FCTUC

Department of Electrical and Computer Engineering

Faculty of Science and Technology

University of Coimbra

RoboCops: A Study of Coordination Algorithms for Autonomous Mobile Robots in Patrolling Missions

David Bina Siassipour Portugal

A Dissertation presented for the degree of

Master of Science in Electrical and Computer Engineering

September, 2009

FCTUC

Department of Electrical and Computer Engineering

Faculty of Science and Technology

University of Coimbra

RoboCops: A Study of Coordination Algorithms for Autonomous Mobile Robots in Patrolling Missions

Supervisor:

Prof. Doutor Rui Rocha

Juries:

Prof. Doutor Manuel Crisóstomo

Prof. Doutor Rui Cortesão

Project report written for the dissertation subject, included in the Electrical and Computer Engineering Course, submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering

September, 2009

Acknowledgements

Several people have been instrumental for the construction, development and conclusion of this project. I would like to thank them for their important and gracious support. These people include not only the academics and colleagues, but also friends and family who have assisted me in different ways during this last year.

First of all, I would like to thank my supervisor, Prof. Rui Rocha, for his invaluable guidance, with whom, I have discussed this thesis in detail ever since the planning phase of the study. Due to his methods in monitoring, organization and support, I can honestly say that I never felt adrift during the development of the project. I would also like to remark the great motivation he has given to me when things did not go as well as expected. It was a great pleasure to work with him and I hopefully expect that we have the opportunity to work again in the near future.

Special thanks to my closest family: My parents, my sister Sara, who moved to Australia recently, and whom is already missed a lot, also my grandmother from Iran, who has been staying with us for the last three months; and also my beautiful girlfriend Vanessa. These are the people, whom by their merely existence, allow me to go on, because of their unconditional love, care and encouragement throughout my life.

My appreciation goes, as well, to my undergraduate and closest friends from Coimbra for their support, friendship and listening skills, even when they had no idea what I was speaking about.

I further would like to mention my laboratory mates, João Pacheco, João Quintas, Manuel Oliveira, Miguel Ferreira, Paulo Freitas and Ricardo Faustino for the pleasant working environment that subsisted during last year and also for their comments and suggestions. Many thanks to my colleagues José Manuel Marinho and David Delgado, for fruitful discussions and appreciated company for snack breaks. I am also greatly indebted to

many other close colleagues during the five years of the course. Being all in the same boat, we formed a team that helped each one of us to find the path to finish our studies. I would like also to express my deepest gratitude to João Monteiro and José Rui Simões, who have helped me resetting the laboratory desktop, which drastically reduced my stress levels. In addition, I acknowledge the contribution of Richard Szabó from Eötvös Loránd University, in Budapest, Hungary for sending me his work in the "Topological navigation of simulated robots using occupancy grid" project.

Along with the above people, I am grateful to my english aunt Julie for the proofread of this document.

Resumo

Hoje em dia, a evolução na área de robótica já permite libertar os seres humanos em bastantes tarefas, principalmente as mais monótonas e repetitivas ou, eventualmente, com carácter de perigosidade. Uma destas tarefas é o patrulhamento de infra-estruturas. O desenvolvimento e concepção de métodos de patrulha através de robôs móveis é uma área científica cujo interesse continua em crescimento.

Esta dissertação foca-se em algoritmos de patrulhamento para equipas de robôs móveis dotados de um mapa do ambiente e da capacidade de se localizarem e de navegarem no mesmo. O principal objectivo é desenvolver e validar um algoritmo inovador, escalável para ambientes de grandes dimensões, patrulhados por um número elevado de agentes, tornando eficiente a sua cooperação, obtendo o máximo desempenho na execução da missão e possuindo robustez face à inoperacionalidade de um subconjunto dos agentes inicialmente disponíveis para a realização da missão.

O algoritmo apresentado baseia-se em partição multi-nível do mapa do ambiente, atribuindo diferentes regiões a cada agente móvel. Cada região corresponde a um sub-grafo extraído da representaçao topológica existente. O algoritmo lida com a tarefa de patrulhamento local de cada robô, tendo sido designado por Multilevel Subgraph Patrolling (MSP) Algorithm.

De forma a assistir no desenvolvimento do algoritmo e a analisar o seu desempenho, construiu-se um simulador que serviu como uma ferramenta essencial do projecto. As suas características são apresentadas ao longo deste documento.

Palavras-Chave: Patrulhamento; coordenação de robôs móveis; mapas topológicos; particionamento; teoria de grafos.

Abstract

Nowadays, the evolution in the robotics field allows to free human beings in a set of tasks, mainly the most monotonous and repetitive or, eventually, with dangerous concerns. One of these tasks is the patrolling task in infrastructures. The development and conception of patrolling methods using mobile robots is a scientific area which has a growing interest. This work focuses on patrolling algorithms for teams of mobile robots endowed with the environment map and the ability for self-location and navigation. The main objective is to develop and validate an original algorithm, scalable to large dimension environments, patrolled by a large number of agents with efficient cooperation, resulting in a high-performance execution of the mission and robustness in the case a fault occurs in a subset of the initially available agents.

The presented algorithm is based on multilevel partitioning of the environment map, assigning different regions to each mobile agent. Each region corresponds to a subgraph extracted from the existing topological representation. The algorithm deals with the local patrolling task assigned for each robot, being named Multilevel Subgraph Patrolling (MSP) Algorithm.

In order to support the algorithm's development and evaluate its performance, a simulator was built, which represented an essential tool during the project. Its properties are presented throughout this document.

Key Words: Patrolling; Mobile Robots Coordination; Topological maps; Partitioning; Graph Theory.

Declaration

The work in this dissertation is based on research carried out at the Mobile Robots Laboratory of ISR (Institute of Systems and Robotics) in Coimbra, Portugal. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Contents

List of Figures

List of Tables

Notation

n_r Number of patrolling robots.

σ Standard Deviation ($\times 10^{-3} s^{-1}$).

$\sum d_T$ Total patrolling distance travelled (m).

[C] Cyclic Algorithm.

[MSP] MSP Algorithm.

F Average point frequency of passage ($\times 10^{-3} s^{-1}$).

Chapter 1

Introduction

This report describes the work that has been done in the "RoboCops: Study of Autonomous Mobile Robots Coordination Algorithms for Patrol Mission" project, which took place in the Mobile Robots Laboratory of ISR (Institute of Systems and Robotics) in Coimbra last year. The main goals of this master degree project is the study of coordination algorithms for teams of cooperative mobile robots in patrolling missions and the corroboration of an innovative, scalable and fault-tolerant algorithm by means of conception and testing an algorithms' simulator, which represents one of the final deliverables of the project.

There were four main work phases. In the beginning, graph theory and its relationship to different patrolling strategies with multi-agents systems, implemented in previous works, was studied in detail. The second phase consisted of the conception of the simulation environment to provide the topological information extracted from the environment map, which is essential for correct robot navigation. Afterwards, the new patrolling strategy, named Multilevel Subgraph Patrolling (MSP) Algorithm, was developed, refined, and validated and, finally, using also the simulator, the algorithm was compared to a classic algorithm from the literature and the results were collected.

In this chapter, an overview of the document is given and the context of the research is specified as well as some outlines for the usage of multi-agents systems.

1.1 Context and Motivation

Advances in the field of robotics have been massive in the last decades. Issues like patrolling, map learning, autonomous navigation, self-location, graph-exploration, cooperative dynamics, obstacle avoidance, pursuit-evasion, surveillance and inspection have become very popular in recent years and represent interesting challenges in many scientific areas like mobile robotics, computer vision, control, real-time systems, networking or artificial intelligence.

In the particular case of infrastructure patrol, which has high utility and impact on society, every position in the environment (or at least the ones that require surveillance) must be regularly visited, assuring a minimum frequency for verifying the existence of intruders or other anomalies. As well as being monotonous and repetitive, this task may also be dangerous, hence an alternative approach for prevention of human beings is to allow technology to assist through the use of multiple mobile robots to pursue the mission, collaborating to guard the grounds from intrusion.

Most of the proposed patrolling algorithms, using teams of mobile robots are normally based on the environment abstraction by using undirected graphs whose vertices represent places in the environment and whose edges represent the connectivity between places. Usually, these algorithms are not generally scalable to large dimension environments or patrolled by a large number of agents. In this context, to develop a scalable and fault-tolerant algorithm with efficient cooperation and high-performance execution of the patrolling task, an extensive study of these techniques and the effort made in the area must take place. Note that it is assumed in this work that the environment is known and robots have the ability to self-locate and navigate within the boundaries of the structures.

1.2 Multi-Robot Systems

There are several advantages of using multi-robot systems in patrolling missions as well as in other applications. In some cases, due to the need of combining multiple tasks and the dynamics of the environment it is only viable to achieve the mission with a multiple distributed autonomous robotic system.

A fully-equipped autonomous mobile robot with sensors of different types may adequately

fulfil the assignment, but it may also prove to be costly and have diminutive fault-tolerance. On the other hand, according to [Roc05], "Cooperative multi-robot systems (MRS) have received significant attention by the robotics community for the past two decades, because their successful deployment have unquestionable social and economical relevance in many application domains". Multi-robot systems are characterized by distributed control, autonomy, greater fault tolerance, and communication. A team of cooperative agents may accomplish the task with better performance than a single robot. Without the help from other robots, a single robot may be vulnerable to hostile environment or enemies, for example in some military actions. In many other applications, an agent may get assistance from other nearby agents during emergencies, such as failures or malfunctions.

Multi-robot systems include the advantage of having many robots being in many places and carrying out diverse tasks at the same time, i.e. space distribution.

When the problem denotes more complexity, sometimes it is useful to divide it in simpler subtasks and assign them to different robots of the team. The decomposition of complex problems linked with effective cooperation is a major advantage of these systems. This feature can be used, for example, in exploration of unknown environments. To increase reliability and robustness, in comparison to a single autonomous mobile robot incorporated with all kind of sensors and abilities, a team of multiple robots may be heterogeneous by having spread resources. For that reason, each unit becomes simpler and, as a consequence, its cost may be reduced. Furthermore, these systems enable redundancy and graceful degradation, remaining functional if some of the agents fail.

Another main motivation for adopting robots is the possibility of reducing the risk to human operators in the face of dangerous exploration missions. Multi-robot systems can ease arduous, tiring and time-consuming tasks like surveillance, infrastructure security or monitoring. In addition, one or more robots could replace humans in dangerous situations like search and rescue operations, which require strong effort by rescuers in a very dangerous scenario that poses many threats to human rescue teams. Replacing people with autonomous robots in these environments provides inestimable benefits.

Also, as reported before, these systems may offer the possibility to relieve human beings from monotonous or repetitive tasks enabling them to be occupied in nobler tasks.

1.3 The approach followed in this dissertation

This work mainly addresses the multi-robot systems' problem of efficiently patrolling a given infrastructure environment. Usually, a blueprint of the environment is available. Based on this representation, we are able to produce a graph-like topological map. Moreover, having this map enables us to establish important places and their connectivity as well as planning routes for the robots.

Clearly, the need to develop a simulator to obtain the topological representation from the initial map and to replicate the robots' motion within the environment arose. Using the same inputs, simulations can provide fast and approximate notions of real-life situations. By analysing simulation results using different environments, it is possible to improve the routing algorithms for robots as well as understanding in which environments the algorithm is better suited. Only when it was felt that the MSP algorithm was complete, a comparison with a previously proposed approach took place.

