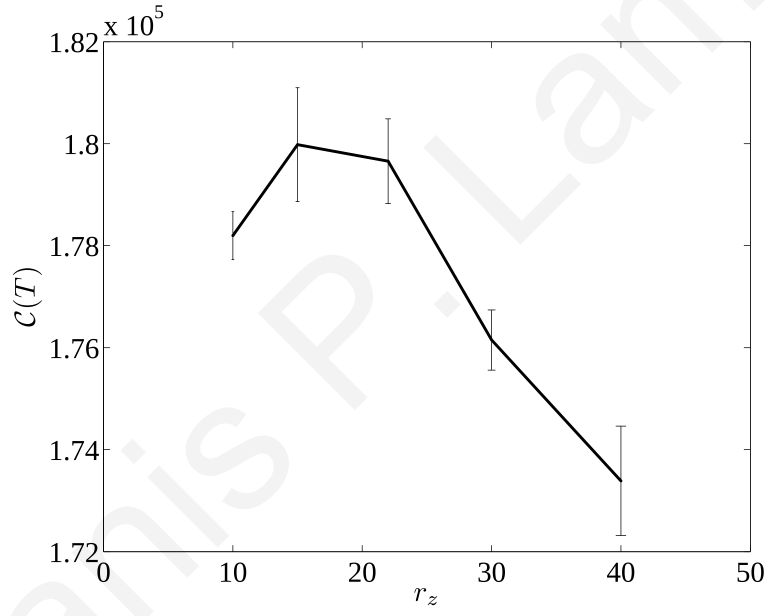


Figure 10.5: The average dynamic coverage accomplished over 100 sensor fields by a mobile node after 2000 moving steps for different r_z values when $S = 300$

Finally the last simulation considers how the optimal r_z value is affected by ρ, the distance that the mobile can move in one time step. Distance ρ also indicates how frequently the target (biggest coverage hole centroid) in the searching neighborhood is computed with respect to the distance moved. Again, we evaluate the optimal radius r_z^* using Lemma 10.5.1 as shown in Table 10.3.

r_d	S	ρ	${r_z}^*$	$Var(V_{\Delta\mathcal{R}})$
5	200	1	54.8	846.99
5	200	2.5	21.9	846.99
5	200	4	13.8	846.98
5	200	5	11.1	846.97

Table 10.3: The optimal search neighborhood r_z^* for different ρ values

Fig. 10.4 indicated that the optimal r_z for $\rho = 2.5m$ is about $22m$ while Fig. 10.6 indicated that for $\rho = 4m$ the optimal r_z is about $15m$. Both of these results are consistent with the Lemma 10.5.1 predictions shown in Table 10.3. Therefore, when the mobile is searching for targets, once it moves farther (bigger ρ) from its previous position the optimal r_z value decreases.

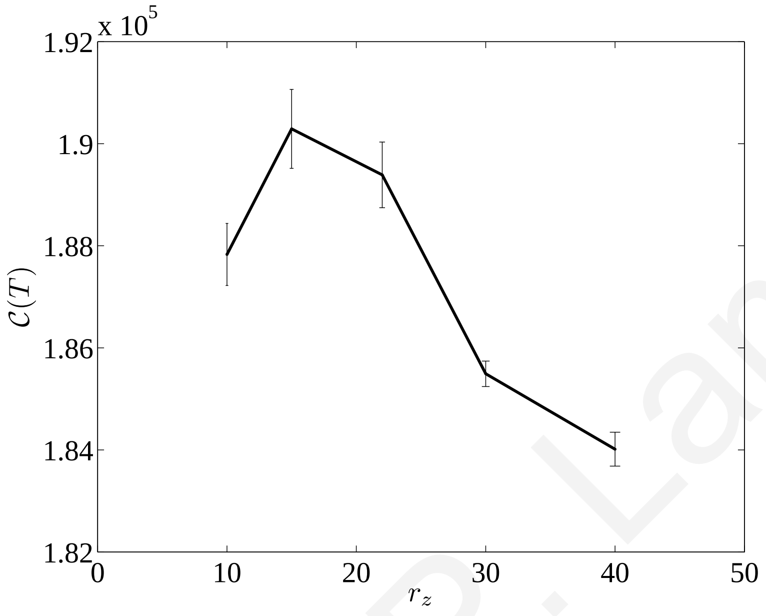


Figure 10.6: The average dynamic coverage accomplished over 100 sensor fields by a mobile node after 2000 moving steps for different r_z values when $\rho = 4m$

Lastly, we have to mention that the approximation method presented for the single mobile node remains valid for the case of multiple mobiles given that the coverage process is mainly governed by the initial distribution of stationary nodes (e.g. when the number of mobiles as well as their coverage rate are small enough). Additionally, this approximation is also valid when the forgetting factor $f < 1$ and hence the mobile's objective is to improve the dynamic coverage rate over time in a small amount of time.

10.7 Conclusion

In this chapter we propose a method to approximate the optimal searching neighborhood that enhances the dynamic coverage performance in a mixed sensor network architecture in conjunction with other parameters used in the path planning method presented in previous chapters. This approximation is based on the variance of vacancy of the binomial coverage process. Obtained results from numerical evaluations of the mathematical approximations have been verified by Monte Carlo simulation outcomes of the dynamic coverage over time performance of the path planning method. In the next chapter, the proposed path planning algorithm has been transferred from simulation to a real-world testbed to provide experimental validations of simulation results and to further investigate the performance of the proposed approach experimentally.

Chapter 11

A Testbed for Coverage Control using Mixed Wireless Sensor Networks

11.1 Summary

This chapter presents the development of an experimental testbed for mixed WSNs consisting of stationary and mobile sensor nodes that collaborate to improve the sensing coverage and event detection of the network in a given deployment area. The chapter describes the hardware and infrastructure of the testbed as well as a case study for coverage control that was investigated using the testbed. We point out that the developed testbed can be used for the evaluation and validation of different algorithms for coverage control that involve collaboration between stationary and mobile sensors to improve the WSN's monitoring capabilities. In addition, it can also be used to investigate other objectives as well as other concepts (e.g., network control).

11.2 Introduction

In the previous chapters we proposed a distributed collaborative framework for complete area coverage where a small set of mobile nodes collaborate with the stationary sensor nodes and with each other in order to improve the area coverage. However the proposed approach has been evaluated only in simulation. In this chapter, the proposed framework has been transferred from simulation to a real-world testbed to further investigate the performance of the proposed path planning algorithm experimentally.

This chapter provides an overview of a mixed WSN experimental testbed. The testbed includes several stationary nodes as well as some nodes mounted on mobile robots. A main objective of the testbed is to demonstrate and validate algorithms that allow the mobile nodes to move autonomously in the area in order to achieve their objectives. For the purposes of this chapter, the objective is to improve the network's sensing coverage. Furthermore, the testbed can be used to demonstrate the collaboration between the sensor nodes. Moreover, we emphasize that the testbed can also be used to demonstrate other objectives (e.g., network connectivity) as well as other concepts such as networked control or control over wireless networks, however, these objectives are out of the scope of this chapter.

The overwhelming majority of approaches for solving the sensing coverage problem has been validated and evaluated based only on simulation models. Experimental work with WSNs and teams of cooperative autonomous robots has been limited, primarily due to cost and practical implementation challenges. Nevertheless, experimental validation is particularly important in distributed WSNs and multi-robot systems research since several factors that can significantly affect the behavior and performance of various approaches may not be accurately modeled. Such factors may include asynchronous communication, delayed or dropped packets, hardware limitations, inaccurate sensing etc. While several experimental testbeds involving either stationary sensors or cooperative autonomous robots have been proposed (e.g. [86, 88, 89, 3, 90, 91, 92, 93, 94, 147]), to the best of our knowledge, non of these specifically addresses the development of mixed WSNs.

The contribution of the chapter is the development of an inexpensive testbed that enables validation and evaluation of mixed WSNs and multi-robot cooperation algorithms through both simulation and experimentation . A case study for improving area coverage in mixed WSNs using the proposed distributed collaborative path planning algorithm is also presented.

The remaining of the chapter is organized as follows. Section 11.3 presents an overview of the developed testbed and describes its main components. Section 11.4 defines the coverage control problem while Section 11.5 presents a case study that is investigated using the testbed and involves the distributed coverage control problem. This section presents all algorithms executed by the mobiles for their collaborative path planning. Section 11.6 presents the results obtained from the testbed under different scenarios. Finally, the chapter concludes with Section 11.7 where some limitations of the testbed as well as plans for future developments are presented.

11.3 Mixed WSN Testbed Overview

This section provides an overview of the implementation of the proposed mixed WSN on real hardware. We describe the various components of the testbed which include the stationary sensors, the mobile sensors, as well as the sensor positioning system which is based on an overhead camera. A schematic overview of the testbed is shown in Fig. 11.1.

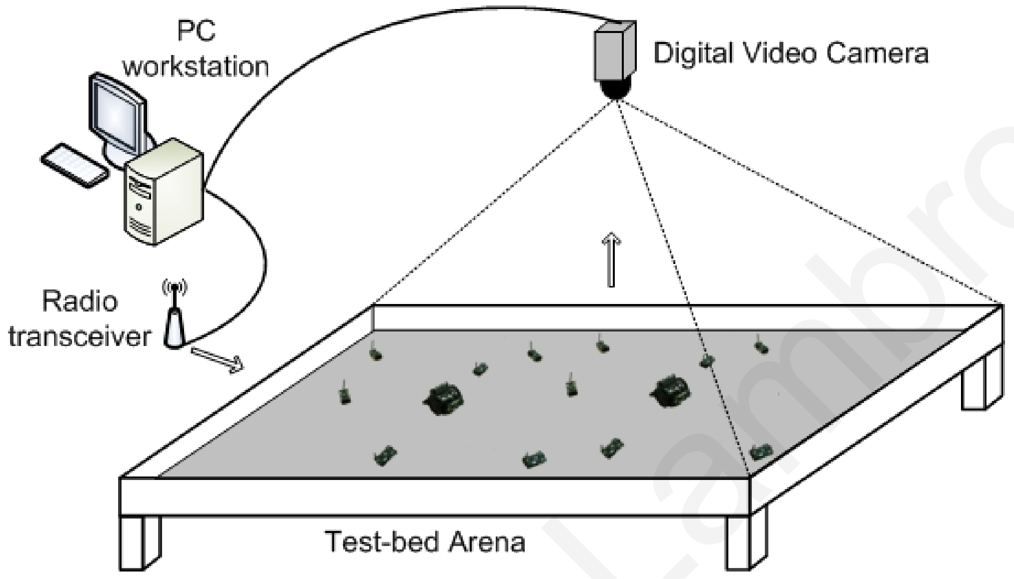


Figure 11.1: Testbed overview

We point out that the testbed has been designed and developed using commercial off-the-shelf components in order to reduce the overall system cost. Furthermore, we emphasize that the coverage control problem is important when the area to be monitored is very large, however, it is not possible to confine such a large network in the space of a research lab, thus, an attempt was made to scale down the detection range r_d and the communication range r_c of each node. For the detection range, we assume that an event source can be detected by a node if it is within a distance r_d from the sensor while in practice we use sources that produce very weak signals. For the communication range, every node drops packets that are received from nodes that are further than r_c as explained in the following subsections.

11.3.1 Stationary Sensor Nodes

MICAz motes [1] are developed by Crossbow Technology and are equipped with an ATMEL ATmega128L [148] processor operating at 8 MHz. The ATmega128 is a low-power 8-bit Microcontroller with integrated Flash, EEPROM, and SRAM memory, it has several internal timers and supports a variety of serial interfaces, including I2C, SPI, USART (Universal Synchronous Asynchronous Receiver Transmitter) and analog inputs through a built in 8-channel, 10-bit Analog-to-Digital converter which for instance gives the possibility to connect up to 8 analog sensors such as photocells, thermistors, microphones, accelerometers etc. The MICAz mote features an IEEE 802.15.4/ZigBee compliant RF transceiver (Chipcon CC2420 [149]), operating in the 2.4 GHz band with a 250-kbps data transfer rate. The MICAz runs TinyOS [150] and is compatible with existing sensor boards that are easily mounted onto the mote. A photo of a MICAz sensor mote is presented in Fig. 11.2.