With the aim of relevantly contributing to the scientific community, the new patrolling algorithm was fully developed and it will be shown that the results prove its efficiency, also that it scales well with large environments and arbitrary high team sizes. On the other hand, it is also confirmed that there is no need of having exaggeratedly equipped agents with sensors or other capabilities to finish the mission and that fact does not diminish the robots' autonomy.

In this dissertation, the followed approach is based on balanced graph partitioning, assigning a patrolling region for each robot. The main purpose of the patrolling strategy is to maximize the frequency of visiting every point of the environment.

The MSP algorithm will be presented in much more detail later on.

1.4 Outline of the document

This document is organized in different sections, each referring to distinct issues related to this dissertation. The first chapter introduces the context and motivation, important advantages of multi-robot systems and some important aspects of the work done throughout the project.

Chapters 2 and 3 review the most relevant research work previously carried out in the

areas of mapping, representations, obstacle avoidance, patrolling, exploration, navigation, surveillance, coordination mechanisms using multi-robot teams and associated problems. One of the aims of this study was to point out advantages and disadvantages of the methods considered to assist the choice of an approach for the developed algorithm.

Chapter 4 deals with acquiring the topological map of the environment using metric representation. The extraction of the graph is crucial for robot navigation.

Chapters 5 and 6 provide detailed information about the MSP algorithm conception. The partitioning phase of the MSP algorithm is presented in chapter 5 and chapter 6 describes the local patrolling phase.

In chapter 7, the simulation results are revealed and the advantages, potential application and properties of the MSP approach are discussed.

To finish, a final chapter sums up the work, provides final conclusions and prospects interesting future directions for this research. The work contributions for this document are presented in a separate references section.

Chapter 2

Mapping and Obstacle Avoidance Overview

In this chapter, a study of important introductory issues is presented. Some preliminary concepts are established to set up the framework to the problems being discussed at a later stage. Firstly, It addresses mapping and environment representation issues and then obstacle avoidance. In [Por09] a more extended state of the art is presented, which also analyses the problem of dealing with intruders. This report is available on the project's CD release.

2.1 Mapping

In navigation tasks, it is normally assumed that the environment is known. On the other hand, exploration is generally related to completely or partially unknown environments. For both cases, obtaining or building representations is one of the key issues to successful completion of the task. In this context, it is indeed very important for autonomous mobile robots to learn and maintain models of the environment.

2.1.1 Representations

According to [Thr98], the two major distinct paradigms produced for mapping indoor environments are grid-based maps and topological maps. Grid-based methods produce accurate metric maps, they are easy to build, represent and maintain, but they often suf-

fer from huge space and time complexity, because of its fine resolution restrictions, which results in memory problems and also harder efficient planning, as well as navigation in large scale infrastructures.

Topological maps, on the other hand, produce graph-like maps that can be used much more efficiently. They are simpler, permit efficient planning and do not require accurate determination of the robot's position. In these maps, vertices correspond to important places or landmarks, which are connected by edges that represent the paths between them. However, the inaccuracy of the method makes it harder for recognition and maintaining consistency in large scale environments, particularly if sensory information is ambiguous, which may result in perceptual aliasing, i.e., difficulties in recognizing similar places that look alike.

In [Thr98], the author presents a hybrid approach that gains advantage from both methods. This method is based on partitioning grid-based metric maps into coherent regions and generation of compact topological maps on top of the grid representation. The number of topological entities is significantly smaller than the number of cells in the grid representation. This approach has exclusive advantages such as accuracy, consistency and efficiency that cannot be found for each approach alone.

Another important work in this area is [Sza04] which presents a method for building a topological navigation graph on the top of an occupancy grid.

An occupancy grid consists of a metric representation method wherein each cell of the grid contains a probability value which indicates whether the related location is free space or part of an obstacle. It is relatively simple to implement and has an iterative nature.

To obtain the topological navigation graph from the occupancy grid, a process of skeletonization (creating vertices), followed by chaining the skeleton to form edges, graph optimization and navigation using the graph, is conducted.

In this project, a topological representation of the map is assumed, given that the developed patrolling algorithm was assigned to be based on a graph-based representation of the environment.

2.1.2 Applications using topological maps

A set of works deal with mobile robots navigation using topological maps like [ZFV94] that focuses on constraints on navigation by mobile robots when using topological representations. The main limitations found were: "handling of very inaccurate position (and orientation) information as well as implicit modelling of complex kinematics during an adaptation phase". Since there were no distance measuring sensors, it was only possible to detect neighbour vertices in the topological graph by transversing their connections and not simply by sensing or recognizing them. Despite all these constraints, the path planning methods used in the work produced adequate results. Also, topological maps have the advantage of being often very flexible and offer a lot of alternative routes, allowing navigation replanning to be done more quickly.

As well as navigation, map validation and self-location using topological maps is an important issue extensively studied in [DJM+93] and [DJM+97]. In these works, the robot is given an input map and its current position and orientation with respect to the map, and by exploring the world systematically, using distinct markers that can be dropped and picked up for recognition motives, determine whether the map is correct. An exploration algorithm that works by building up a known subgraph of the environment and incrementally adding unknown edges and vertices is also involved. The outgoing edges lead to unknown places that must be explored. The self-location problem consists of giving only the map of the environment and determining the position and orientation of the robot.

2.1.3 Map Building

Vast work has already been made on mapping unknown environments using autonomous mobile robots or multi-robot teams. The progressive accumulation of positional errors makes the building of maps based on metric systems much more complex. Therefore, map building is often based on the topological approach.

In [Alt03], a system was built to send information to a mobile robot about the existence of corridors and rooms. This enables appropriate navigation of the robot on the environment and the possibility of building its own topological map of the area with simple algorithms and low computational complexity in order to autonomously navigate throughout the

whole environment. This mechanism establishes the bridge between two main geometrical representations used by mobile robot navigation systems: the representation of individual objects close to the robot and maps reflecting the large-scale structure of the area. The topological maps are represented by vertices and edges. For this case (indoor navigation), the vertices are placed in the centre of corridors and between the door posts. Vertices in rooms are positioned in places that are important for the navigational task. As usual, the edges define how the vertices are connected. The author endorses that these types of maps are very compact representations and are usually easy to construct due to their low complexity. Another advantage is that they only include information which barely changes over time (rooms or corridors). Hence, they are still valid after, for example, refurnishing an office space. Remarkably, motivated by the general structure of an indoor environment, five navigation behaviours were defined: go to, obstacle avoidance, corridor following, wall avoidance and door passing.

In [DJM+98], the problem of exploring and building topological maps of unknown environments using mobile-robot teams is investigated and it was demonstrated that a group of identical robots, each equipped with its identity marker, can explore and map an unknown graph-like environment. This was possible by coordinating the robots that depart from a common vertex to simultaneously explore different regions of the environment, and by meeting in a commonly known location, after an agreed time interval, to merge their individual partial world models, creating a shared merged map and re-partioning the unknown portions of the common map to repeat the process until the environment is fully explored.

This approach has to deal with several problems like the rendezvous and the merging problem. The first one consists of formalizing the characteristics of the meeting between robots considering their limited communication over long distances. The second problem is consistently merging different portions of the global environment map, because it requires a mechanism to avoid ambiguities in the representations collected by each robot. Note that markers are used to avoid redundancies in the exploration task. Markers solve position uncertainties, enabling the robots to identify already explored areas.

A very similar work is described in [SUP+07], in which there is also merging of maps but, in this case, this method is not based on features, but rather on occupancy grids.

The maps built are very precise and the exploration strategy is to move to the closest location where the robot can gather information about a cell that has not been sufficiently explored. Firstly, the robot explores the area around itself and then seeks a minimum distance to the next point. The merging task is simpler than in the previous work, because it is easier to identify overlaps between maps using grids and the produced maps are more accurate. On the other hand, this approach involves much more computational complexity.

2.2 Obstacle Avoidance

Another important issue related to adequate motion of mobile robots within an area is obstacle avoidance. To overcome this problem, robots are normally equipped with sensors that can be based on computer vision, distance measures or other types. The main goal is to avoid hitting walls and objects of the structure and collisions with other mobile robots. [BK91] produced a new method that detects unknown obstacles and avoids collisions in real-time while simultaneously steering the mobile robot toward the target. It was called the vector field histogram (VFH). This method has significant advantages when compared to earlier obstacle methods.

Basically, the VFH method uses a two-dimensional Cartesian histogram grid as a world model, whose values are updated continuously according to information from on-board range sensors. Subsequently, the histogram grid is transformed into a one-dimensional polar histogram constructed around the robot's momentary location, whose sections contain a value representing the polar obstacle density in that direction. Next, the algorithm selects the most suitable sector with a low polar obstacle density to steer the robot to the corresponding direction. The authors conclude that the algorithm is computationally efficient, robust and insensitive to misreading, allowing continuous and fast motion of the mobile robot without stopping for obstacles. Also, the robot is able to pass through narrow openings or traverse narrow corridors without oscillations.

On the other hand, [DK07] centres its research on decentralized agents' navigation with proper mechanisms to avoid collisions between them, using limited sensing capabilities. The goal is to derive a set of control laws that drives the team of agents from any initial

configuration to a desired goal configuration avoiding collisions at the same time. In this work, each agent is aware of the existence of all agents in the workspace. However, in order to avoid collisions, they only have to know the precise positions of agents within its sensing circle at each time instant. It is unnecessary to know the desired destinations of the other agents. According to the positions of other agents in each robot sensing zone, the algorithm chooses a specified direction for the robot, taking into account not only collision avoidance but also the goal target of the robot.

2.3 Summary

Now that the issues of mapping, representations and obstacle avoidance were stated, different patrolling strategies that implicitly incorporate these subjects can be studied in more detail in the next chapter.

In the case of the MSP strategy presented later on in this dissertation, the environment is known and maps often result from robots sensor data. As for representation, the topological approach is used. Finally, an obstacle avoidance mechanism was also computed for simulations and will be presented at a later stage.

Chapter 3

Patrolling Algorithms Background

This chapter presents a survey of patrolling, surveillance, navigation and exploration strategies already implemented in the literature.
In the last decades, related algorithms for multiple robot teams have piqued the interest of the robotics community, becoming a remarkable growing area. One of the main reasons contributing to this fact is the variety of approaches that these algorithms can comprise. Many authors contributed with studies that involve many different strategies to solve these problems.
In this section, some of the background work related to different strategies is presented. The reader will note that it is not imperative to follow just one of these strategies. In fact, most of them result from mixed strategies, though more evident characteristics related to the strategy being discussed are focused in each section.

3.1 Cyclic Strategies

Description: Deterministic approaches that aim to maximize visiting frequency of all the points in a known workspace. Robots normally follow the same path over and over again.

The problem of generating patrolling paths for a team of mobile robots within a certain environment following a frequency optimization criterion is considered in [EAK07]. The area patrol algorithm developed guarantees that each point in the target area is cov-

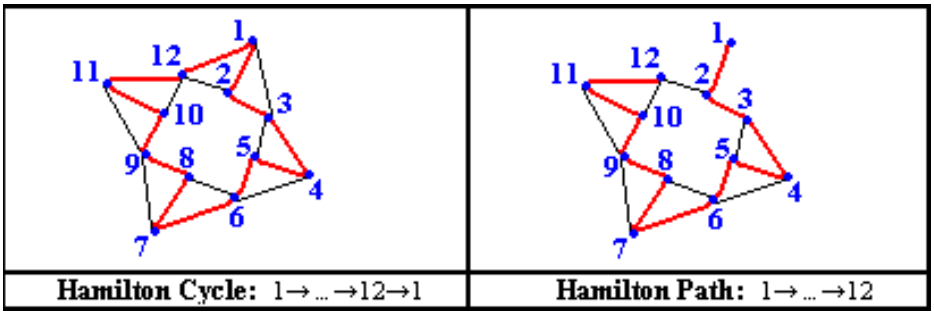


Figure 3.1: Examples of a Hamilton Cycle and Path.

ered at the same optimal frequency. This is possible by computing cyclic patrol paths that visit all points in the target area, i.e. Hamilton cycles. Agents are uniformly distributed along this path and they follow the same patrol route. One of the key aspects of this strategy is that it is totally decentralized and robust in the sense that it is independent of the number of robots. Uniform frequency of the patrolling task is achieved as long as there is, at least, one robot working properly. Logically, the visit frequency grows when the number of robots rises. A possible disadvantage of this approach is its deterministic nature. An intelligent intruder that apprehends the patrolling scheme may take advantage of the idle time between passages of robots in some points of the area.

As seen before, one of the main concerns of the previously proposed strategy, possibly the most important one, is the computation of Hamilton cycles. These consist of closed paths that contain every vertex of a graph, accordingly to [BM76]. Figure 3.1 shows an example of a Hamilton cycle and also a Hamilton path.

An extensive study of procedures to determine if there are Hamilton cycles (or Hamilton circuits) in a given graph and situate them is presented in [Rub74]. In this work, partial paths are created and some deductions are made in order to determine if these paths correspond to sections of a Hamilton circuit. The process is iterative and it starts with short paths, testing its admissibility, adding vertices and repeating it until every vertex is included in the path.

3.2 Partitioning Strategies

Description: Approaches that use segmentation of the environment and assign regions to each agent in order to navigate or explore.

In [WSB08], the problem of distributing a team of mobile agents in a partial unknown area and assign them destination targets, minimizing the overall exploration time is studied. A way of distributing the agents in the environment, taking into account its structure, is proposed.

The already explored space is partitioned into segments. Each of the segments is assigned to a different agent. Instead of only considering as target locations frontiers between unknown and explored areas (which is a usual approach for the exploring task), robots are sent to the individual segments with the task of exploring the corresponding areas. Based on this segmentation, the robots are distributed over the environment more effectively, which leads to a reduction of redundant work and the avoidance of interference between robots. It is proved that the overall time of exploration is significantly reduced, accordingly to experiments in a simulator and real world experiments.

A slightly different approach is presented in [AKK08] for perimeter patrolling around a closed area using multiple robots. The existence of an adversary attempting to penetrate into the area is assumed. Unlike cyclic strategies, the main goal here is to build a non-deterministic patrolling algorithm. In this sense, the perimeter is divided into segments, assigning an agent for each region per time cycle. The robots' motion is characterized by a given probability and the enemy knowledge/prediction of the patrolling scheme is limited. Three distinct robotic motion models are considered and tested. It is assumed that the intruder observes the patrolling scheme and decides to infiltrate through a place with the lowest probability for its detection. An optimal polynomial-time algorithm that maximizes that probability is described and its efficiency is proved when using the three considered motion models.

3.3 Contamination-Based Strategies

Description: A graph is associated with the given environment. Robots are equipped with range sensors. Edges and vertices can be cleared or contaminated. In the event of being cleared, they have no possible intruders. If contaminated, they can potentially host one or more intruders. Mobile robots must cooperate in order to clear the entire graph.

An important contribution based on this strategy is the GRAPH-CLEAR problem described in [KC07]. In this paper, vertices are associated with rooms, and edges to connections between different rooms. A strategy is defined to enable mobile robots to gradually clear the graph. Note that a clear vertex can become contaminated again if there is an edge connecting it to another contaminated vertex. Edge recontamination is also possible. Two fundamental operations are defined and must be used by mobile robots to finish the task: blocking and clearing. Blocking disallows recontamination by applying actions on edges, consequently clearing the edges. Clearing ensures that all intruders are detected in a vertex and when it is applied the vertex becomes clear.

GRAPH-CLEAR deals with the problem of determining a strategy to clear a given graph with minimal cost, assuming weighted vertices (number of agents needed to perform a clearing action on those vertices) and weighted edges (number of agents necessary to perform a blocking action on that edges). Consequently, it is necessary to ask the minimum number of robots needed to detect all possible intruders, given that weighted graphs are being used. Also, it is assumed that the robots have limited sensory capabilities.

The nature of the problem was deeply investigated and an algorithm for a special case, in which the graph consists of a tree, is also presented. The method was correctly tested in simple simulations and in some practical scenarios, though the results of this paper serve essentially as theoretical foundation.

An extension of this work is [KC08] wherein the authors propose an improved algorithm for the same problem, by presenting criteria for optimal graph partition, subsequently decreasing the number or robots necessary for the surveillance task. In a real robotic application, this fact means the reduction of costs in implementing the proposed solution. Moreover, in [KC08a] the authors aim to take "a first step to close the loop between the problem's graph-based theoretical formulation and practical scenarios" by algorithmically extracting surveillance graphs from occupancy grid maps. The approach is based on Voronoi Diagrams of the environment obtained through the occupancy grid maps. The proposed algorithm is fast, robust and includes certain opportunistic operations that modify the graph, reducing the navigation strategy complexity and enable solutions requiring fewer robots.

3.4 Markers-Based Strategies

Description: Mobile-robot team strategies that involve tags or indicators, using the environment itself as a way of communicating in order to accomplish the exploration task.

A multi-robot dynamic coverage problem is considered in [BS02], which requires all areas of free space in the environment to be covered in the shortest possible time. A metric based on frequency of every-point coverage and a decentralized algorithm that uses available local information about the workspace are proposed to solve this problem.
Robots explore the environment with lack of global information. Neither maps nor localization in a shared frame of reference are used. However, based on local criteria, they drop communication beacons to identify earlier visited areas. These are equipped with a small processor, a radio of limited range and a compass and proceed as local semaphores that allow or block the progression of other robots to the corresponding area. Using this method, robots can decide their motion strategy locally and improve dynamic coverage. The frequency coverage metric calculates how often points in the environment are visited and the essential intent of the decentralized exploration algorithm is: "A robot explores as long as there are open regions left. If all the regions are explored, then the robot picks the direction which was least recently explored."
Furthermore, a pattern of patrolling behaviour becomes evident as robots tend to revisit points over time, because they are not allowed to settle.
This paper does not address target tracking. Instead, it focuses on displacing multiple robots in a planar circumscribed workspace and maximizing their sensor coverage over time.
[Mia07] employs a similar strategy. In this work, the problem of controlling a group of robots in order to cooperatively solve exploration tasks using algorithms inspired on ant colonies is studied. Robots must address these tasks using simple and individual control strategies. The system is based on the utilization of markers that can be placed by robots to identify certain positions in the environment inspired by ants' behaviour that deposit and follow trails of pheromone to solve optimization problems.
The presented algorithm chooses the direction to follow by analyzing first if there are obstacles in front of the agent and then investigating pheromone quantity in each antenna.

After making the decision, agents drop a marker and follow that direction.
The global behaviour that emerges is one where as more ants follow a trail, the more it becomes attractive to chase. There is a positive feedback, in which the probability of an ant choosing a path increases with the number of those that have already chosen the same path.
For robotics, the most attractive characteristic in ant colonies is the interaction capacity of simple individuals towards executing complex tasks without global leadership and with local communication. Robots employ some of these relevant behavioural characteristics and include other characteristics that do not exist in biological ants. This approach has clear advantages like its decentralized nature, being scalable and fault-tolerant.
The deposited markers are virtual. Each agent contains in its internal memory a table with the localization of markers placed by every agent, which is updated continuously. There is no explicit communication between the individuals; they only communicate indirectly using markers.

3.5 Communication-Based Decentralized Strategies

Description: Strategies that involve, at least, limited direct communication between agents while performing a specified multi-robot mission. This is usually used when the environment is partially or totally unknown.

Navigation strategies for cooperative multi-robot teams with local communication capabilities in unknown environments are discussed in [SA03]. Robots share information locally related to their explored areas and mutually assist each other to achieve their goals. Since they have partial knowledge (or perhaps no knowledge at all) about the environment, they must rely on local navigation algorithms to find their way to the target. In this work, instead of establishing motion strategies or spatial distributions to allow robots to help each other, two very interesting strategies are formulated, which enable them to do their best in helping each other whenever they have the chance.
The first strategy is called “Goal-Sharing”. Robots share their targets with each other and mark them with a unique identifier. Each robot needs not only to move in the envi-

ronment towards its goal but also to remember many different goals from other robots. When a robot communicates its goal to another one, the second robot estimates its best-know distance to that target and informs the first one. That path is compared to the one that it has on its local memory and it is replaced in the case of being shorter. Also, two robots share information about other robots and their targets, attending only to the new information in case it guarantees a shorter path to the goal than the value they possess in their memory.

The second strategy is called "State-Sharing" and is much simpler. If they are taking too much time to find their goals, robots switch their state from 'not in trouble' to 'in trouble' and share it with others. When this happens, a robot 'in trouble' is attracted to a close team mate that is 'not in trouble' and heads towards him until it realizes that it is no longer 'in trouble' or it is eventually able to perceive its own goal.

A simulation system with distributed autonomous robots for search and rescue operations is described in [DA99]. The system is used to test different team control schemes and organization. Simulated robots are capable of navigating independently and communicating with a simple message protocol, that is based on three possible states, related to the exploration assignment. This process involves sending and capturing each other's messages with the assistance of limited range sensors. Also, the simulator provides sensory information, tracks the visited regions and plans robots' actions, evaluating its performance afterwards by measuring the percentage of rooms entered in the environment or, in case robots entered in all rooms, the time it took to complete the total coverage.

Experiments show that, generally, robot team performance improves as team size increases. However, it tends to stabilize after team sizes reach certain threshold values that depend strongly on the environment's complexity. Also, as team size increases, team interaction becomes more evident by presenting dispersion behaviour among team members.

3.6 Centralized Cooperation-Based Strategies

Description: Strategies that involve previous centralized motion planning in order to correctly complete a cooperative multi-robot team assignment.

Utilization techniques of collaborative teams of robots for infrastructure security applications are described in [GPM07]. A distributed sensors algorithm for robots' localization and 3D mapping is presented and, also, a motion planning algorithm for multi-robots considering the patrol scenario and response to invasive scenarios is discussed.
The proposed approaches overcome positioning systems, map-building and human robots' trajectory planning limitations.
In order to map the environment, a first stage is started by testing the navigation of robots in the area using the distributed sensors approach. After capturing all the required data from the environment, robots may operate in two different modes: Patrol Mode or Threat Response Scenario Mode.
The motion planning algorithm is executed to select efficient patrol patterns taking the environment into account and each robot becomes responsible for its patrol region. Also, during patrolling, robots may update their 3D maps to incorporate possible changes in the environment. If an intruder is detected, some robots will switch their operation mode to Threat Response Scenario and the algorithm is run to successfully respond to the threat, guaranteeing that robots reach the evader in the quickest possible way.
While some robots respond to threats, others will carry on with the patrol task and replan their trajectories to compensate for those that switched their operation mode.
[LS08] presents an approach for the Path Clearance problem also using a centralized planning strategy. In this problem, a robot tries to reach a given destination goal as quick as possible without being detected by enemies, which are localized in distinct points of the environment. The robot does not know the precise location of the enemies, it just has a list of possible locations, and can go through them if it does not sense the presence of enemy units, otherwise it should take a detour.
The problem is investigated by using auxiliary robots endowed with range sensors to help the main robot. A scalable and efficient approach for the centralized planning problem is proposed. The outline of the approach is sending the scouts to possible enemy locations to sense enemies and avoid as much as possible deviations on the main robot planned trajectory. The plan aims to minimize the expected time for the robot to reach its goal, however the planning problem is hard due to the environments' uncertainties and needs to

reason about possible outcomes of sensing and generate an alternative policy that dictates which path the robot should take as a function of the outcome of each sensing.
Whilst it requires centralized planning, the proposed approach is simple, efficient, computationally cheap, scales very well to large environments and large number of scout robots and it also can be used with heterogeneous teams of scouts.