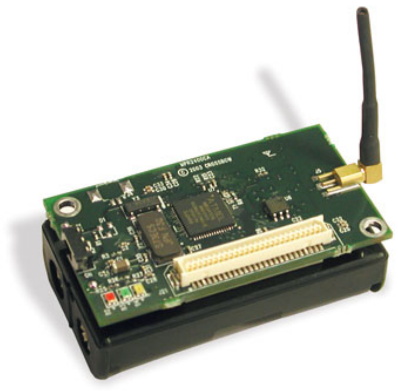


Figure 11.2: MICAz stationary sensor hardware. (Image courtesy of Crossbow Technology [1])

In the context of our testbed, this stationary node is very flexible since it can easily interface with a variety of sensors. However, it has fairly limited computational capabilities, thus it can only execute fairly simple collaboration algorithms. Furthermore, even though it uses a ZigBee transceiver, the version used is not compatible with other ZigBee implementations[1] (e.g., the Xbee Pro RF transceivers which was used as communication devices for the mobile sensor nodes), thus, a TinyOS procedure (component) has been developed to allow interoperability of the Micaz stationary nodes with Xbee Pro RF transceivers.

The stationary sensor nodes have been programmed to receive *position messages* from the base station. These messages are sent during the initialization phase of the testbed and contain information about the stationary sensor's actual position. They can also receive *position request messages* from mobile sensor nodes. When a stationary sensor receives such a packet a decoding procedure extracts the position of the mobile sensor which has sent the packet. The stationary sensor finds its distance to the mobile and decides to respond to the request only if its distance from the mobile is less than r_c. In this case, the stationary node responds to the mobile node and sends a *position reply message* containing its id and position.

Finally, we consider several static event sources that can be placed at various points in the testbed. Such sources include a lamp, a buzzer, a candle or even a chemical source (e.g., an alcohol emitting source). In this case, all sources emit a signal (light, sound, heat or alcohol fumes) that propagates in the testbed environment and can be detected by the appropriate sensor (photocell, microphone, thermistor or chemical sensor [110]). Each stationary sensor periodically samples the environment and when its reading is above a predefined threshold τ_d, called the detection threshold it sends a *detection message* to the base station containing its position. Depending on the strength of the emitted signal, the characteristics of the environment and the sensor hardware, one can determine the detection range r_d of each sensor. As already mentioned, for this

[1]Communication between RF transceivers from different vendors is not straight-forward because of the different ZigBee stacks hosted on transceivers by their vendors.

testbed it is desirable to have sources that emit very weak signals such that not all sensors can detect the presence of the source.

11.3.2 Mobile Sensor Nodes

The mobile sensor node prototypes are based on microchip's PIC16F877A microcontroller [151] and on the Mark III robot chassis platform [152]. The PIC (Peripheral Interface Controller) microcontroller is used for information processing (path planning decisions), for controlling the two servo motors [153] (navigation), for sampling the environment via the various interfaced sensors (microphones, photocells, thermistors, chemical sensors) and for exchanging information with other (mobile or static) sensor nodes or the base station through the interconnected RF transceiver [154]. A photo of a mobile sensor node prototype is presented in Fig. 11.3 while Fig. 11.4 presents a simplified schematic of the PIC microcontroller circuitry along with the several interfaces.

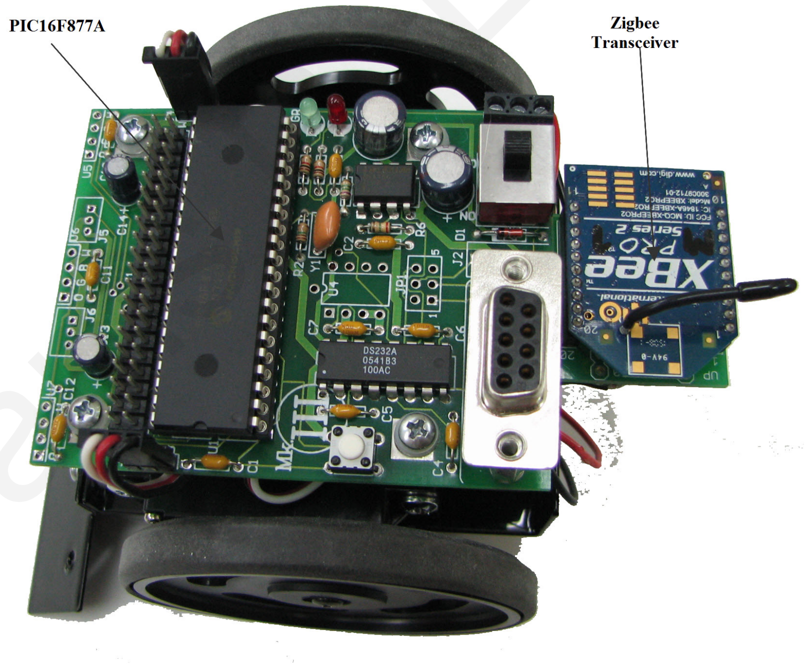


Figure 11.3: The mobile sensor node prototype developed

The developed mobile sensor node has a small size which makes it suitable for experimental investigation for mixed or mobile sensor networks in a laboratory environment. The dimensions ($L \times W \times H$) of the prototype are only $10cm \times 9cm \times 7cm$. Another advantage of the developed mobile node is the low development cost. The complete prototype is built entirely from commercial off-the-shelf components and the cost is of the order of 100 euros. On the other hand, as explained in Section 11.7, the selection of this robot has some drawbacks.

The PIC16F877A is a cheap, popular, powerful and low power 40-pin 8-bit micro-

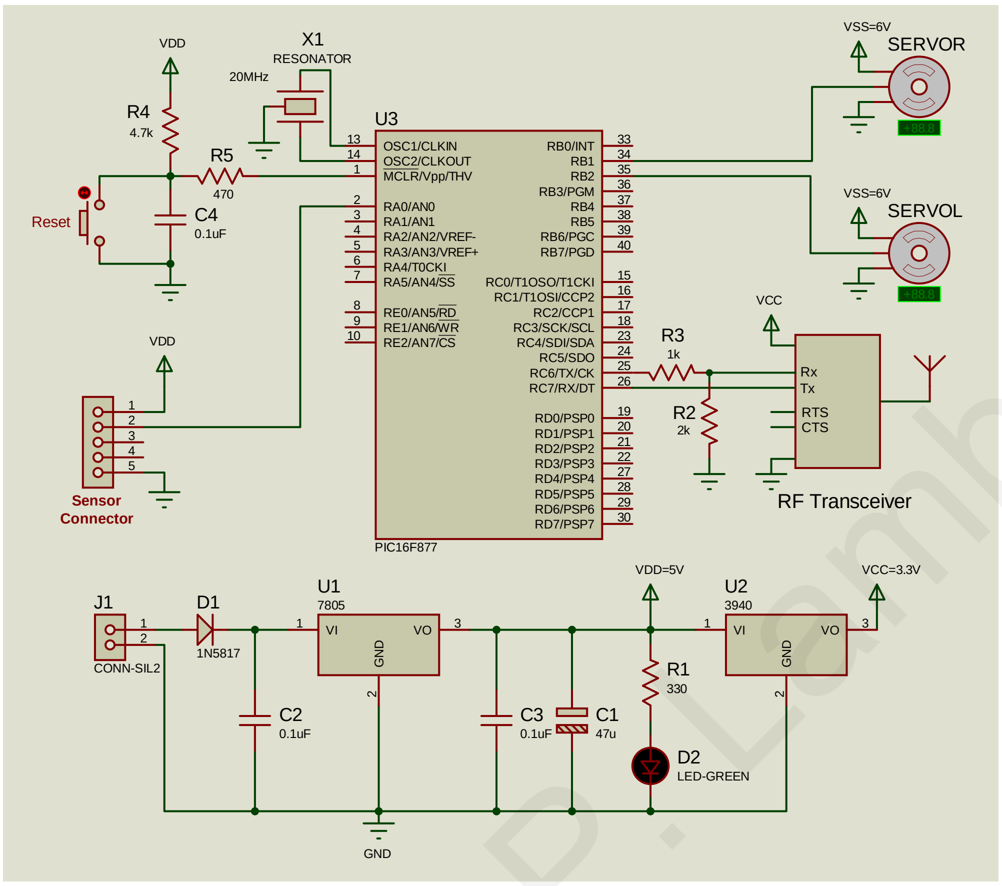


Figure 11.4: A simplified schematic of the mobile sensor node microcontroller circuitry

controller. The 40-pin PDIP package allows rich interconnection and easy assembly. The device offers 14KB (8K × 14-bits words) of Flash program memory, 256 bytes EEPROM, and 368 bytes SRAM data memory which limits the size of the cognitive map that can be maintained by the node. The program memory can be reprogrammed and erased up to 100,000 times which makes the device very good for experimental developments. The PIC16F877A features two 8-bit and one 16-bit internal timers that enable a variety of ways to schedule and manage timing efficiently through hardware. It also supports a variety of serial interfaces like I2C, SPI, USART, digital inputs/outputs (I/O) and analog inputs with 8 built-in channels, 10-bit Analog-to-Digital (A/D) converters. This rich collection of interfaces gives the possibility to connect easily a variety of peripherals such as analog and digital sensors (up to 8 analog sensors), actuators (e.g. motors) as well as other devices like wireless communication modules, external memory modules among many others. The microcontroller is operating at 20 MHz with an external ceramic resonator which enables a 200 ns instruction cycle (5 MIPS).

It is worth mentioning that the PDIP package pinouts is fully compatible with more advanced PICs PDIP package pinouts, like the PIC18F4620 which supports up to 10 MIPS and integrates 64KB program flash and 4KB RAM memory. The internal RAM memory hosted by the PIC micro is vital for robotics applications requiring the implementation of cognitive maps since the size of the memory constrains the size of the map which in turn limits the detailed representation of the environment. Furthermore, the limited computational capabilities of the microcontroller limits the complexity of the path planning algorithms that can be implemented on the node.

An XBee Pro [154] RF transceiver is interconnected to the PIC microcontroller through its integrated USART module. The XBee-PRO module operates within the ISM 2.4 GHz frequency band with a 250 kbps data transfer rate. It is easy-to-use, requires minimal power and provides interoperability with ZigBee devices from other vendors (ZigBee PRO feature set). The RF transceiver uses the ZigBee protocol to communicate with other mobile and static sensor nodes and with the base station (vision positioning system). The PIC microcontroller has been programmed appropriately in order to control the Xbee Pro RF module and to encode or decode the data frames which sends or receives. For instance, using the PIC's USART Received Interrupt, available data frames are directly received from the RF module with negligible delay. Once the PIC receives such a data frame, a decoding procedure extracts the useful information; such information could be the position and orientation of the mobile node or information concerning the cooperation with the neighboring mobile and static nodes.

Specifically, the mobile sensor node receives *position messages* from the base station (vision positioning system) that contain information about its physical position and orientation. These messages are received either periodically or after the mobile explicitly requests the information from the positioning system by sending a *position trigger message* containing its id. As mentioned previously, the mobile nodes also send *position request messages* to the other nodes (mobile or static) of the WSN requesting the position of their neighbors. Once a mobile node receives a position request, it replies with a *position reply message* containing its id and position only if the node that has sent the request is located within a distance r_c from the mobile. Mobile nodes also send/receive collaboration messages to/from their neighboring mobile nodes containing information which enables them to achieve their objectives (e.g. area coverage). The actual information exchanged is described in section 11.5.2.

Like stationary sensors, mobile sensors are sampling the environment with their interfaced sensors (microphone, photocell, thermistor, chemical sensor). When their measurement exceeds a predefined threshold τ_d they also send a *detection message* containing their position to the base station. In a similar fashion to the stationary sensors, there exists a detection range r_d in which the mobile sensors can provide sensing coverage and reliably detect events. The detection range r_d could be adjusted to a value that is equal or smaller than the detection range of stationary sensors.

Finally, the PIC microcontroller of the mobile node uses all the information from its neighbors together with information stored in its cognitive map as well as its own measurements to decide its path. The specific algorithms used for a case study are shown in Section 11.5.