3.7 Other Related Works

Description: Approaches that, in addition to complete multi-robot team missions, use somewhat different strategies or approaches from those previously considered.

A very interesting work which presents an approach that is concerned with robots' safety is discussed in [SYX04]. A distributed model using multiple robots for exploration in unknown environments is discussed, where the concept of risk is introduced and its impact on the efficiency and behaviour of robots is observed.
The main objective is to maximize the exploration efficiency and reduce security risks for robots by designing a robust, efficient, exploration strategy for each robot to achieve a certain degree of safety during the exploration process. Different risk tolerances are attributed to different robots.
New coordination strategies may be created in order to increase the safety of the exploration team. Each robot is equipped with sensing, localization, mapping and communication capabilities and they work asynchronously. At any time, a robot is in one of the following three states: sensing and mapping, bidding or travelling.
This is considered to be an innovative work, since the safety of robots does not normally receive much attention from the research community.
Results show that the robots exhibit certain clustering behaviour when safety is concerned and tend to move apart when no safety is concerned. Also, by introducing the safety, or the risk concept, reliability and robustness can be improved from a protective point of view.
Another distinct work is presented in [Kon03], where the coordination of large scale teams of robots (typically over 100) for indoor environment exploration, map and surveillance

is examined. Each robot is able to self-locate in a map and communicate with nearby fellows. They must work as a team and accomplish a patrolling mission with the possibility of confrontation. Limited autonomy, size, energetic power and complexity in small robots may prove to be difficult restrictions to tackle in terms of communication and coordination levels.

This paper offers one contribution by discussing distributed robot architecture with adaptive collaborative behaviour, scalable, fault-tolerant and efficient which incorporates optimal distribution, by means of map building, communication, monitoring, robot data updates and behaviour prediction.

In terms of communication, messages are exchanged directly between pair members of the team or may be forwarded through other members to extend the range. Also, the approach dynamically assigns tasks and roles to robots, avoiding dependency on a central server. Pair-wise relations between robots enable efficient coordination. Each pair of robots can have four different interactions: none, hypothesis generation (they can communicate without knowing their relative positions), hypothesis verification (they can communicate and verify a relative location hypothesis) and coordination (share maps and perform coordinated exploration).

[Min07] describes a survey on swarm robotics, which represents a group of autonomous robots that interact directly or indirectly with each other to cooperatively fulfill an assignment. These robots are assumed to be homogeneous and very simple. Research and comparison of six different swarm robotics algorithms intended for controlling of large-scale multi-robot systems is discussed. These algorithms are used in two different projects: “The iRobot Swarm” housed at the Massachusetts Institute of Technology and “Swarm-bots” based out of the Université Libre de Bruxelles and are based on the idea that complex macro-level behaviours can emerge from simple local interactions between agents.

Swarm robotics algorithms are typically scalable, fault-tolerant, robust and efficient. Also, the author concludes that markers-based algorithms for swarm robotics research, like Ant Colony Optimization (ACO), are highly unfavourable and should be avoided.

The paper describes the swarm robotics algorithms used in both projects and specifies which characteristics should be incorporated in these kinds of algorithms, like simple

and elegant individual behaviour of each model, scalability, decentralization and local interaction.

3.8 Analysis and comparative studies between different strategies

Description: Works or papers that present comparison of patrolling strategies, pointing out advantages, disadvantages, properties and applications of each one.

[ARS+04] studies and compares existent patrolling approaches using multiple robots: heuristic agents, negotiation mechanisms, reinforcement learning techniques, graph-theory based techniques and others. The work's objective is to serve as a benchmark and guideline for new multi-agent patrolling approaches and improving the current ones.
The authors mention that "Patrolling is a complex multi-agent task, which usually requires agents to coordinate their decision-making, in order to achieve optimal performance of the group as a whole" and they believe multi-agent patrolling approaches are based on three techniques: Operations Research Algorithms, Non-Learning Multi-Agent Systems and Multi-Agent Learning. Comparisons are made by evaluating the approaches' performance using different topological maps, where vertices correspond to specific locations and edges to possible paths. Benefits and disadvantages of each one are specified. We observe that the best strategy depends on the structure of the environment modelled as a graph and the agents' population size.
Generally, it was concluded that the single-cycle approach, based on the Travelling Salesman Problem, has the best performance for most cases. This can be explained by its disciplined coordination scheme, which is very effective. However, this architecture will have problems in dynamic environments, enormous graphs and patrolling priorities assigned regions, due to its predefined and fixed nature. On the other hand, pioneer agents approaches studied generally have the worst performances. These approaches are based on several multi-agent architectures varying parameters. Firstly, agents moving randomly achieved very bad results. Secondly, agents with no communication ability, whose strategies consisted of moving towards the vertex with the highest idleness, performed nearly

as well as the most complex pioneer algorithm implemented. In general, the Heuristic Agents and Reinforcement Learning techniques considered have the second best performance, followed by the considered Negotiation Mechanisms techniques.

Another study is carried in [Che04], wherein diverse patrolling strategy classes are described and compared, focusing mostly on cyclic strategies and partitioning strategies.

A good strategy is considered to be the one that minimizes the time lag between two passages to the same place and for all places. The author makes reference to the fact that very simple strategies with nearly no communication ability can achieve impressive results and also, an approach that is based on partitioning the patrolling area into regions, assigning for each region a patrolling agent, can also work well.

This paper aims to answer some main questions, such as whether the existing algorithms generate optimal strategies, whether there are efficient algorithms generating near-optimal strategies, and how good partitioning and cyclic algorithms are, having agents following the same fixed path with uniform distribution.

Both strategies are proposed and examined in this paper. The main conclusion presented is that cyclic and partitioning strategies have generally good performance; the first one is better suited for graphs that are highly connected or have large closed paths and the second one is better when graphs have long corridors separating regions.

In order to conclude and sum up the most relevant properties of the strategies considered in this chapter, the following table was built.

Table 3.1: Pros and Cons of the considered strategies.

Strategies	Advantages	Disadvantages
Cyclic	+ Simple, effective, scalable, decentralized, fault-tolerant and robust + Optimizes visit frequency of all points for any number of robots + No communication or sensing other agents needed + Appropriate for patrolling operations	- Predefined: easier for intrusions - Problems with dynamic environments, enormous graphs, regions priority schemes and task allocation

Partitioning	+ Reduces redundant work and interference between robots + Better for larger graphs and intrusion prevention (when the robot path is not deterministic in its region) than cyclic strategies + Achieves effective and balanced distribution + No communication or sensing other agents needed + Facilitates topological localization + Appropriate for patrolling and exploration in partially known environments	- Not optimal nor as robust as cyclic strategies - Less fault-tolerant: needs redefinition of paths when agents fail
Contamination-Based	+ Highly effective for intrusion detection + Robots' number can be minimized using these strategies + Only needs limited sensorial capability for effective detection of intruders + Can cope with dynamic, large environments and heterogeneous agents + Possible to merge with partitioning approaches + Appropriate for high security building surveillance.	- Complex: e.g. GRAPH CLEAR is NP-complete. Solutions are based on restrictions of the problem - The minimum number of robots needed depends exclusively on the complexity of the environment - No fault-tolerance and limited robustness

Markers-Based	+ Decentralized, scalable and fault-tolerant + Unnecessary to map the environment: decisions are local + No direct communication between agents is involved + Highly effective: distributed coverage behaviour emerges from local interactions + Appropriate for exploration operations and dynamic environments	- Difficult to implement in real world, due to its interaction with the environment - Poor performance when using small number of robots - Depends too much on equipment like sensors and markers - Expensive (uses varied sensors and may use complex markers)
Communication-Based	+ Often decentralized, scalable and fault-tolerant + Robots may help each other by sharing the information they have obtained + Copes with dynamic and complex environments with good overall performance + Enables task allocation and provides team organization and coordination + Appropriate for search, rescue and exploration operations	- Depends too much on communication, local memory, sensing and mapping - Expensive (because of complexity of each robot) - Poor performance when using small number of robots
Centralized Cooperation-Based	+ Effective and scalable + Enables efficient agents' navigation according to the specified environment + Cooperative robots may compensate each other in different situations + Can cope with dynamic, large environments and heterogeneous agents + Appropriate for exploration and high security applications	- Centralized planning nature (depends on a central entity) - Cooperation mechanisms are not normally simple, especially when they involve large number of robots - Replanning is necessary when faults occur

3.9 The MSP approach

After studying distinct approaches for the patrolling problem and considering their advantages and disadvantages, a hybrid strategy based on graph partitioning and cyclic algorithms began to grow. The idea of the new approach is to obtain multiple subgraphs by partitioning the graph, and assign them to each mobile agent for local patrolling. The local patrolling task will be mostly based on cyclic strategies. In this context, the new approach will benefit from the advantages of both strategies, namely simplicity, effectiveness, scalability, robustness, distribution, with good overall visiting frequency, non-redundant, without the need for communication between agents, with good performance in larger graphs and intrusion prevention and with the ability to allow fault-tolerance mechanisms.

3.10 Summary

To be able to develop the proposed approach, it is necessary to obtain the representative graph of the environment from the metric map available. This process is described in the next chapter. Also, the two subsequent chapters explain the MSP approach. Figure 3.2 sums up all the required phases to obtain a patrolling route from an initial metric representation of the environment.

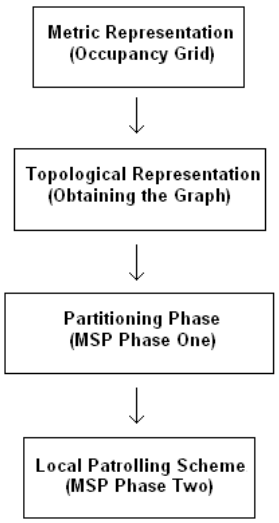


Figure 3.2: Required phases to obtain a patrolling scheme from an initial metric representation of the environment.

Chapter 4

Acquiring the Topological Map

The first stage of this project consisted of extracting the global environment graph and relevant topological information from the available metric representation.
The method used to extract the topological representation of the environment is similar to the one used by [Sza04], mentioned on chapter 2. The method's input is a greyscale occupancy grid, which is a metric representation of the environment. Then, the skeleton computation takes place. After that, all vertices are identified and the topological information is extracted. The final desired output is a navigation graph built on top of the original occupancy grid.

4.1 EVG-THIN: Obtaining the Skeleton

A tool named EVG-THIN developed by Patrick Beeson [BJK05] was used to acquire skeletons from occupancy grids, in this project. EVG stands for Extended Voronoi Graph and THIN accounts for the pixel-based "thinning" algorithm that finds skeletons of bitmaps. The thinning approach is a fast approximation of the Voronoi diagram. Its code was written to be applied in real-time to occupancy grids, where cells are either occupied, free, or unknown, but it also works on greyscale bitmap images for other domains, which means that any greyscale representation will be treated like a standard occupancy grid. The application runs in any Linux console with some important parameters that are described in table 4.1. Also, a skeleton generation is presented in figure 4.1.
Note that this program is released under the GNU General Public License (GPL), which

is intended to guarantee your freedom to share and change free software.

Command	Description
min-unknown [N]	The minimum greyscale value (1-254) of unknown cells. Occupied cells are 0-(N-1).
max-unknown [M]	The maximum greyscale value (1-254) of unknown cells. Free cells are (M+1)-255.
pruning [0|1]	Turns pruning on or off. Pruning removes all "branches" of the skeleton except those that meet one of these conditions: 1) The branch touches the edge of the grid. 2) The branch touches unknown cells.
min-distance [R]	Bleeds obstacles by R cells before calculating skeleton. This removes branches that come too close to obstacles.
max-distance [S]	If the skeleton exceeds the S cells from the nearest occupied cells, it switches to following the occupied cells S away.
robot_loc [X Y]	This location is used to select which skeleton is valid, given complex images with multiple, disjoint skeletons. By default, the "robot" is located at the centre of the image.
robot-close [0|1]	The robot location (see above) is used to choose which skeleton is valid (if multiple exist). This is done by Euclidean distance between the robot's location at the skeletal points. This option turns off the checking mechanism except for points where the robot's distance is within the distance of the skeletal point to its closest obstacle.

Table 4.1: EVG-THIN parameters.

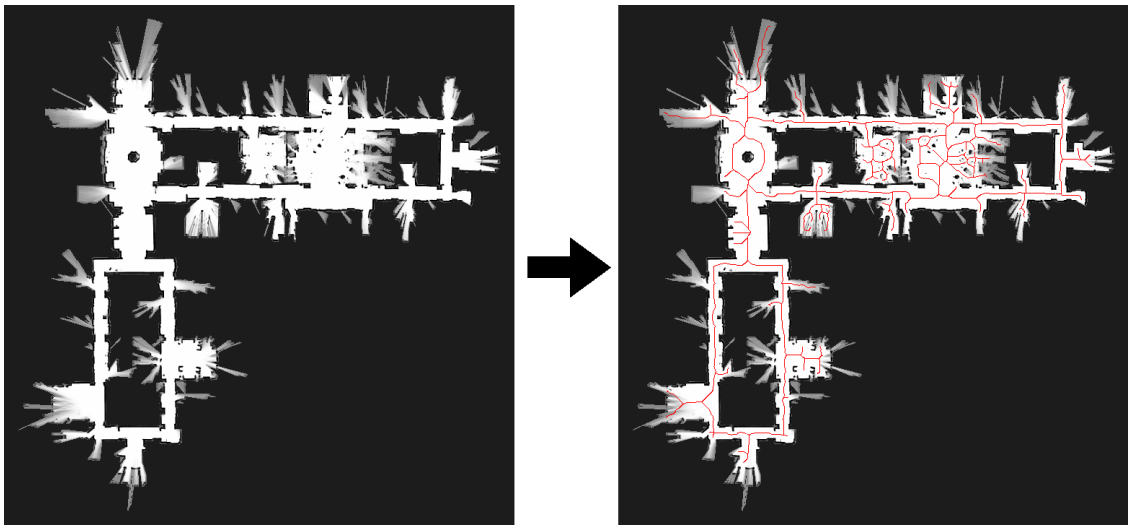


Figure 4.1: Skeleton obtained with the EVG-THIN application.

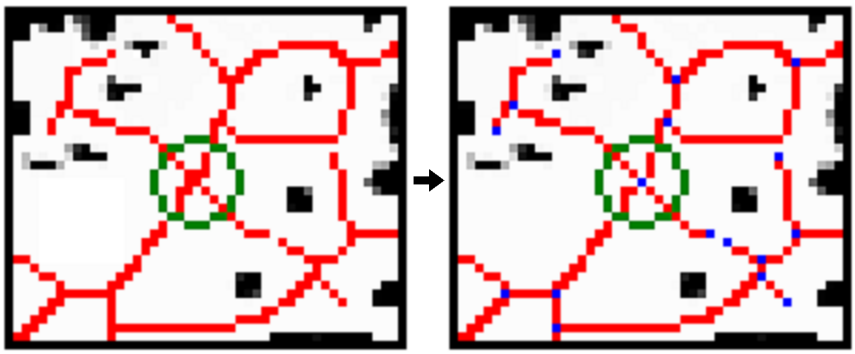


Figure 4.2: Cluster detection and correction.

4.2 Graph and Topological Information

Having the skeleton, the graph is obtained by extracting all vertices and computing the topological information (edges, connectivity, etc.). The identification of vertices was done by image processing on top of the previously generated skeleton. Vertices result from the intersection of skeleton lines and also dead-ends. Additionally, sometimes the skeleton representation given by the EVG-THIN algorithm produces clusters. A protective function was developed to detect these clusters and correct them by using a pixel shifting strategy, avoiding future graph data corruption. An example of a cluster and its correction is shown in figure 4.2. Note that the vertices are displayed with blue pixels on the right side image.

At this time, having identified each vertex and its coordinates, it is necessary to compute the rest of the topological information: their number of neighbour vertices, their cost to each neighbour (computed by evaluating the distance between them in pixel units) and their neighbour directions. With this information, edges can be defined and graphs are created, establishing the framework for agents' navigation.

4.3 Simulator: Incorporation of methods to extract topological maps

As previously mentioned, a patrolling simulator was created. The programming language used was C++ and the Graphical User Interface (GUI) was generated using QT Open Source Edition 4.4.3, which is also based on C++ Language. Both the simulator source

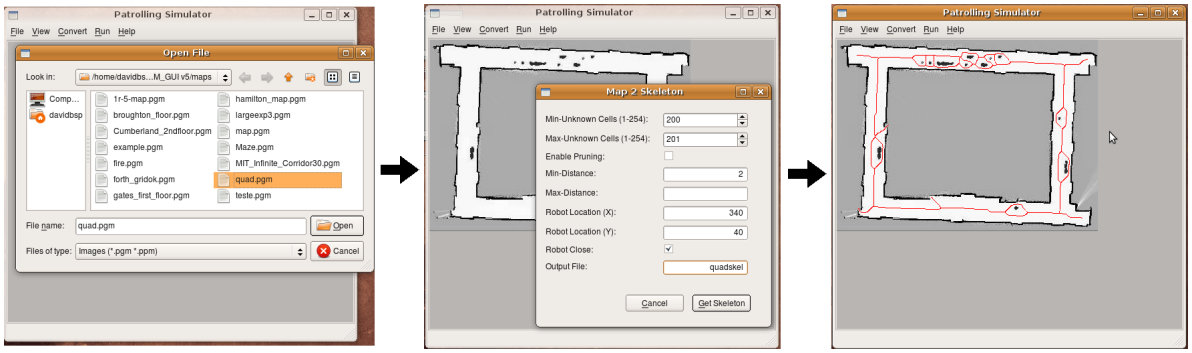


Figure 4.3: EVG-THIN incorporated in the simulator.

code and the reference manual with the main data structures, functions and variables used in the progress of the simulator are included in the project's CD release.

The produced simulator in this work has a very important role in developing and testing the patrolling algorithm. However, it is assumed that a topological representation of the environment is available, which is modelled as an undirected graph whose vertices represent places and whose edges represent connectivity between places. So, the methods described previously to obtain the skeleton and the topological information had to be incorporated in the simulator.

For the EVG-THIN, which is a command line application, a graphical interface was created to generate map skeletons, as seen in figure 4.3. The method described in the previous section to obtain the graph from the skeleton was also successfully incorporated as figure 4.4 portrays.

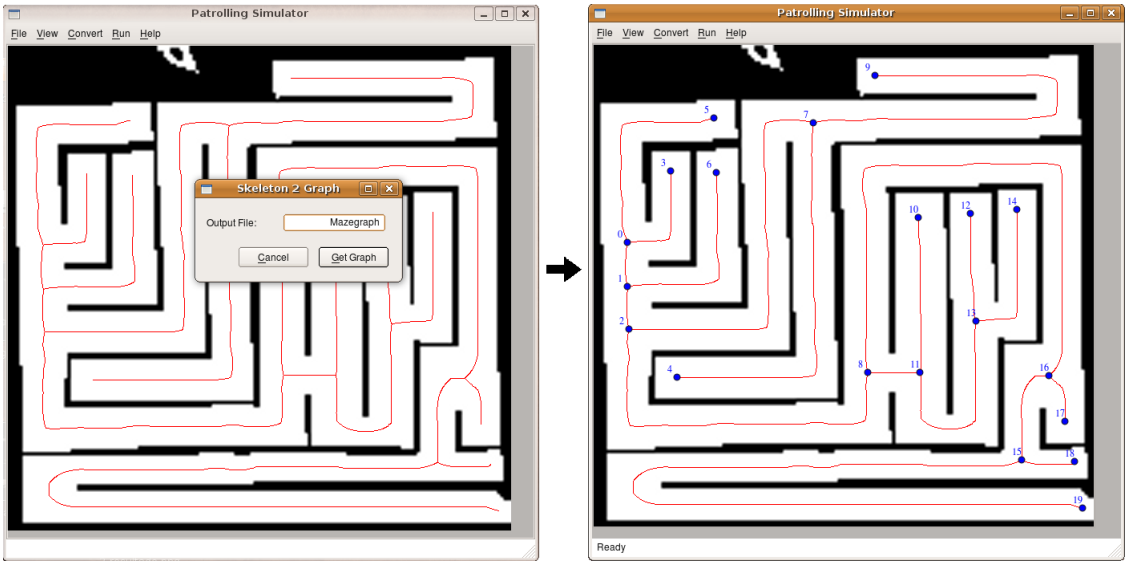


Figure 4.4: Obtaining the Graph from the Skeleton using the simulator. Note that the skeleton was correctly computed even though the map does not represent a conventional occupancy grid.

4.4 Summary

Once the topological maps are obtained, the necessary conditions for the algorithm assembly are met. The next two chapters present the two main phases of the MSP approach: the partitioning phase and the local patrolling phase.

Chapter 5

MSP Algorithm – Partitioning Phase

The MSP Algorithm consists mainly of partitioning the global environment graph in several regions (or subgraphs) and, in a second phase, assigning a region to each mobile agent for efficient local patrolling. Therefore, after building the navigation graph on top of the original occupancy grid, the partitioning phase of the algorithm may begin. In this chapter, the process of partitioning a graph in a balanced way is described, as well as the generalization of partitions to create up to eight different regions. Note that graph partitioning is an early problem in graph theory.

5.1 Multilevel Graph Partitioning

In [KK98], a high quality multilevel approach for partitioning irregular graphs was presented. There are three main phases in this method: coarsening, partitioning and uncoarsening.

Basically, in the coarsening phase, a sequence of smaller and less complex graphs, each with fewer vertices, is obtained by collapsing vertices and edges into single vertices of the next level, which are called multinodes. An example of a coarsened graph with multinodes is presented in figure 5.1. Then, in the partitioning phase, the coarse graph obtained is bisectioned and lastly, in the uncoarsening phase, the partitioning is refined while the original graph is being restored. Diverse methods for all the three phases of the scheme are described in the document and comparisons were made between these methods. Analysing the results and their properties, some methods were selected for this work with some mi-

nor changes in a few cases.

For the coarsening method, the *Heavy-Edge Matching* was used. Vertices of the graph are visited in random order. To form multinodes, a vertex is matched with a neighbour vertex, such that the weight of the edge between them is maximum over all valid incident edges (heavier edge). Careful observation of the final multinodes generated shows that vertices which are highly related tend to be included in the same multinode. This method was chosen mainly due to its good results and simple approach.