11.3.3 Vision Based Positioning System

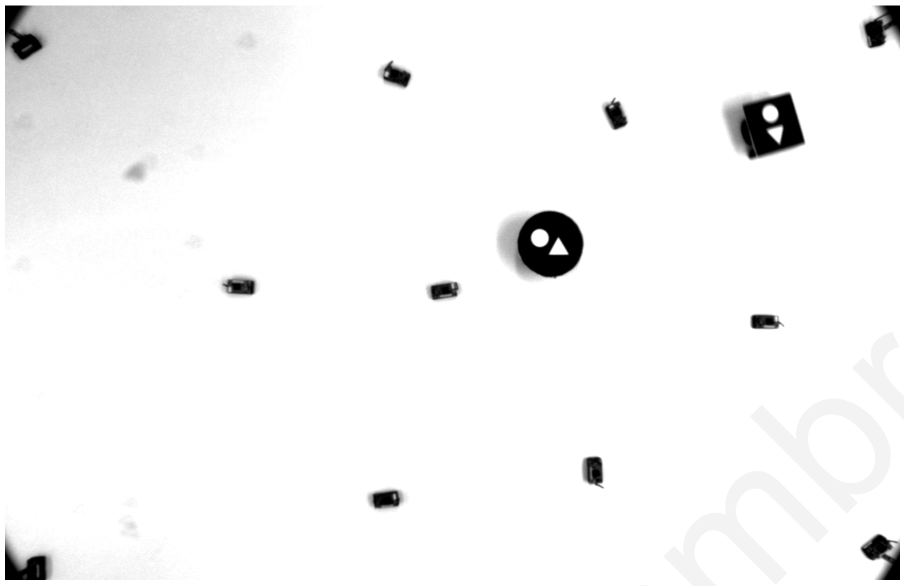
The majority of algorithms proposed for sensor networks and cooperative autonomous agents are based on the assumption that each node (agent) must be aware of its position and orientation as well as the position of the nodes in its neighborhood. In the context of coverage control in large areas, it is anticipated that the mobile nodes will be equipped with a positioning system (e.g. GPS) in order to determine their current position. However, since the test-bed is implemented in a limited size indoor environment, an alternative positioning system is needed to provide the id, position and orientation of the mobile nodes.

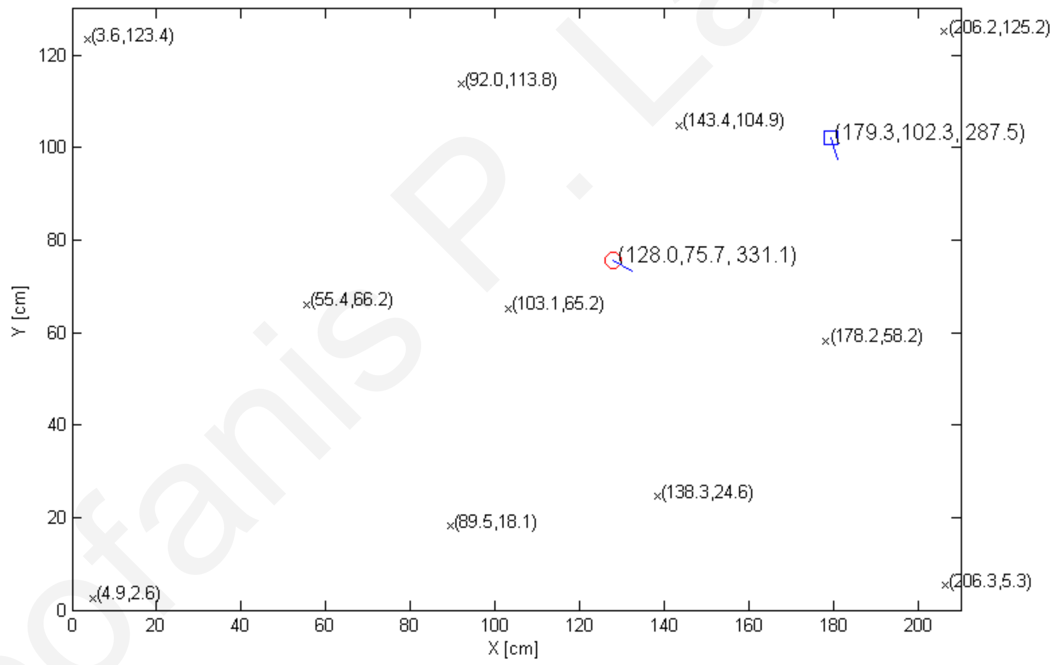
Among the options available we decided to use a vision based system since RSSI (Received Signal Strength Indicator) based positioning has some disadvantages such as increased noise vulnerability [155] and in addition, it does not easily provide direction measurements. Vision-based tracking is used in many robotic laboratories to extract agent position, orientation and trajectory. However, there is currently no accepted standard software solution available, so many research groups resort to developing and using their own custom software. In contrast to other systems, we have implemented our vision system using MATLAB which makes the implementation easy (some basic routines are available) however, the positioning speed is rather slow.

For the coverage control problem, measuring and improving the area coverage requires the position of all sensors. For stationary sensors the problem is fairly easy since the coordinates of each sensor can be preprogrammed at the initialization phase of the testbed (either using the information provided by the vision positioning system or simply by (manually) measuring the location of each node). However, for the mobile sensors, there is a need to record the position and orientation at every step of the experiment, thus the information provided by the localization system is required periodically (at every step).

The testbed arena is captured with a monochrome camera mounted on the ceiling (see Fig. 11.1). The developed vision positioning system consists of the following components: a CCD 1392×1040 pixels camera (Pulnix TM-1325), a camera lens with appropriate focal length (8mm^2) for capturing the whole arena area, a frame grabber (NI PCI-1426) which connects the camera to a PC workstation using camera link interface, an image post processing algorithm (developed in MATLAB) and finally a ZigBee transceiver (XBee-PRO) for transmitting the position/orientation information to the sensor nodes.

The camera can capture images of the sensor field either periodically at a rate of 5 frames per second or when triggered (queried by a mobile sensor node via position triggering messages). A typical field image is shown in Fig. 11.5(a). The testbed arena is $2.20m \times 1.40m$ and covers a size of roughly 1200 by 770 pixels on the image (Fig. 11.5(a)). The captured image is processed in a MATLAB environment in order to extract the positions of the stationary nodes as well as the position and direction of the mobile nodes. The processing of such a frame takes under 200ms and the results are illustrated in Fig. 11.5(b). We point out that in order to identify the mobile nodes


(a) An image of the arena acquired by the overhead camera



(b) Processed image with the position of all sensors and the orientation of the mobile nodes

Figure 11.5: Images of the vision-based positioning system

and their direction, a set of markers are used as shown in Fig. 11.5. The developed positioning system can provide the position of the sensor nodes with precision of $2cm$ and the orientation of the mobile sensor nodes with precision of 3^o degrees. More details on the image post processing algorithm are presented in the Appendix A Once the positioning system identifies the position of each mobile node it sends a unicast *position message* to each mobile node with its coordinates and orientation. Once each mobile has the position information for every other sensor in its neighborhood it can run the decentralized path planning algorithm in order to determine where to go next.

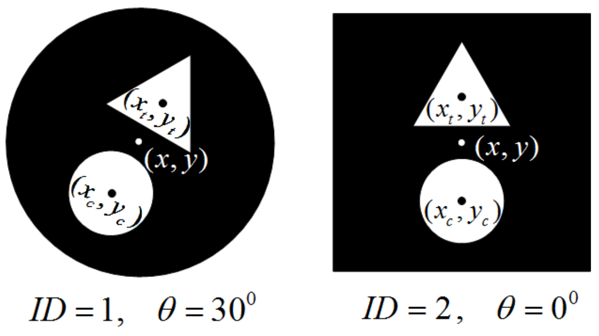


Figure 11.5: Two sample markers coded based on shape compactness. External shape allows the extraction of the mobile node's ID. Internal shapes (circle and triangle) allows the extraction of robot's orientation.

The specifics of the path planning algorithm are described in a following section.

Positioning messages between the base station and the mobile nodes are exchanged in single hop since in coverage control applications it is assumed that the nodes are capable to determine their own location (e.g., they may be equipped with a GPS receiver), thus such communication is not relevant. However, for other applications/objectives, it is also possible to use multi-hop communication with the base station using the distance r_c to emulate limited communication range. Finally, the PC workstation serves also as the base station of the mixed WSN and thus it receives event *detection messages* from sensor nodes and also it measures the performance of the network (area coverage) and visualizes the operations taking place in the mixed sensor network (track mobile nodes trajectories).

11.3.4 Node Communication Protocols

All nodes communicate using the ZigBee protocol stack, though we have used chips from different manufacturers and thus some adaptations were necessary. ZigBee is a low-cost, low-power, wireless networking standard. A ZigBee Personal Area Network (PAN) is formed by nodes joining to a coordinator or to a previously joined router. Once the coordinator defines the operating channel and PAN ID (preprogrammed initially), it can allow routers and end devices to connect to it. When a node joins a network, it receives a 16-bit network address (associated with the preprogrammed id). Once a router has joined the network, it can allow other nodes to join by connecting to it. ZigBee is built upon the physical layer (PHY) and medium access control (MAC) portion of the data link layer (DLL) defined in IEEE standard 802.15.4 for Low-Rate Wireless Personal Area Network (WPAN). The ZigBee protocol stack supports both beacon and non-beacon enabled networks. In non-beacon-enabled networks, an unslotted CSMA/CA channel access mechanism is used and ZigBee devices typically have their receivers continuously active. In beacon-enabled networks, nodes may sleep between beacons (e.g. beacon intervals may range from 15ms to 250s at 250 kbps) thus lowering their duty cycle and extending their battery life.

The developed testbed has implemented a beacon enabled network with star topology. The module connected to the PC workstation serves as the ZigBee coordinator (ZC), the modules on the mobile robots are set as routers (ZR) and the stationary nodes are set as ZigBee End Devices (ZED). As indicated earlier, the deployment area of the testbed is rather small thus all nodes are generally able to hear transmissions from all other nodes. In order to emulate the limited communication range of each node we use the distance r_c thus a node drops packets that have been received from a node which is at a distance greater than r_c. At this point we should also point out that we can extend this packet dropping policy to also emulate non-omnidirectional propagation models (e.g., nodes with directional antennas).

In the ZigBee transceivers there are two types of data transfer transactions. In the first transaction, the data is transferred to the coordinator and in the second transaction the data transfer from the coordinator to the device. When a device needs to transfer data to the coordinator in a beacon enabled network, it listens for the beacon. When the device finds a beacon it synchronizes and transmits the data to the coordinator using slotted CSMA-CA. The coordinator may send an optional acknowledgement frame to complete the transaction. When the coordinator needs to transfer data to a device in a beacon enabled network, it indicates in the network beacon that a data message is pending. The device periodically listens to the network beacon and if a message is pending, transmits a MAC command requesting the data using slotted CSMA-CA. The coordinator acknowledges the successful reception of the data request from the device by transmitting an optional acknowledgement frame. The requested data frame is then sent by the coordinator using slotted CSMA-CA. The device may send an optional acknowledgement frame. The coordinator will then remove the frame from its list of pending frames in the beacon.

11.4 Area Coverage Problem

Area coverage is a measure of the effectiveness of the sensor network to monitor the entire field. It measures the percentage of the field area that is monitored by at least one sensor. To compute the area coverage the entire sensor field (arena area) is discretized into an $X \times Y$ grid and thus, the current state of the sensor field is represented by an $X \times Y$ matrix G_k, $k = 0, 1, \cdots$ stored in the memory of the base station (PC Workstation in Fig. 11.1). This G_k matrix represents the accurate state of the sensor field and is updated as the mobiles move around the field. At every step, we use the following updating rule for every element of matrix G_k.

$$G_{k+1}(i,j) = \begin{cases} 0.5 \cdot G_k(i,j) + 0.5, & \text{if } (i,j) \in D_{\bar{r}_d}(\bar{\mathbf{x}}_s) \\ G_k(i,j), & \text{otherwise} \end{cases} \tag{11.1}$$

where $\bar{\mathbf{x}}_s$ are the coordinates of sensor s in the grid G_k and $D_{\bar{r}_d}(\bar{\mathbf{x}}_s)$ is the set of grid cells covered by sensor s with sensing range r_d.