In the partitioning phase, the approach used was based on the *KL Algorithm*[1], which starts with an initial bipartition of the graph and, in each iteration, it searches for a subset of vertices, from each part of the graph such that swapping them leads to a partition with smaller edge-cut. In our case, the edges are not cut. Every edge belongs to a unique region and the graph is partitioned in frontier vertices, as shown in figure 5.2. The green lines show where the graph partitioning was done. The frontier vertices define the boundaries of each subgraph. Basically, the difference in the approach used is that the initial bipartition starts by separating the largest multinode from the rest, evaluating an equilibrium condition (cost of all edges in both regions) and swapping multinodes from one side to the other only if the equilibrium condition improves.

As for the uncoarsening system, the *KL(1) Refinement* algorithm is used. The projected partition of the coarsened graph is assumed as a good initial partition for the upper level graph and vertices are swapped during the first uncoarsening phase, to improve the global equilibrium condition.

Please refer to [KK98] for further details on these methods and others.

5.2 Generalized Partitioning

As was previously seen, the Multilevel Graph Partitioning method creates a bisection of the graph, resulting in two smaller subgraphs. To be able to create more than two balanced regions, subgraph partitioning was developed based on the same methods and unbalanced partition conditions were also produced. For instance, a three region balanced partitioning is done by firstly dividing the main graph into two regions with an unbalanced

[1] named after Kernighan and Lin.

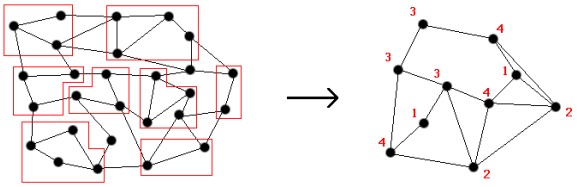


Figure 5.1: Example of a coarsened graph.

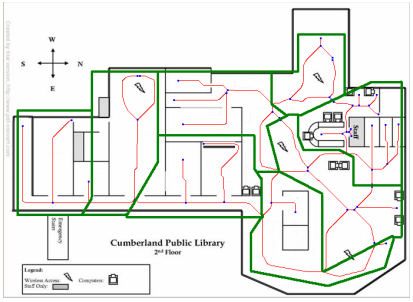


Figure 5.2: Example of a partition in 8 regions.

partition condition of 33.33% and 66.66% of its dimensions. Then, the largest subgraph obtained is divided into two regions with a 50% - 50% balanced condition. Three regions with 33.33% of the graphs dimension is the aim for the final result.

The patrolling simulator developed provides partitions from two up to eight balanced graph regions. Note that eight is the maximum partition number obtained with three stages, as shown in Table 5.1, which also presents the conditions used for generalized partitioning.

The number of regions created is defined according to the number of robots defined for simulation. Logically, if the input is only one robot, then no partition is needed and a patrolling scheme for the whole graph is implemented.

5.3 Summary

The partitioning phase is now complete. The generated partitions aim to be as balanced as possible. However, this is not always possible, due to graph complexity and connectivity limitations. The not so-balanced partitioning will affect global performance of the algorithm as shown later in the results chapter. Therefore, this is one of the most important phases of the project and also because local patrolling hugely depends on subgraph topology.

In the next chapter, the second phase of the MSP algorithm – the local subgraph patrolling phase – is studied.

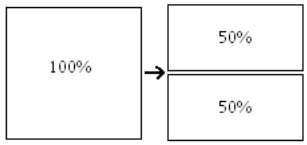
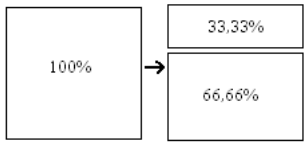
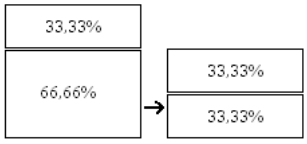
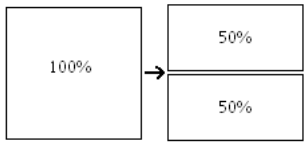
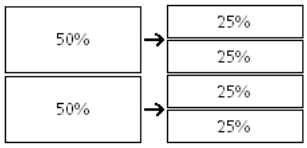
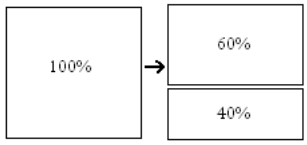
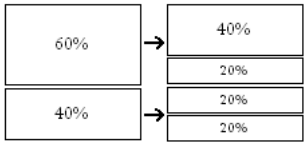
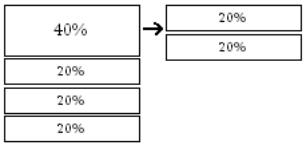
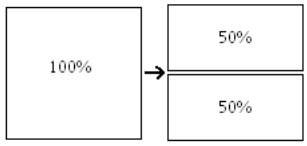
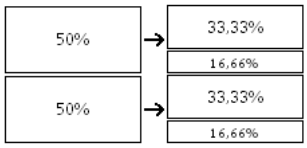
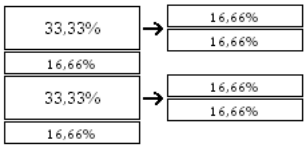
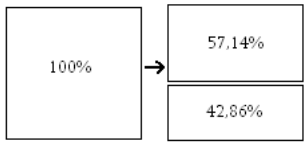
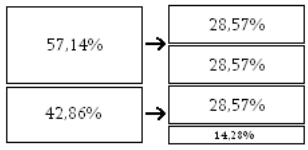
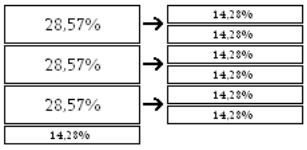
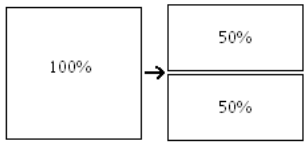
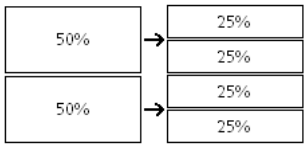
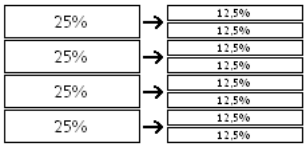
#	Stage 1 Partition	Stage 2 Partition	Stage 3 Partition
2			
3			
4			
5			
6			
7			
8			

Table 5.1: Generalized Partitioning.

Chapter 6

MSP Algorithm – Local Patrolling Phase

After the partitioning phase, the second phase takes place, which is the local patrolling scheme definition. Each mobile robot patrols one of the regions/subgraphs that were previously created. In order to do so, a staged search for a main patrolling path for every agent is done. After defining a main path, the patrolling route is completed by visiting the edges that are not included in the main path. Finally, if the main path is not cyclic then a return path to the route's initial vertex is computed.
Note that the main path is a non-repetitive sequence of edges that is intended to cover as much of the subgraph as possible, ideally the whole graph. Thus, the definition of each robot's main path is the most important factor for local patrolling performance. There are three key stages for defining a main path: firstly, searching for Euler and Hamilton circuits and paths, then the longest paths and finally, non-Hamiltonian cycles. All of them focus on classic graph theory problems, as we will discuss shortly.

6.1 Euler and Hamilton Circuits and Paths

The first stage of finding a main path is to search for Euler circuits and paths, these are paths that visit all edges of the graph once only. Euler circuits, which are also called Euler cycles, start and end on the same vertex whereas Euler paths do not. These paths are extremely appealing for patrolling purposes, because the whole subgraph is immediately

covered just by following them. Figure 6.1 shows an example of an Euler cycle and an Euler path.

There are necessary conditions for the existence of Eulerian circuits and Eulerian paths. According to [Hie73], all vertices in the graph must have an even degree, i.e. even number of neighbour vertices, to recognize the existence of an Euler cycle. For the Euler path case, all but two vertices must have an even degree. Those two vertices represent the endpoints.

These circuits and paths can be easily computed using Fleury's Algorithm [Fle83]. However, even though testing and finding Eulerian circuits and paths in a graph is relatively simple, most of the graphs, especially the ones that represent topological maps do not have these paths, because there are normally a high number of vertices that only have one neighbour (a dead-end) or have an odd number of neighbours.

After checking the non-existence of Euler cycles and paths, a search procedure for Hamilton circuits and paths takes place. Hamilton circuits and paths visit all vertices of the graph once only, the difference being that Hamilton circuits, or Hamilton cycles, start and end on the same vertex. Logically, these also represent very appealing main path choices for the local patrolling missions. Nevertheless, note that visiting all vertices does not imply visiting all edges of the graph.

There is no necessary condition to acknowledge the existence of Hamilton circuits and paths. Determining if they exist is NP-complete and represents a classic graph theory problem. Known solutions are normally based on heuristics.

In this work, the chosen detection method was the UHC, a fast algorithm proposed in [AV79] for finding Hamiltonian Paths and Circuits in undirected graphs. Again, most of the graphs do not have these paths, because there are normally a high number of vertices that only have one neighbour (a dead-end), which means that, when the edge connecting those vertices is visited, there is no edge connecting to a different vertex to be visited next. In these cases, further solutions for a main path must be computed as alternatives of Euler and Hamilton Circuits and Paths.

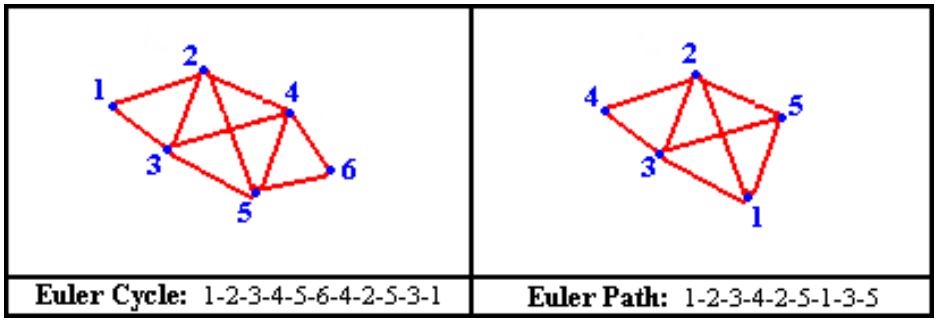


Figure 6.1: Examples of Euler Cycle and Path.

6.2 Longest Path

If Euler and Hamilton Circuits and Paths do not exist for the graph being considered (a common case), a longest path search procedure is done. The reason for this procedure is based on the fact that the longest path in a graph visits the maximum number of edges without repeating any; which, for patrolling purposes, is very attractive.

Like Hamilton paths, determining the longest path in a graph is also NP-complete. There is no optimal solution for this problem; therefore an approximate and simplified search procedure was created, in order to find the longest path from any vertex to another in the considered graph.

Firstly, from analysis of all the graphs formed after the Voronoi Diagram computation and consequent processing (Chapter 4), it was evident that it is frequent to have a substantial number of dead-ends in graphs, i.e. vertices with only one neighbour (degree one), also called leaf vertices. For this reason, the longest path search procedure was simplified to start and end in one degree vertices, leaving the vertices with higher degree for the Non-Hamiltonian cycle procedure, which will be analysed in the next section.

The created algorithm for searching and approximating the longest path is presented in pseudo-code (Algorithm 6.1). Note that its simplifications and its own nature makes it possible to find an appropriate solution relatively fast for our case, and, although it does not guarantee finding the longest existing path, it will always return a valid option for main path.

The first step is to build a list of all the vertices with degree one, then go through them and choose a source vertex and a destination vertex, run the algorithm in the pseudo-code and save the final path. The algorithm is run with different sources and destinations and

Algorithm 6.1 Longest-Path Approximation Algorithm

```
    /* Inputs                                                              */
    source: Source Vertex
    dest   : Destination Vertex
    list   : Degree One Vertices List
 1  All edges connected to degree one vertices ← visited; // This will avoid visiting them.
 2  source and dest connected edges ← unvisited;                  // Except for these.
 3  find_dest ← false;
 4  last_element ← source;
 5  path ← add_to_path(source);
 6  while true do
 7  |   last_element ← current last path element;
 8  |   num ← number of unvisited edges of last_element;
 9  |   if num = 0 then                                    // No Available Neighbours.
10  |   |   find_dest ← true;
11  |   |   forall neighbours of last_element do
12  |   |   |   if degree > 1 and not included in path and edge(last_element,neighbour) =
    |   |   |   visited then
13  |   |   |   |   edge(last_element,neighbour) ← unvisited;
14  |   |   |   end
15  |   |   end
16  |   |   path elements - -;        /* Withdraw last element of the path.  It will not
    |   |   comeback to the same, because its edge is marked as visited.  */
17  |   else
18  |   |   if dest is a neighbour then
19  |   |   |   if find_dest or num = 1 then /* There is only one vertex left.  Time to
    |   |   |   reach the destination.  */
20  |   |   |   |   path ← add_to_path(dest);
21  |   |   |   |   break;                                          // The End.
22  |   |   |   else
23  |   |   |   |   Withdraw dest from the list of neighbours to consider;
24  |   |   |   end
25  |   |   end
26  |   |   if none of the neighbours has degree >= 4 then
27  |   |   |   Choose next neighbour randomly;
28  |   |   else
29  |   |   |   Choose first neighbour with degree >= 4;
30  |   |   end
31  |   |   if find_dest then
32  |   |   |   find_dest ← false;
33  |   |   end
34  |   |   edge(last_element,neighbour) ← visited;
35  |   |   path ← add_to_path(neighbour);
36  |   end
37  end
38  return path;
```

every time a larger path in length is found, it should be kept. There is a tag on each edge to check if it was visited or not, and the *find_dest* variable allows us to delay the arrival at the final destination from the source vertex. In the end, the best path found by the algorithm is selected, for example, the one on the left in Figure 6.2.

6.3 Non-Hamiltonian Cycles

Sometimes graphs include cycles that are non-Hamiltonian. These cycles may or may not represent good main paths, depending on the number of elements and overall algorithm

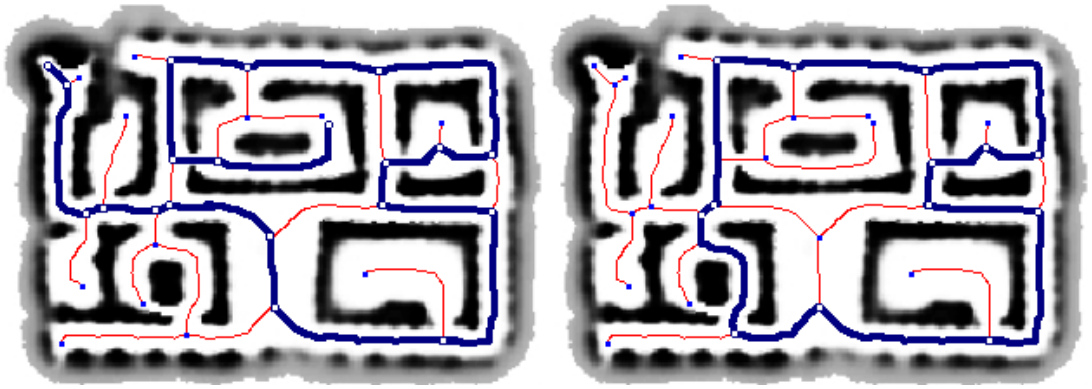


Figure 6.2: Longest Path and Non-Hamiltonian Cycle.
Obtained for a patrolling scheme of one robot in a graph.

performance when compared to the longest path found for the same graph. Small cycles with few elements are common but not desirable. On the other hand large cycles are very appealing. Since they have the advantage of returning to the starting point without needing to compute a return path, a cycle with an inferior number of edges may be a better choice for main path than the longest path.

When there is no cycle in the graph, the longest path is assumed as the main path, and when there are one or more cycles, they are only considered if they include at least half of the vertices of the graph. This measure was determined by undertaking simulation performance comparisons between cycles and longest paths for the some graphs.

The simplified method used to find cycles is also original and is very similar to the one used for longest paths, the differences being that a list of vertex with degree higher than one is computed and the while loop stops when we reach the source vertex again (a cycle was found or no cycle exists).

The algorithm should be run with distinct vertices and if a longer cycle is found, it should be saved. Finding the same cycle more than once, with different starting points, is expected. As an example, on the right side of Figure 6.2, the best cycle found for a given graph is shown.

6.4 Detours

Now that the main path is defined, it is necessary to complete the patrolling route by visiting the edges that are not included in the main path, except for the Euler circuits

and paths case, which already cover every edge of the graph.

All of the main path vertices will be visited in order and detours will be computed and included in the final patrolling route. This is done by inspecting whether the vertex has neighbours that are still not visited. If so, a detour is started with a maximum allowed number of edges to visit. Edges are added to the detour route, the ones linking vertices with inferior degree first, because they have less probability of being visited later by other detour routes. Also, it returns, when it arrives at a vertex that is already in the main path. The detour route should include all the edges accessible from the original vertex of the main path as long as it does not reach the maximum allowed number of edges to be visited. When it does, it checks if there is a direct edge linked to the original vertex and if not, it returns to the original vertex by a returning path that is computed and updated for every vertex that is visited in the detour. When the detour route(s) for this vertex is done, we move on to the next vertex of the main path until we reach the final vertex. Finally, when this process ends, if all of the edges in the subgraph were not visited, the maximum number of edges allowed in the detour route is increased and the process is repeated.

The idea, in the end, is to have a patrolling route that covers all edges and vertices with the least possible maximum number of edges allowed to visit by the detour route. The algorithm described is adaptive and the number of times required to repeat this process can be minimized establishing lower quotas and upper quotas. For instance, check if all edges are not covered when a maximum of one edge detour is allowed (in fact, this is generally enough when an Hamilton circuit or path is detected). This will be the starting lower quota. Afterwards, check an upper quota of half the number of edges in the graph, which is a pessimistic value. If all vertices were covered, than the next test number of edges should be between the lower and upper quota interval, if not, the previously considered upper quota should become a lower quota and a new upper quota should be established. This procedure should be done until the least possible maximum number of edges allowed to visit is identified. Using this approach, the correct value is found much faster than a pure incremental approach.

6.5 Returning to the route's initial vertex

If the main path is not an Euler, Hamiltonian or non-Hamiltonian circuit/cycle, it is necessary to compute a path to return to the route's initial vertex. Two ways were considered to do so: Inverse Path and Shortest Path. The first one is done by inversing the patrolling route back to the first vertex and the second one simply computes the shortest path, using Dijkstra's Algorithm [Dij59], from the last vertex to the initial one.
Both were compared and the decision to use the first one was justified by better results, in terms of overall patrolling frequencies for every point, in 96% of all studied cases.

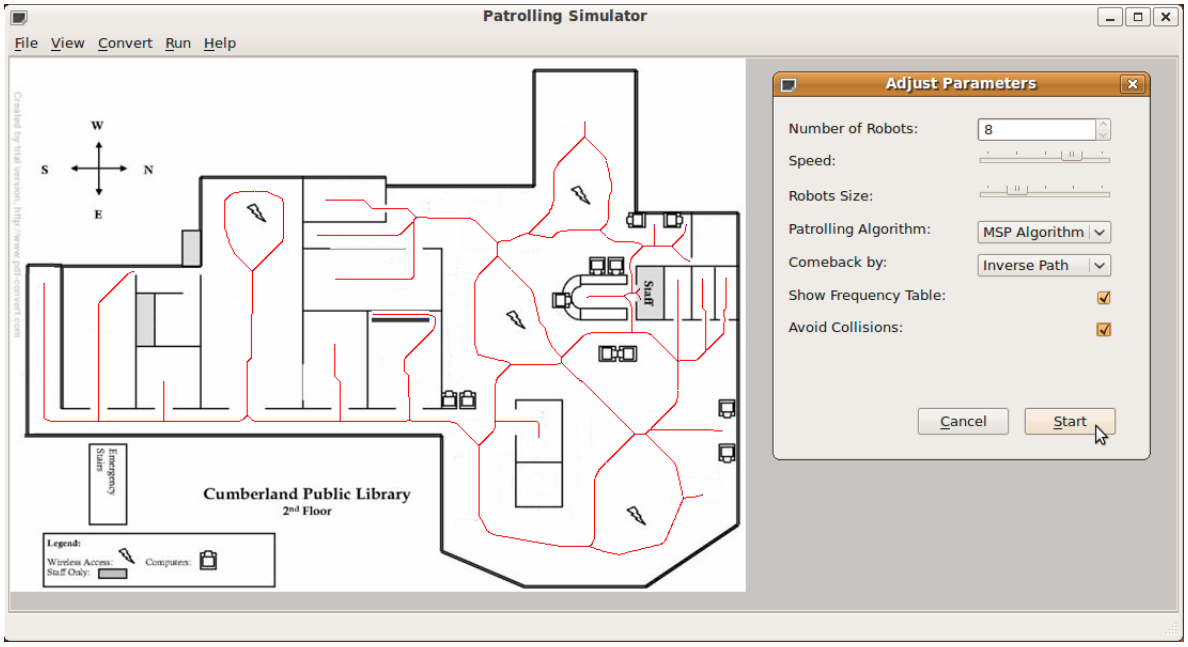


Figure 6.3: Simulation input parameters.