The area *coverage* C_k over a time interval $[0, k]$ is defined as

$$C_k = \frac{1}{X \times Y} \times \sum_{i=1}^{X} \sum_{j=1}^{Y} G_k\left(i, j\right) \tag{11.2}$$

11.5 Case Study: Distributed Collaborative Path Planning

In this section we present a case study that was investigated using the testbed described previously. The case study is motivated by the coverage control problem and involves a collaborative path planning algorithm that is used by the mobile sensors in order to achieve their objective which is to search (cover) areas that are not covered by static sensors or any other mobile sensor.

The path planning algorithm used in this chapter is exactly the same as the one presented in chapter 6.4. At each step, the mobile considers a finite set of future positions where the node can move to. For each candidate position, the mobile's PIC microcontroller evaluates the cost associated with the position and moves to the one that has the minimum overall cost defined by eq.(11.3)).

$$J(\mathbf{y}) = w_s J_s(\mathbf{y}) + w_t J_t(\mathbf{y}) \tag{11.3}$$

where $J_t(\cdot)$ and $J_s(\cdot)$ are specific objectives and w_t and w_s are non-negative constant weights such that $w_s + w_t = 1$.

11.5.1 Path Cost Functions

As presented in chapter 6.4, $J_t(\cdot)$ is a cost function which penalizes positions that are away from large coverage holes and $J_s(\cdot)$ is a cost function which penalizes positions that are close to static or mobile sensors (i.e., areas covered by other nodes).

The cost function $J_s(\mathbf{y})$ is given by

$$J_s(\mathbf{y}) = \max_{j \in \mathcal{H}_{r_c}(m)} \left\{ \exp\left(- \frac{\|\mathbf{y} - \mathbf{x}_j\|^2}{r_d^2} \right) \right\} \tag{11.4}$$

where $\mathcal{H}_{r_c}(m)$ is the set of all nodes in the communication range r_c of the mobile m. The detection range r_d quantifies the size of the region around the mobile m to be repelled by its neighbors.

The cost function $J_t(\mathbf{y})$ is given by

$$J_t(\mathbf{y}) = \frac{\|\mathbf{y} - \mathbf{x}_t\|}{r_z} \tag{11.5}$$

where given the matrix P_k^m of the mobile [2], r_z defines the radius of the neighborhood where the mobile will search for the center of the biggest coverage hole from its current location and $\mathbf{x}_t$ defines the center of the coverage hole found using the zoom algorithm [9]. As shown in [136, 137], r_z range must be fairly small compared to the sensor field area which implies that a "local" search is sometimes better than a "global" search. Also note that a smaller r_z is advantageous since it implies that less information (i.e. less computation and communication) is needed for the coverage hole estimation.

Note that all cost functions used in eq. (11.3) can be easily computed by the limited resources available at each mobile node.

11.5.2 Distributed Collaboration between Mobile Nodes

Since every mobile determines its path autonomously, a possible problem arises when two or more mobiles are located close to each other. In this case, it is very likely that the information they will use to estimate the next target position will be the same and as a result they will all estimate target locations that are either the same or they are located very close to each other. Clearly, this is not a good collaboration strategy since there is no benefit if they *all* converge to the same point. To avoid this problem we utilize a collaboration protocol that enables mobile nodes to exchange some information in order to avoid converging to the same point.

As mentioned earlier, at every step a mobile node i receives the ids and positions of its neighbors (stationary and mobile nodes) using the *position reply messages* in order to update its P_k^i cognitive map. When i discovers other mobiles in its neighborhood, it sends a unicast *collaboration request message* to these mobile nodes. Once a mobile node j, $j \neq i$ is queried for collaboration, it replies with a *collaboration reply message* which contains its id j, its current target coordinates $\mathbf{x}_t^j(k)$ and possibly its current cognitive matrix P_k^j depending on a flag value described next.

Each mobile node j has a small array in its memory where it tracks the ids of the mobile nodes that were in its communication neighborhood during step $k-1$. If node j was in communication range with node i then the corresponding flag $F_i = 1$ otherwise $F_i = 0$. When a mobile node j receives a *collaboration request message* from mobile i, $j \neq i$, it checks its flag value F_i and replies with a *collaboration reply message* that contains its current P_k^j if $F_i = 0$. If $F_i = 1$ then the *collaboration reply message* contains only the mobile's id j and its current target coordinates $\mathbf{x}_t^j(k)$ (not P_k^j since it was sent at a previous step). Thus as the mobiles stay in a neighborhood range r_c,

[2] P_k^m matrix is stored in the PIC's internal RAM and therefore is restricted by the limited data memory available on mobile nodes, however an approximate state is good enough for the mobiles to achieve their objectives.

they exchange *only* their current positions $\mathbf{x}(k)$ (*position reply messages*) and target coordinates $\mathbf{x}_t(k)$ (*collaboration reply messages*). This protocol significantly limits the communication overhead between mobiles. More information about reducing the collaboration information that needs to be exchanged between the mobiles without serious loss of the performance can be found in chapter 7.

After mobile node i has exchanged collaboration messages with its neighboring mobiles it has all the necessary information to execute the collaboration protocol. At first, it merges its cognitive map P_k^i with the cognitive maps P_k^j, $j \neq i$ it received from its neighbors, so that it does not explore areas already explored by other mobile nodes. Subsequently, the mobile node i utilizes the current target position information received by its neighboring mobiles $\mathbf{x}_t^j(k)$, $j \neq i$ in order to avoid going towards the same point. Once mobile i has received all target points from its neighbors, it forms the matrix $D_{\bar{r}_c}(\bar{\mathbf{x}}_i)$ (which is a copy of the set of the P_k^i cells that correspond to the distance r_c from the mobile i current position) and updates the $D_{\bar{r}_c}(\bar{\mathbf{x}}_i)$ matrix by assuming that the received targets points constituted covered areas. Finally, the mobile node executes the zoom algorithm [9] where the input is the $D_{\bar{r}_c}(\bar{\mathbf{x}}_i)$ matrix and the output is the mobile's next target point $\mathbf{x}_t^i(k)$ which is guaranteed to be different than the target points of its neighboring mobile nodes.

11.5.3 Firmware of Mobile Nodes

The microcontroller firmware was written using the MikroC compiler, developed by MikroElektronika [156]. The mikroC is a powerful, feature rich development environment for PIC microcontrollers. A simplified flowchart of the firmware running on each PIC microcontroller for executing the proposed path planning algorithm is illustrated in Fig. 11.6.

The path planning algorithm described earlier determines the next point where the mobile should move to. This point is given by the distance ρ and the direction θ that it has to turn to. These parameters (ρ, θ) are further processed by the PIC in order to generate the lower level control signals that will drive the two servo motors of the mobile node in order for the mobile to go to the new position. Servomotors have a very simple electrical interface; they have 3 wires, one for power, one for ground and the other for the pulse train. The data wire receives encoded signal in the form of pulse width modulation (PWM). Two PIC's digital outputs ports have been programmed to output such a PWM signal. The duty-cycle of each pulse is related to the rotation direction of the servomotor, while the number of pulses sent is proportional to the rotation angle of the servomotor. The pulses sent have 20ms period (implemented using a 20 ms Timer 1 interrupt) and the positive pulse width contained within those 20ms varies from 1ms to 2ms (for 1ms pulse, the servo rotates in the clockwise direction, for 1.5ms pulse, the servo holds still and for 2ms pulse, the servo rotates in counter-clockwise direction).

Finally, in order to make all interfaces work properly, several initialization and calibration procedures involving the servos, the sensors and the RF transceiver of the mobile nodes have been developed (e.g. generate the appropriate pulse that will drive

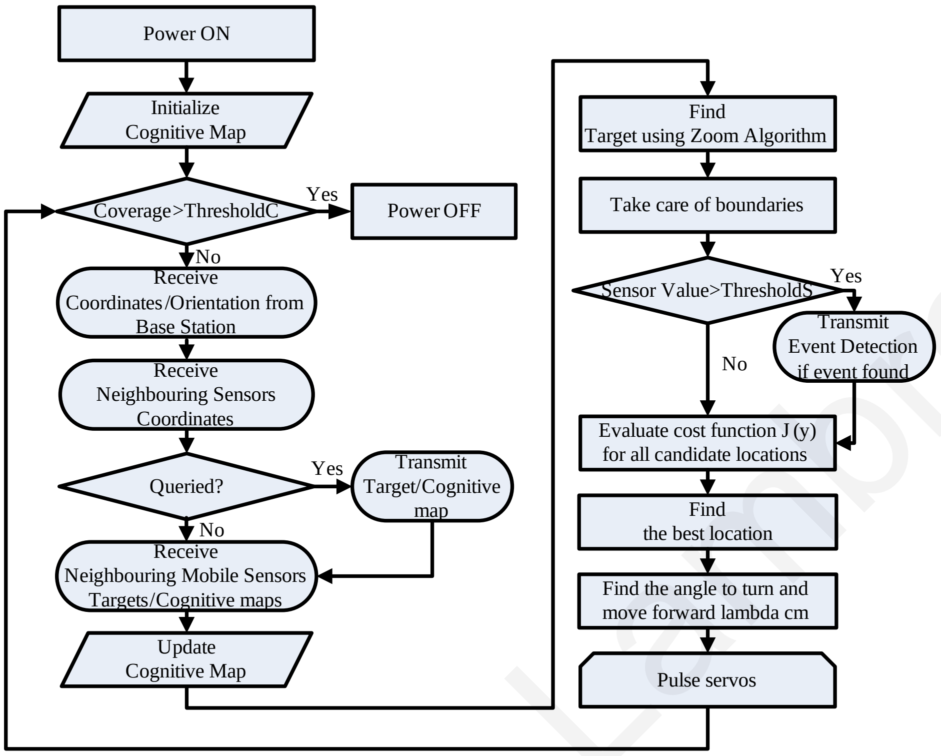

Power ON
Initialize Cognitive Map
Coverage>ThresholdC
Yes
Power OFF
No
Receive Coordinates/Orientation from Base Station
Receive Neighbouring Sensors Coordinates
Queried?
Yes
Transmit Target/Cognitive map
No
Receive Neighbouring Mobile Sensors Targets/Cognitive maps
Update Cognitive Map
Find Target using Zoom Algorithm
Take care of boundaries
Sensor Value>ThresholdS
Yes
Transmit Event Detection if event found
No
Evaluate cost function J (y) for all candidate locations
Find the best location
Find the angle to turn and move forward lambda cm
Pulse servos

Figure 11.6: A simplified flowchart of the firmware running on each PIC microcontroller

the servos in order to navigate the robot to the next desired location).

11.6 Obtained Results for the Experimental Case Study

In this section we present a representative scenario of the movement of the mobile sensor nodes to illustrate and validate the behavior of the proposed path planning algorithm.

The experimental setup consists of a sparse wireless sensor network with 12 stationary nodes and 2 mobile sensor nodes (see Fig. 11.7). The monitored region (arena) has dimensions $220cm \times 140cm$. For the purposes of the experiment, the detection radius of all sensors is set to $r_d = 12cm$ and the dynamic search area range is selected as $r_z = 38cm$. The neighborhood range is set to $r_c = r_z + r_d = 50cm$ and it is also fixed in the sensor node firmware such that any node (mobile or static) should drop any packets received from nodes that are located at a distance greater than r_c. The constant weights are set to $w_t = w_s = 0.5$ and the mobile maneuverability parameters are set to $\rho = 5cm$ and $\phi = 60°$ while for every decision $\nu = 5$ candidate next positions are considered.

Fig. 11.8(a) shows the paths that the mobile nodes followed in the experimental testbed and Fig. 11.8(b) shows the paths followed in the simulation environment. The same set of parameters was used for both the simulation environment as well as the

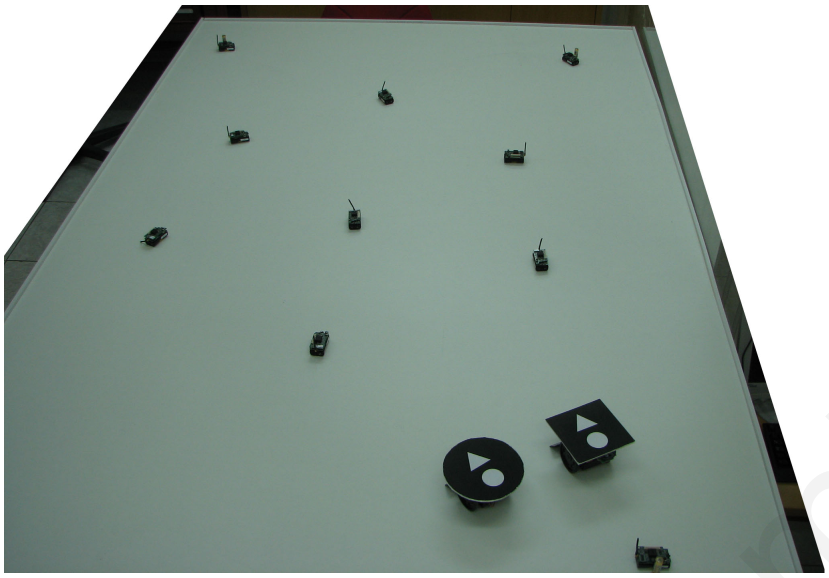


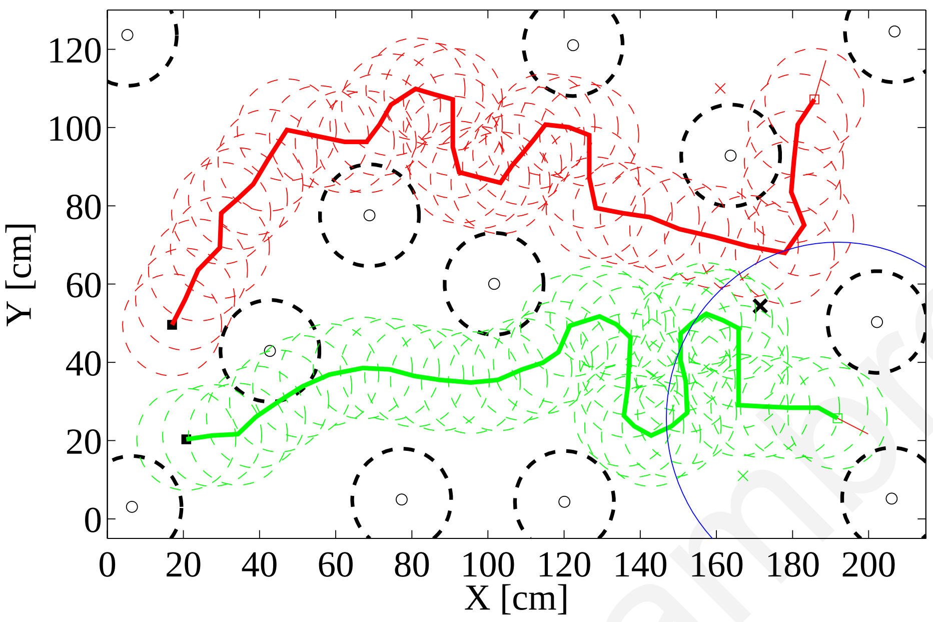
Figure 11.7: The experimental mixed WSN scenario setup consisting of twelve stationary and two mobile sensor nodes

testbed environment. The two mobile nodes navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors. As seen from the paths followed, there is collaboration between the mobile and the stationary sensors in the sense that the mobiles have found two different paths that are least covered by the stationary sensors. Also notice how the two mobiles collaborate and select different paths at the beginning of their journey.

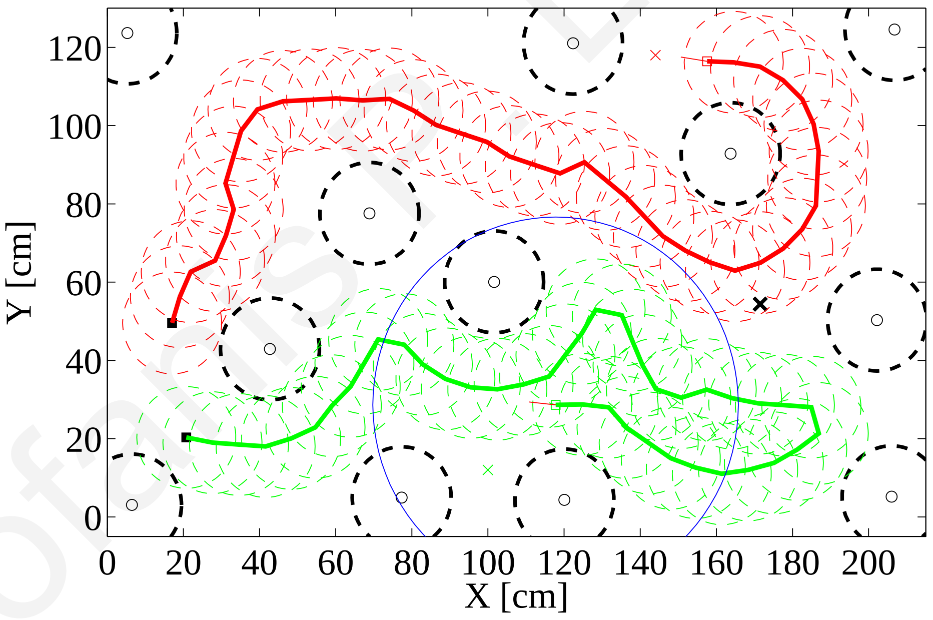
Fig. 11.8 presents a comparison between the area coverage improvement achieved by the mobile nodes in the testbed and simulation environments. The area coverage improvement in the simulation environment is clearly larger compared to the area coverage improvement in the testbed. As indicated in Fig. 11.8(b), in the simulation environment the mobile nodes avoid almost perfectly the regions covered by stationary nodes. In contrast, in the testbed Fig. 11.8(a), there exists some overlap between the area covered by static and mobile sensors which are attributed to various unmodeled parameters (dropped packets, uncertain motion of the mobile nodes).

Finally, for a given scenario (fixed positions of the static nodes and fixed initial positions for the mobiles), we run the path planning algorithms twenty times and recorded the average area coverage due to the navigation of the mobiles. In the simulation environment, the paths of the mobiles for all twenty repetitions are identical (there is no randomness involved). In the experimental testbed however, the paths vary significantly as a result of unmodeled phenomena (dropped packets, inconsistencies in the robot motion).

The average area coverage improvement over all repetitions together with the recorded standard deviation are depicted in Fig. 11.9. These results indicate that the area coverage improvement obtained in the simulation environment is rather "optimistic". The discrepancy between the results obtained through simulation and testbed is due to several factors ranging from hardware limitation of the mobile platform (uncertain motion



(a) The paths followed by two mobile sensor nodes in the experimental testbed



(b) The paths followed by two mobile sensor nodes in the simulation environment

Figure 11.8: The paths followed by two mobile sensor nodes to collaboratively improve the sensor area coverage of the sparse stationary sensor network deployment. The same set of parameters used in both testbed and simulation. The sensing range of each sensor node is indicated by circles with dotted line. The big (blue) circle indicates is the communication range r_c. A video clip of the motion of the two mobile nodes can be found at [2]

and processing accuracy) to asynchronous communication faults (delayed or dropped packets) which are currently being further investigated. These factors affect the path of the mobiles in a rather random way and thus the variance recorded from the twenty repetitions is significant.

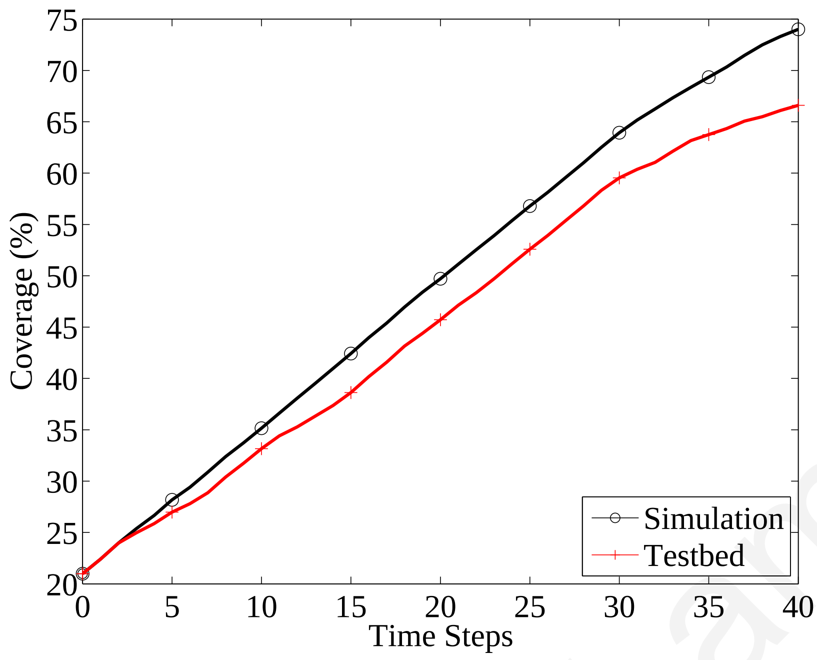


Figure 11.8: Comparison of the coverage improvement accomplished when mobile nodes navigate in the testbed area and when they navigate in the simulation environment

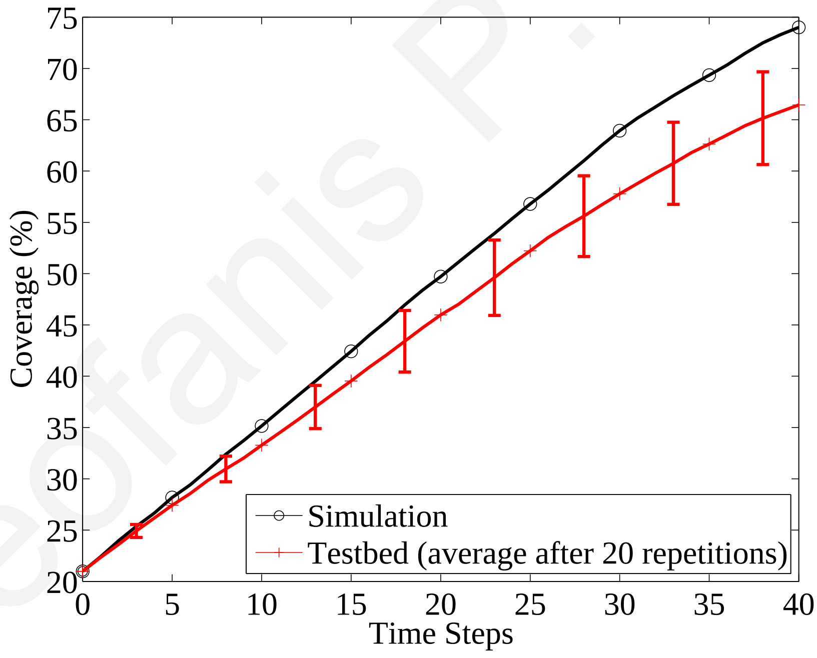


Figure 11.9: The average coverage improvement accomplished after twenty experimental repetitions when the mobile nodes navigate in the testbed versus the coverage improvement accomplished when the mobile nodes navigate in the simulation environment

11.7 Conclusion

Motivated by the need for an experimental platform to validate distributed path planning algorithms in mixed sensor networks, we have developed a new testbed for studying distributed area monitoring techniques in mixed WSNs. The chapter presents an overview of the testbed developed together with details on the static and mobile sensor nodes used and the overall infrastructure developed. The testbed has been used to validate a coverage control case study where mobile nodes collaboratively sample areas not adequately monitored by the static sensor network. The case study has also indicated some of the limitations of the testbed mainly due to the selected mobile platform which turned out to be a little difficult to accurately control.

In the future we plan to upgrade the hardware and software of our testbed in order to make it faster and more robust. Particular attention will be given to the controlled motion of the mobile platforms. In the current version, it was evident that the mobile nodes could not accurately implement the path suggested by the path planning algorithm. In addition, we plan to use the testbed to come up with more accurate simulation models such that the discrepancies between simulation and experimentation are eliminated. This approach can lead to better simulation models but more importantly to more robust algorithms that can tolerate unexpected events.

A Image Post Processing Algorithm for Mobile Node Localization

This section presents the main steps for processing the images received by the camera in order to extract the physical coordinates and orientation of the mobile nodes.

1. The PC workstation triggers the camera to capture an image which is represented by a 1392×1040 matrix **Im** where every matrix element is a number between $0 - 255$ that represents the grayscaling of the pixel.
2. **Im** is converted to a binary image **Ib** by applying an appropriate thresholding technique.
3. The noise of the binary image is removed using some morphology functions based on object size filtering.
4. Identifies the external boundaries of the objects (external connected components).
5. Obtains a labeled external connected components image a from the binary image (distinguish different objects).
6. Extracts the center, perimeter and area of each labeled object.
7. Separates static from mobile nodes using the object area. (During the initialization phase of the test-bed it also finds static sensor nodes centers).
8. Finds mobile nodes ids and centers based on markers that are placed on the top of mobile nodes (see Fig. 11.5) and compactness metric. Compactness is defined by $Compactness = \frac{4\pi.Area}{Perimeter^2}$ (e.g. a circular object has a compactness value of 1)

9. Finds enclosed shapes (circle and triangle) ids and centers in mobile nodes shapes (using compactness metric) and extract mobile nodes orientation by computing θ as follows: $\theta = arctan\left(\frac{\Delta y_{tc}}{\Delta x_{tc}}\right)$ where $\Delta y_{tc} = (y_t - y_c)$ and $\Delta x_{tc} = (x_t - x_c)$ (see Fig. 11.5). Note that Δy_{tc} and Δx_{tc} can be positive or negative, and their signs together define the heading direction of the mobile robot. Thus the actual robot orientation is obtained as follows:

$$\theta = \begin{cases} arctan\left(\frac{\Delta y_{tc}}{\Delta x_{tc}}\right), & if \Delta x_{tc} \geq 0 \; and \; \Delta y_{tc} \geq 0 \\ arctan\left(\frac{\Delta y_{tc}}{\Delta x_{tc}}\right) + \pi, & if \Delta x_{tc} < 0 \; and \; \Delta y_{tc} \geq 0 \\ arctan\left(\frac{\Delta y_{tc}}{\Delta x_{tc}}\right) + \pi, & if \Delta x_{tc} \leq 0 \; and \; \Delta y_{tc} < 0 \\ arctan\left(\frac{\Delta y_{tc}}{\Delta x_{tc}}\right) + 2\pi, & if \Delta x_{tc} > 0 \; and \; \Delta y_{tc} < 0 \end{cases} \tag{11.6}$$

10. Determines the positions of the mobile sensor nodes (or static nodes during the initialization phase) in the physical coordinate system (see Fig. 11.10). Where

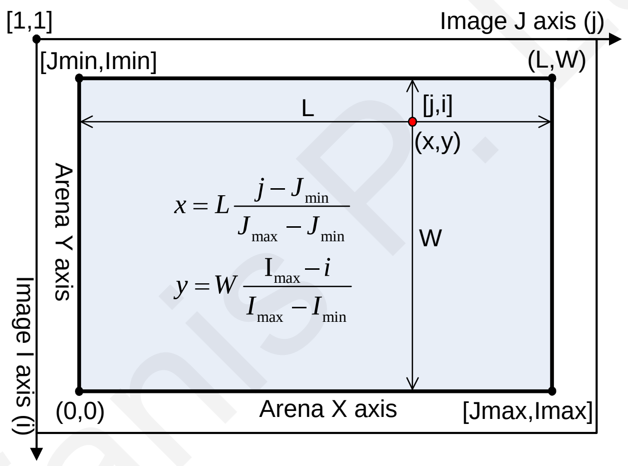


Figure 11.10: Transform the image to physical coordinates in the Test-bed arena

(x, y) is the physical coordinate point in the $L = 220$ cm $\times$ $W = 140$ cm Test-bed arena which corresponds to the pixel point $[j, i]$ (i.e., in the i-th row and j-th column) in the image of $(I_{\max} - I_{\min})$ rows and $(J_{\max} - J_{\min})$ columns of pixels.

11. Transmits the physical position-orientation information to mobile sensor nodes through a USB connected XBee-PRO ZigBee transceiver.

The intra-pixel spatial resolution of the image is given by $\max\left\{\frac{L}{J_{\max}-J_{\min}}, \frac{W}{I_{\max}-I_{\min}}\right\}\frac{cm}{pixel}$. In general, the lower this intra-pixel spatial resolution is, the higher the accuracy of the computed physical position (x, y). Besides, the accuracy of the computed physical position depends also on optical distortion (associated with lens focal length) especially when wide angle lens are used. Thus a calibration process is needed to fix this issue.

Chapter 12

Conclusions, Remarks and Future Directions

12.1 Conclusions

This dissertation contributes to the field of collaborative path planning of multi-robot systems and collaborative monitoring in the context of mixed sensor networks. The main contribution of this research work is the development of a *distributed collaborative area monitoring framework in the context of mixed sensor networks* for efficient dynamic area coverage and fast event detection. It was shown that the selected cost functions as well as the distributed dynamic target assignment scheme achieves near-optimal performance, when compared to the other path-planning approaches investigated. The proposed distributed on-line path-planning algorithm enables the mobile robots to collaborate and compute their path dynamically using only "local" information in the context of mixed WSNs . The algorithm is computationally efficient and adaptive to large numbers of mobile sensor nodes, different static WSN deployments, event source localization algorithms, and can cope with dynamic environments in which spatially and temporally random events occur.

The various parameters and functions associated with the development of the proposed distributed path-planning algorithm were fully investigated and fine-tuned, while the proposed algorithm was compared with other motion planning techniques, which were modified to fit the context of WSNs and require either global or local information. Moreover, different collaboration-communication schemes between the sensor nodes were introduced in a way that ensures a reduced amount of information exchange between nodes without significant loss of performance. In addition, results regarding the optimal path as well as the near optimal search neighborhood of the proposed algorithm have been derived. Finally, the proposed mixed WSN framework has been transferred from simulation to a real-world test-bed to further investigate the performance of the proposed approach experimentally.

Although the coverage and event detection problem, in the context of mobile WSNs, was previously studied, to the best of our knowledge, this is the first time that a general architecture, which combines the coverage and event detection problem with distributed collaborative path-planning algorithms where mobile sensors are searching the environment to minimize the event detection delay, is proposed.

12.2 Remarks

In the course of this research several lessons, concerning the design and implementation of collaborative motion planning methods for multi-robot systems in the context of sensor networks, were drawn.

First, mixed WSNs is a new area of research and methods proposed, in the context of mobile WSNs, usually considered random mobility models or they do not even consider the actual path that mobile nodes should follow (e.g. solve the deployment problem). Second, other methods proposed for finding the worst-case coverage path are based on the concept of Voronoi diagrams and do not consider the complete coverage-search problem. These approaches provide only a single path between two given points in a centralized and static manner (do not consider changes in the field) and hence do not support multiple mobile nodes.

Efficient path planning under random static sensor deployment is complicated, especially when the algorithm is intended to be robust to the sensor field density. This issue has been resolved by normalizing path cost functions as well as using the dynamic receding horizon policy.

In the context of sensor networks and mobile robotics, it is vital for the algorithms used to be simple, smart and efficient in terms of computation and communication. The intention was to follow these requirements throughout the research in this thesis. Simplicity might sometimes opposed to the accuracy of the operations performed and therefore one can further trade off these issues.

The dynamic target policy appears to be a very efficient way of improving both coverage and event detection and also enables simple and efficient collaboration protocols regarding the amount of information that is needed to facilitate the collaboration between mobile nodes. Moreover, it can easily support different event source localization algorithms by switching targets using appropriate decision rules.

Finding the optimal path that a mobile sensor should follow, in order to either maximize the coverage rate over time or minimize the event detection delay, is a very hard problem in the context of mixed WSNs, especially under random deployment of stationary nodes. Even when special simplified cases are considered, the optimal solutions are NP-complete. However, by solving such simplified cases we gain important insights that help us to further improve the heuristics used in the developed path-planning approach.

For instance, it became clearer that large enough uncovered regions, which are close to the mobiles, should be searched first; before moving towards the biggest uncovered regions of the WSN field that are located far away from the mobiles. However, if the searching neighborhood is too small, then the mobile may consume time searching into relatively insignificantly small holes while missing much larger holes. On the other hand, if the neighborhood is too big, then the mobile will waste time when moving straight towards much larger holes avoiding significant holes that are located close to it. This result inspired the further investigation of the search neighborhood size and led to the finding of a surrogate metric that approximates the optimal neighborhood radius. This approximation was based on the statistical properties of the uncovered regions created by the stationary sensor random deployment.

Finally, we have learned that analysis is only useful up to a point, after which the true proof of performance is in the implementation of the algorithms on real robot platforms. This is a fundamental engineering perspective, as no tractable simulation model will be able to capture the intricacies of the dynamics, noise, and computational processes so the final proof must be an implementation. Therefore, the proposed distributed collaborative path-planning algorithm has also been evaluated experimentally in a developed test-bed. Implementation can lead to deeper investigation, enabling more robust algorithms. However, scaling down such large scale systems in a small-sized test-bed sometimes becomes an even more difficult task than in reality.

12.3 Future Directions

Future research, stemming from this dissertation, can lead to a number of several directions.

The first, immediate, goal to be achieved is to extent the proposed path-planning approach further by incorporating the probabilistic sensing model (see eq.2.2) and a dynamic speed policy. Using the probabilistic sensing model one should reexamine how the detection rule, as well as the map update and merging policies, should be updated. One could think of a detection rule based on concurrent samples of neighboring mobile nodes (consider the case of dynamic events), a map update rule like $G_{k+1} = 1-(1-G_k)*(1-p)$ (see eq.2.2) and a merging policy like in chapter 7. However, considering such a merging policy might be too pessimistic. Using a dynamic speed policy (i.e. modifying the speed of the mobile at each step) should enable mobiles to make more precise movements, which might minimize the distance the mobiles moved (i.e. the energy needed for mobility) and simultaneously maximized the dynamic coverage performance over time. This dynamic speed policy can be developed by extending the next step decision policy to consider candidate positions on a circular sector instead of an arc.

Another direction for future work is to move beyond the class of cost functions considered so far. It would be interesting to consider cost functions that depend on time or the residual energy of mobile nodes. Also, this will require the modeling of the energy consumption in the mobile node, which mainly depends on mobility and communication. Additionally, other cost functions could be investigated; for instance, an obstacle-avoidance cost function as well as a communication cost function that will force mobile nodes to stay in communication and, therefore, enable a swarm behavior.

Furthermore, another worth-exploring issue is the investigation of the lower and upper bounds for the dynamic area coverage and event-detection performance of the proposed mixed WSN under various search strategies of mobile sensor nodes. For each search strategy, ranging from random mobility to optimal cooperation, it would be very important to derive approximate formulas for the performance of the mixed WSN. This could support the network designer to easily determine the required number of mobile, or static sensors, or the speed of mobile sensors, as well as other parameters of the mixed WSN, to achieve a predetermined area-coverage in a given period of time, or to achieve a desired expected detection delay in the deployment area.

Experimental evaluation of the proposed framework indicates issues that need further investigation and development. First, we need to overcome hardware and software limitations of the experimentation infrastructure to make the testbed faster and more robust. Particular attention should be given to accurately controlling the motion of the mobile platforms as well as improving the accuracy and time response of the localization system. In addition, experimentation indicates various unmodeled parameters such as asynchronous communication faults (dropped or delayed packets) and uncertainties in motion control. These parameters can be integrated in the developed simulation model which will seek to create more robust algorithms that can tolerate such unexpected events. Another important direction would be to investigate the impact of node mobility in Medium Access Control (MAC) protocols. For instance, the IEEE 802.15.4 MAC protocol (used in ZigBee) for low-rate wireless personal area networks is designed mainly for static sensor networks and its capability to support mobile sensor networks has not yet been established.

Additionally, the developed collaborative path-planning method can be extended to solve other types of path-planning problems such as the mobile sink path-planning problem [158] for collecting information in WSNs, the automated guided vehicles routing problem for transporting containers in port terminals as well as the coverage path-planning problem for automated agricultural machines. For the last two problems, desirable solution approaches should support automated vehicle navigation with variable speed (as well as a stopping option). This is easily implementable in the proposed path-planning algorithm.

Finally, another long term future direction would be to address the problem of odor plume source localization. This problem has become a prominent research area in recent years. However, only a limited number of proposed approaches support multiple collaborative mobile agents. To address this problem we must first develop an efficient and realistic odor plume simulation model and then develop a localization algorithm that can support multiple cooperative mobile agents. Afterwards, additional issues, such as consensus algorithms and swarm stability, could be investigated.

Research work on the aforementioned open questions is still ongoing and we expect that these issues will motivate new results and new insights for multi-robot control and mobility in wireless sensor networks.

Bibliography

[1] *Crossbow Micaz sensor mote*, 2010, [online]. Available from: `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf`.

[2] [online]. Available from: `http://www2.ucy.ac.cy/~faniseng/senetslab/robots.html`.

[3] K. Dantu and G. Sukhatme, "Robomote: Enabling mobility in sensor networks," in *IEEE/ACM Fourth International Conference on Information Processing in Sensor Networks*, Apr 2005, pp. 404–409.

[4] T. Lambrou, C. Anastasiou, and C. Panayiotou, "A nephelometric turbidity system for monitoring residential drinking water quality," in *1st International Conference on Sensor Networks Applications, Experimentation and Logistics, SensAppeal 2009*, Athens, Greece, Sept. 2009.

[5] A. Ghosh and S. K. Das, *Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions.* John Wiley & Sons, 2006, ch. 9.

[6] M. Cardei and J. Wu, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems.* CRC Press, 2004, ch. 19.

[7] N. Ahmed, S. S. Kanhere, and S. Jha, "The holes problem in wireless sensor networks: a survey," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 4–18, 2005.

[8] X. Shen, J. Chen, and Y. Sun, "Grid scan: A simple and effective approach for coverage issue in wireless sensor networks," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 8, 2006, pp. 3480–3484.

[9] T. Lambrou and C. Panayiotou, "Collaborative event detection using mobile and stationary nodes in sensor networks," in *The 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2007*, New York, USA, Nov. 2007.

[10] G. Wang and T. L. Porta, "Movement-assisted sensor deployment," in *Proceedings of the IEEE Infocom*, 2004.

[11] G. Wang and T. LaPorta, "A bidding protocol for deploying mobile sensors," in *Proceedings. 11th IEEE International Conference on Network Protocols*, 4-7 Nov. 2003, pp. 315–324.

[12] A. Ghosh, "Estimating coverage holes and enhancing coverage in mixed sensor networks," in *Local Computer Networks,*, 16-18 Nov. 2004, pp. 68–76.

[13] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *Pervasive Computing*, vol. 2, no. 4, pp. 34–43, 2003.

[14] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *WSNA '03.* New York, NY, USA: ACM Press, 2003, pp. 115–121.

[15] S. Dhillon, K. Chakrabarty, and S. Iyengar, "Sensor placement for grid coverage under imprecise detections," in *FUSION*, 2002.

[16] J.-C. Latombe, *Robot Motion Planning.* New York: Kluwer, 1992.

[17] M. Polycarpou, Y. Yang, Y. Liu, and K. Passino, *Cooperative Control: Models, Applications and Algorithms.* Kluwer Academic Publishers, 2003, vol. 1, ch. Cooperative Control Design for Uninhabited Air Vehicles, pp. 283–321.

[18] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *INFOCOM*, 2001, pp. 1380–1387.

[19] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *MobiCom '01.* New York, NY, USA: ACM Press, 2001, pp. 139–150.

[20] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," in *Proceedings of the ACM MobiHoc 2005 Conference*, 2005.

[21] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Proceedings of 44rd IEEE Conference on Decision and Control*, 2005.

[22] D. W. Gage, "Command control for many-robot systems," in *Proceedings of the AUVS-92*, 1992.

[23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[24] S. Poduri and G. Sukhatme, "Constrained coverage for mobile sensor networks," in *Robotics and Automation, 2004. Proceedings. ICRA.*, vol. 1, 2004, pp. 165–171 Vol.1.

[25] A. Howard, M. Mataric, and G. Sukhatme, "Mobile sensor network deployment using potential fields:a distributed, scalable solution to the area coverage problem," in *DARS*, 2002.

[26] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proceedings of the IEEE Infocom*, 2003.

[27] B. Liu and D. Towsley, "A study of the coverage of large-scale sensor networks," in *in Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Oct. 2004, pp. 475–483.

[28] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, pp. 61–91, 2004.

[29] P. B. A. Hossain and S. Chakrabarti, "Sensing models and its impact on network coverage in wireless sensor network," in *in Proceedings of the 3rd IEEE International Conference on Industrial and Information Systems*, Dec 2008.

[30] P. Hall, *Introduction to the Theory of Coverage Processes.* John Wiley & Sons, 1988.

[31] A. Baddeley, *Stochastic Geometry.* Springer Berlin, 2007, ch. Spatial Point Processes and their Applications, pp. 1–75.

[32] G. Veltri, G. Qu, Q. Huang, and M. Potkonjak, "Minimal and maximal exposure path algorithms for wireless embedded sensor networks," in *IN PROC. OF SENSYS.* ACM Press, 2003, pp. 40–50.

[33] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: Concepts and applications of Voronoi diagrams*, 2nd ed., ser. Probability and Statistics. Wiley, 2000, 671 pages.

[34] A. Howard, M. Mataric, and G. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots, Special Issue on Intelligent Embedded Systems*, vol. 13, pp. 113–126, 2001.

[35] G. Wang, G. Cao, and T. F. L. Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[36] S. Skyum, "A simple algorithm for computing the smallest enclosing circle," *Information Processing Letters*, vol. 37, no. 3, pp. 121–125, 1991.

[37] M. Bisnik, A. Abouzeid, and A. Isler, "Stochastic event capture using mobile sensors subject to a quality metric," *Robotics, IEEE Transactions on*, vol. 23, no. 4, pp. 676–692, Aug. 2007.

[38] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation.* MIT Press, June 2005.

[39] Y. K. Hwang and N. Ahuja, "Gross motion planning - a survey," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 219–291, 1992.

[40] C. Eldershaw, "Heuristic algorithms for robot motion planning," Ph.D. dissertation, University of Oxford, 2001.

[41] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning-a chronological review," in *in Proceedings of World Academy of Science, Engineering and Technology*, 2007, pp. 101–106.

[42] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[43] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 2, pp. 224 –241, mar/apr 1992.

[44] D. Koditschek, "Exact robot navigation by means of potential functions: Some topological considerations," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1987.

[45] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[46] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, pp. 2861–2866.

[47] S. Carpin, "Randomized motion planning - a tutorial," *International Journal of Robotics and Automation*, vol. 21, no. 3, pp. 184–196, 2006.

[48] L. E. Kavraki, S. P., J.-C. Latombe, and O. M., "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[49] S. M. Lavalle, *Planning algorithms.* Cambridge University Press, 2006.

[50] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Netw.*, vol. 8, no. 1, pp. 125–133, 1995.

[51] C. Kozakiewicz and M. Ejiri, "Neural network approach to path planning for two dimensional robot motion," in *Intelligent Robots and Systems '91. Proceedings IROS '91*, Nov 1991, pp. 818–823 vol.2.

[52] A. Zhu and S. Yang, "A neural network approach to dynamic task assignment of multirobots," *Neural Networks, IEEE Transactions on*, vol. 17, no. 5, pp. 1278–1287, Sept. 2006.

[53] J. Solano and D. Jones, "Generation of collision-free paths, a genetic approach," in *Genetic Algorithms for Control Systems Engineering, IEE Colloquium on*, May 1993.

[54] R. Ramakrishnan and S. Zein-Sabatto, "Multiple path planning for a group of mobile robot in a 2-d environment using genetic algorithms," in *SoutheastCon 2001. Proceedings. IEEE*, 2001.

[55] H.-Q. Min, J.-H. Zhu, and X.-J. Zheng, "Obstacle avoidance with multi-objective optimization by pso in dynamic environment," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 5, Aug. 2005, pp. 2950–2956.

[56] M. Saska, M. Macas, L. Preucil, and L. Lhotska, "Robot path planning using particle swarm optimization of ferguson splines," in *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, Sept. 2006, pp. 833–839.

[57] L. Wang, Y. Liu, H. Deng, and Y. Xu, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, Dec. 2006, pp. 1233–1238.

[58] E. Masehian and M. Amin-Naseri, "A tabu search-based approach for online motion planning," in *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, Dec. 2006, pp. 2756–2761.

[59] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[60] "Trilobite 2.0 vacuum cleaner," 2009. [Online]. Available: http://trilobite.electrolux.com

[61] "Husqvarna automower," 2009. [Online]. Available: http://www.husqvarna.com/us/homeowner/products/robotic-mowers/automower-solar-hybrid/

[62] Y. H. Z. Cao and E. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *Journal of robotic systems*, pp. 87–102, 1988.

[63] A. Zelinsky, R. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *In Proceedings of International Conference on Advanced Robotics*, 1993, pp. 533–538.

[64] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001.

[65] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[66] D. W. Gage, "Many-robot mcm search systems," in *Proceedings of Autonomous Vehicles in Mine Countermeasures Symposium*, 1995.

[67] T. Balch, "The case for randomized search," in *IEEE International Conference on Robotics and Automation*, 2000.

[68] M. Mazo and K. H. Johansson, "Path-planning for robust area coverage: Evaluation of five coordination strategies," 2008.

[69] S. Yang and C. Luo, "A neural network approach to complete coverage path planning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 718–724, Feb. 2004.

[70] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "Cooperative sweeping by multiple mobile robots," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, Apr 1996.

[71] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: an algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, pp. 109–142, 2008.

[72] R. M. Murray, "Recent research in cooperative control of multi-vehicle systems," 2007.

[73] B. O. Koopman, "The theory of search ii - target detection," *Operations Research*, vol. 4, pp. 503–531, 1956.

[74] L. D. Stone, *Theory of optimal search.* Academic Press, New York, 1975.

[75] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint, and M. Baum, *Cooperative Control and Optimization.* Kluwer Academic Publishers, 2002, vol. 66, ch. Cooperative Control for Autonomous Air Vehicles, pp. 233–271.

[76] A. A. M. Yanli Yang and M. M. Polycarpou, "Decentralized cooperative search in uav's using opportunistic learning," in *In Proceedings of the 2002 AIAA Guidance, Navigation and Control Conference*, 2002.

[77] Y. Liao, Y. Jin, A. Minai, and M. Polycarpou, "Information sharing in cooperative unmanned aerial vehicle teams," in *In Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, Dec. 2005, pp. 90–95.

[78] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," vol. 1, 2003.

[79] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 1327–1332.

[80] ——, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, 2004.

[81] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *Computer Graphics*, 1987, pp. 25–34.

[82] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, June 2003.

[83] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *Automatic Control, IEEE Transactions on*, vol. 51, no. 3, pp. 401–420, March 2006.

[84] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[85] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *Control Systems Magazine, IEEE*, vol. 27, no. 2, pp. 71–82, 2007.

[86] C. Cassandras and W. Li, "Sensor networks and cooperative control," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec. 2005, pp. 4237–4238.

[87] *Khepera II Robots*, 2010, [online]. Available from: `http://www.k-team.com/mobile-robotics-products/khepera-ii`.

[88] M. Rahimi, R. Mediratta, K. Dantu, and G. Sukhatme, "A testbed for experiments with sensor/actuator networks," Institute for Robotics and Intelligent Systems, Tech. Rep., 2002.

[89] S. Bergbreiter and K. Pister, "Cotsbots: an off-the-shelf platform for distributed robotics," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2003, pp. 1632 – 1637 vol.2.

[90] S. Graham and P. R. Kumar, "The convergence of control, communication, and computation," in *In Proceedings of Personal Wireless Communication (PWC*. Springer-Verlag, 2003, pp. 458–475.

[91] Z. Wang, Z. Song, P.-Y. Chen, A. Arora, D. Storniont, and Y. Q. Chen, "Masmote - a mobility node for mas-net (mobile actuator sensor networks)," in *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, 2004, pp. 816 –821.

[92] L. Cremean, W. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. Murray, "The caltech multi-vehicle wireless testbed," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, 2002, pp. 86–88.

[93] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, "Coordination and control experiments on a multi-vehicle testbed," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6, 2004, pp. 5315–5320 vol.6.

[94] Y. Gu, B. Seanor, G. Campa, M. Napolitano, S. Gururajan, and L. Rowe, "Autonomous formation flight: Hardware development," in *Control and Automation, 2006. MED '06. 14th Mediterranean Conference on*, June 2006.

[95] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in

Robotics and Automation, 2007 IEEE International Conference on, April 2007, pp. 361–366.

[96] A. Jensen, M. Baumann, and Y. Chen, “Low-cost multispectral aerial imaging using autonomous runway-free small flying wing vehicles,” in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, July 2008.

[97] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, “Cooperative grasping and transport using multiple quadrotors,” in *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, Nov 2010.

[98] K. Alexis, G. Nikolakopoulos, and A. Tzes, “Constrained optimal attitude control of a quadrotor helicopter subject to wind-gusts: Experimental studies,” in *American Control Conference, ACC 2010*, Baltimore, Maryland, USA, Jun. 2010, pp. 4451–4455.

[99] K. Alexis, G. Nikolakopoulos, A. Tzes, and L. Dritsas, “Coordination of helicopter uavs for aerial forest-fire surveillance,” in *Applications of Intelligent Control to Engineering Systems*, ser. Intelligent Systems, Control and Automation: Science and Engineering, K. P. Valavanis, Ed. Springer Netherlands, 2009, vol. 39, pp. 169–193.

[100] A. Bicchi, A. Danesi, G. Dini, S. La Porta, L. Pallottino, I. Savino, and R. Schiavi, “Heterogeneous wireless multirobot system,” *Robotics Automation Magazine, IEEE*, vol. 15, no. 1, pp. 62 –70, march 2008.

[101] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki - a lightweight and flexible operating system for tiny networked sensors,” in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, ser. LCN ’04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462.

[102] C. Reed, R. Hudson, and K. Yao, “Direct joint source localization and propagation speed estimation,” in *Acoustics, Speech, and Signal Processing, 1999. ICASSP ’99. Proceedings., 1999 IEEE International Conference on*, vol. 3, mar 1999, pp. 1169 –1172 vol.3.

[103] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli, “Blind beamforming on a randomly distributed sensor array system,” *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 8, pp. 1555 –1567, oct 1998.

[104] D. Li and Y. H. Hu, “Energy based collaborative source localization using acoustic micro-sensor array,” *J. Applied Signal Processing*, vol. 2003, pp. 321–337, 2003.

[105] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, 2007.

[106] G. Strang, *Linear algebra and its applications*, 3rd ed. Harcourt Brace Jovanovich College Publishers, 1988.

[107] G. T. Csanady, *Turbulent diffusion in the environment.* Springer, 1973.

[108] H. Ekkehard, *Environmental Modeling Using MATLAB.* Springer, 2007.

[109] R. A. Russell, "Survey of robotic applications for odor-sensing technology," *I. J. Robotic Res.*, vol. 20, no. 2, pp. 144–162, 2001.

[110] "Figaro engineering inc." [Online]. Available: http://www.figaro.co.jp

[111] H. Ishida, G. Nakayama, T. Nakamoto, and T. Moriizumi, "Controlling a gas/odor plume-tracking robot based on transient responses of gas sensors," *Sensors Journal, IEEE*, vol. 5, no. 3, pp. 537–545, June 2005.

[112] A. Lilienthal, H. Ulmer, H. Frohlich, A. Stutzle, F. Werner, and A. Zell, "Gas source declaration with a mobile robot," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 2004.

[113] J. L. W. Richard W Dearden, Zeyn A Saigol and B. J. Murton, "lanning for auvs: Dealing with a continuous partially-observable environment," in *In Workshop on Planning and Plan Execution for Real-World Systems, ICAPS 2007*, September 2007.

[114] J. Farrell, S. Pang, and W. Li, "Plume mapping via hidden markov methods," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 33, no. 6, pp. 850–863, Dec. 2003.

[115] G. Kowadlo and R. A. Russell, "Robot odor localization: A taxonomy and survey," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 869–894, 2008.

[116] W. L. R. T. C. J. Farrell, J. Murlis, "Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes," *Environmental Fluid Mechanics, vol.2*, pp. 143–169, 2002.

[117] E. K. C. Lytridis and S. Virk, "A systematic approach to the problem of odour source localisation," *Autonomous Robots, Springer*, pp. 261–276, 1996.

[118] D. B. Dusenbery, *Sensory ecology: How organisms acquire and respond to information*, 1992.

[119] R. Russell, D. Thiel, R. Deveza, and A. Mackay-Sim, "A robotic system to locate hazardous chemical leaks," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, May 1995.

[120] M. T. T. N. T. M. H. Ishida, K. Hayashi and R. Kanzaki, "Odour-source localization system mimicking behavior of silkworm moth," *Sensors and Actuators A: Physical*, pp. 225–230, 1996.

[121] W. Li, J. Farrell, S. Pang, and R. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *Robotics, IEEE Transactions on*, vol. 22, no. 2, pp. 292–307, April 2006.

[122] A. Hayes, A. Martinoli, and R. Goodman, “Distributed odor source localization,” *Sensors Journal, IEEE*, vol. 2, no. 3, Jun 2002.

[123] Q. Liao and E. Cowen, “The information content of a scalar plume - a plume tracing perspective,” *Environmental Fluid Mechanics, vol.2., no. 1-2., pp.9-34*, pp. 9–34, 2002.

[124] M. Vergassola, E. Villermaux, and B. I. Shraiman, “ ‘infotaxis’ as a strategy for searching without gradients,” *Nature*, vol. 445, no. 7126, pp. 406–409, January 2007.

[125] S. Pang and J. A. Farrell, “Chemical plume source localization,” *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, vol. 36, pp. 1068–1080, 2006.

[126] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, “A biologically-inspired algorithm implemented on a new highly flexible multi-agent platform for gas source localization,” in *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, Feb. 2006, pp. 573–578.

[127] V. Christopoulos and S. Roumeliotis, “Multi robot trajectory generation for single source explosion parameter estimation,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, April 2005.

[128] D. Zarzhitsky, D. Spears, and W. Spears, “Distributed robotics approach to chemical plume tracing,” in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug. 2005.

[129] J. R. C. Savarese and K. Langendoen, “Robust positioning algorithms for distributed ad-hoc wireless sensor networks,” in *USENIX Technical Annual Conference*, 2002.

[130] M. Michaelides and C. Panayiotou, “Event detection using sensor networks,” in *Proceedings of 45rd IEEE Conference on Decision and Control*, Dec. 2006.

[131] V. Ablavsky and M. Snorrason, “Optimal search for a moving target: a geometric approach,” in *Proceedings of AIAA Guidance, Navigation, and Control Conference*, Denver, CO, USA, Aug. 2000.

[132] T. Lambrou and C. Panayiotou, “Improving area coverage using mobility in sensor networks,” in *International Conference on Intelligent Systems And Computing: Theory And Applications, ISYC 2006*, 6-7 Jul. 2006.

[133] A. Dhariwal, G. S. Sukhatme, and A. A. Requicha, “Bacterium-inspired robots for environmental monitoring,” in *IEEE International Conference on Robotics and Automation.* New Orleans, Louisiana: IEEE, Apr 2004.

[134] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, “Range-free localization schemes for large scale sensor networks,” in *MobiCom ’03: Proceedings of the 9th annual international conference on Mobile computing and networking*, San Diego, CA, USA, 2003.

[135] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach.* Morgan Kaufmann Publishers Inc., 2004, ch. 2.

[136] T. Lambrou and C. Panayiotou, "Collaborative area monitoring using wireless sensor networks with stationary and mobile nodes," *EURASIP Journal on Advances in Signal Processing*, pp. 1–16, 2009.

[137] ——, "Distributed collaborative path planning in sensor networks with multiple mobile sensor nodes," in *17th Mediterranean Conference on Control and Automation, IEEE MED 2009*, Thessaloniki, Greece, Jun. 2009.

[138] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209 – 219, 2006.

[139] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[140] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237 – 244, 1977.

[141] T. Lambrou, S. Felici-Castell, C. Panayiotou, and B. Beferull-Lozano, "Exploiting mobility for efficient coverage in sparse wireless sensor networks," in *10th International Symposium on Wireless Personal Multimedia Communications, WPMC 2007*, Jaipur, India, Dec. 2007.

[142] T. Lambrou, C. Panayiotou, S. Felici-Castell, and B. Beferull-Lozano, "Exploiting mobility for efficient coverage in sparse wireless sensor networks," *Wireless Personal Communications*, pp. 1–15, 2009.

[143] J. M. Dietrich Stoyan, Wilfrid S. Kendall, *Stochastic Geometry and Its Applications, 2nd Edition.* John Wiley & Sons, 1995.

[144] P. Hall, "Mean and variance of vacancy for distribution of k-dimensional spheres within k-dimensional space," *Journal of Applied Probability*, vol. 21, pp. 738–752, 1984.

[145] M. G. Kendall and P. A. P. Moran, *Geometrical Probability.* New York: Hafner, 1963.

[146] T. Lambrou and C. G. Panayiotou, "Area coverage vs event detection in monitoring applications using mixed sensor networks," in *8th World Congress of the International Federation of Automatic Control, IFAC WC 2011*, Milano, Italy, Aug. 2011, accepted.

[147] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "Swistrack - a flexible open source tracking software for multi-agent systems," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, pp. 4004 –4010.

[148] *ATMEL ATmega128L Microcontroller*, 2010, [online]. Available from: `http://www.atmel.com/dyn/resources/prod_documents/2467S.pdf`.

[149] *Chipcon CC2420 RF transceiver from Texas Instruments*, 2010, [online]. Available from: `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[150] *TinyOS operating system*, 2010, [online]. Available from: `http://www.tinyos.net/`.

[151] *Microchip PIC16F877A Microcontroller*, 2010, [online]. Available from: `http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf`.

[152] *Mark III Chassis Kit*, 2010, [online]. Available from: `http://www.junun.org/MarkIII/Info.jsp?item=2`.

[153] *Parallax (Futaba) Continuous Rotation Servo*, 2010, [online]. Available from: `http://www.parallax.com/Portals/0/Downloads/docs/prod/motors_/900-00008-CRServo-v2.0.pdf`.

[154] *Digi XBee-PRO 802.15.4 OEM RF Module*, 2010, [online]. Available from: `http://ftp1.digi.com/support/documentation/manual_xb_oem-rf-modules_802.15.4_v1.xAx.pdf`.

[155] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks.* Wiley, 2005, ch. 9: Localization and positioning, p. 526.

[156] *MikroElektronika mikroC Compiler for PIC*, 2010, [online]. Available from: `http://www.mikroe.com/eng/products/view/7/mikroc-pro-for-pic/`.

[157] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *in IEEE Computer Vision and Pattern Recognition, IEEE CVPR 2005*, San Diego, CA, USA, Jun. 2005.

[158] T. Lambrou and C. Panayiotou, "A survey on routing techniques supporting mobility in sensor networks," in *5th International Conference on Mobile Ad-hoc and Sensor Networks, IEEE MSN 2009*, Wu Yi Mountain, China, Dec. 2009.

Biography

Theofanis P. Lambrou has received his Diploma Degree in Electrical and Computer Engineering from the National Technical University of Athens, Greece in 2004. From 2004 to 2005 he was a Research Assistant at the microelectronics and electronics sensors laboratory at the National Technical University of Athens. Since January 2006, he is a Ph.D. candidate in Electrical and Computer Engineering at the University of Cyprus. His research interests include wireless sensor networks, sensor technologies and signal conditioning, sensor and actuator systems design, cooperative control for distributed robotic systems and motion planning. He is involved in various projects funded by the European Commission and the Research Promotion Foundation of Cyprus that involve path planning algorithms, collaboration between mobile and stationary sensor nodes, sensors design and development. He is a member of the IEEE and the Technical Chamber of Cyprus (ETEK).