6.6 MSP Simulation Environment

The current and previous chapter presents the two phases of the MSP algorithm. The procedures described were developed in the Patrolling simulator tool that was created for this project. To simulate the Multilevel Subgraph Patrolling Algorithm, the user must fill the input parameters necessary for simulation (implying that the topological map is already acquired) as seen in Figure 6.3. Note that a collision avoidance mechanism was also developed. It basically prevents agents from being on the shared border vertices of

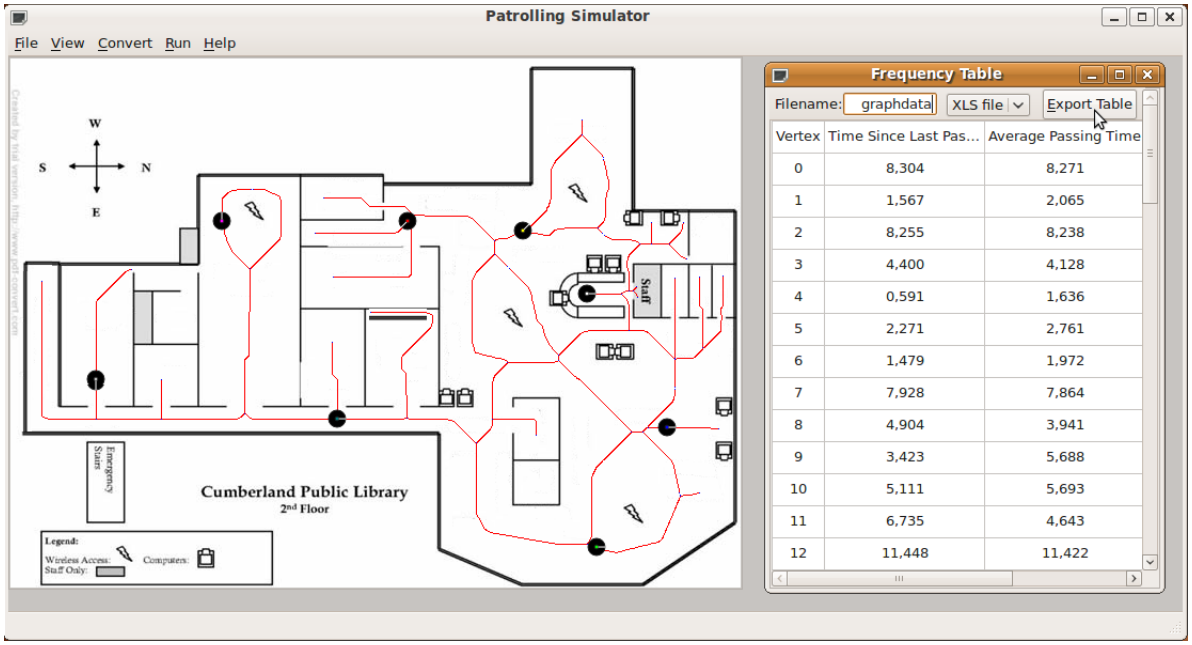


Figure 6.4: Simulation Environment.

the subgraphs at the same moment by making the one(s) which is/are furthest away from the vertex wait for the closer one. When the *Start* button is pressed on the Graphical User Interface, the environment graph is immediately partitioned in the same number of regions as inserted by the user for the robots' number. Later on, each local patrolling route is computed and the simulation then starts. The user will watch as the robots cover the entire graph, each in its own confined region. There is also the possibility of displaying a table, which outputs timing data, during simulation, for each vertex as illustrated in figure 6.4. This data can be exported in pure text format (txt), as well as in Excel Worksheet format (xls).

Also, a cyclic-based strategy (Section 3.1) was developed to allow comparisons between the two approaches. Details on this strategy and the comparison results will be presented in the next chapter.

6.7 Summary

In this chapter, the local patrolling phase of the MSP Algorithm has been explained. A good choice for the main path of the subgraph patrolling route is crucial in terms of performance of the algorithm and the steps that were taken to define it aim precisely to endorse good performance results. Logically, this deeply relies on the topology of the subgraph.

Simulation results were collected and will be discussed shortly in the next chapter. An alternative patrolling strategy was also computed and results for both schemes are presented and compared.

Chapter 7

Results and Discussion

After developing and testing the MSP algorithm, it is time to evaluate its performance. In this chapter, various simulation results are presented and its inference is discussed.

7.1 Implemented Algorithm for comparisons

A so-called cyclic algorithm was computed to have a comparison measure to contrast with the proposed approach. This algorithm searches for Hamiltonian cycles in a graph and places the robots along the path [EAK07]. Since there is no defined behaviour when no Hamilton cycles exist, the same methods as ours were used, in these cases, for computing the best possible path in the entire graph. As a result, both approaches are usually equivalent when only one agent is used. Note, however, that, with the cyclic approach, all mobile robots follow the same exact route, through the whole graph, uniformly lagged in time.

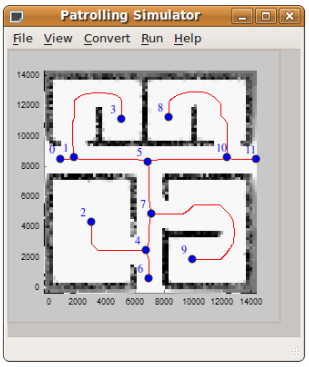
This algorithm was chosen for comparison because it is designed for maximizing the visiting frequency of all points of the graph and is known to have good overall results when compared to different approaches [ARS+04]. Also, its fixed and predetermined nature, makes it relatively simple to compute.

7.2 Results of the comparison with cyclic algorithm

A large set of maps were used to compare average point frequency performance for both approaches. Due to space limitations, only six representative environments are presented in this chapter. They were denoted as maps A to F and are illustrated in figures 7.1 to 7.6.

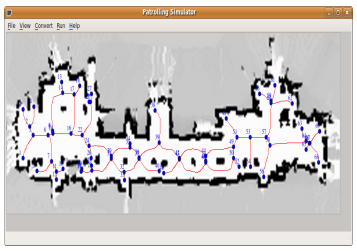
Map A has a very simple graph with 12 vertices and due to minimum size dimension constraints, for each subgraph, there is a maximum number of partitions according to the graph complexity, in this case four. Map B, C and E are medium graphs with relatively defined paths. B has 70 vertices, C has 41 and E has 74. Map D represents a more complex graph with 268 vertices and Map F is a completely different case, due to the existence of a main Hamilton cycle in the graph and Hamilton paths in the generated subgraphs. This is a much simpler graph with 12 vertices and very good connectivity between vertices. Again, there is a partition number limitation like the one mentioned for map A.

Now that the six samples are identified, we are able to analyse the results.



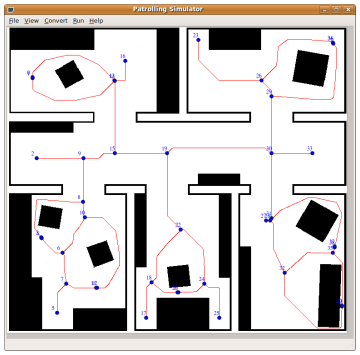
n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	2564	2564	0.84	0.84	0.49	0.49
2	2564	1995	1.68	1.77	0.98	1.03
3	2564	1725	2.51	2.71	1.48	1.55
4	2564	1881	3.35	3.68	1.97	2.09

Figure 7.1: Results for map A.



n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	8017	8017	0.34	0.34	0.16	0.16
2	8017	7536	0.67	0.68	0.33	0.35
3	8017	7608	1.01	0.96	0.49	0.55
4	8017	7029	1.35	1.34	0.65	0.79
5	8017	6692	1.68	1.68	0.82	0.99
6	8017	6451	2.02	1.92	0.99	1.24
7	8017	6454	2.36	2.34	1.14	1.42
8	8017	6073	2.70	2.76	1.30	1.80

Figure 7.2: Results for map B.

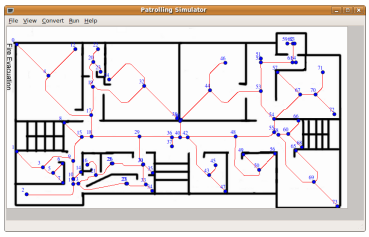


n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	15666	15666	0.16	0.16	0.08	0.08
2	15666	14383	0.33	0.32	0.16	0.19
3	15666	13432	0.49	0.47	0.25	0.29
4	15666	13048	0.65	0.61	0.33	0.42
5	15666	12658	0.81	0.79	0.41	0.49
6	15666	12023	0.98	0.94	0.49	0.64
7	15666	11538	1.14	1.11	0.57	0.82
8	15666	11246	1.30	1.21	0.66	0.97

Figure 7.3: Results for map C.

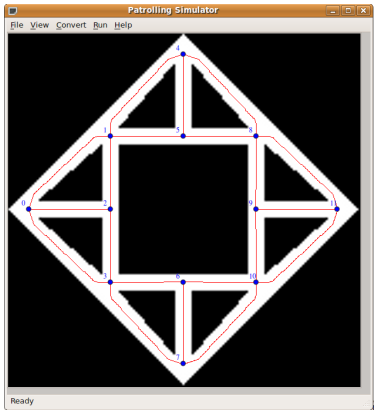
n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	21949	21949	0.13	0.13	0.06	0.06
2	21949	20923	0.26	0.27	0.13	0.13
3	21949	20039	0.39	0.41	0.19	0.21
4	21949	20037	0.52	0.55	0.25	0.29
5	21949	19375	0.66	0.68	0.32	0.38
6	21949	18951	0.79	0.80	0.38	0.44
7	21949	18403	0.92	0.94	0.44	0.51
8	21949	18269	1.05	1.08	0.51	0.64

Figure 7.4: Results for map D.



n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	13485	13485	0.20	0.20	0.10	0.10
2	13485	13136	0.40	0.40	0.20	0.22
3	13485	12498	0.60	0.56	0.30	0.35
4	13485	11668	0.81	0.73	0.40	0.45
5	13485	11402	1.01	0.99	0.50	0.62
6	13485	10839	1.21	1.24	0.60	0.74
7	13485	10768	1.41	1.27	0.70	0.84
8	13485	10479	1.61	1.49	0.80	0.96

Figure 7.5: Results for map E.



n_r	$\sum d_T$[C]	$\sum d_T$[MSP]	F[C]	F[MSP]	σ[C]	σ[MSP]
1	3931	3931	0.57	0.57	0.10	0.10
2	3931	3985	1.14	1.16	0.20	0.17
3	3931	3697	1.72	1.67	0.30	0.30
4	3931	3641	2.29	2.34	0.40	0.34

Figure 7.6: Results for map F.

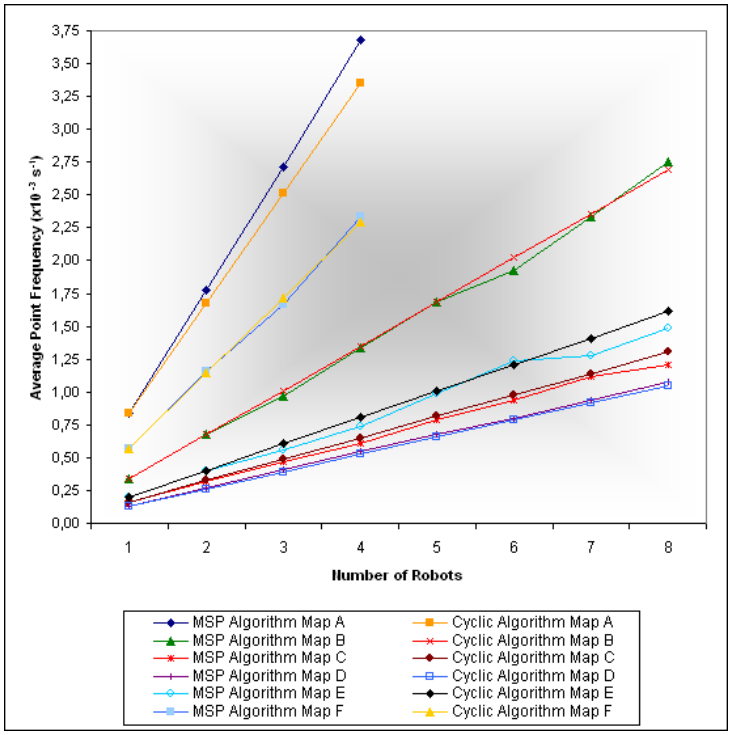


Figure 7.7: Results for the six different maps.

7.3 Discussion

The MSP Algorithm got better performance results with an arbitrary number of robots for map A and D, as seen by the correspondent curves in figure 7.7. The cyclic algorithm got higher visiting frequencies in the case of map C. For map B, the MSP was only better in the case of two, five and eight robots, being overly very close to the cyclic strategy in the other tests. Similarly to map B, in map E's case, the MSP Algorithm got close results to the Cyclic Algorithm up to six robots (being actually better for the six robots' case). For seven and eight robots, the results were not so good, mostly due to low quality partitions as a consequence of graph limitations. As for the special case of map F, the MSP was globally better, having only lower frequency in the case of three robots.

As seen by their representations Maps B, C and E are medium graphs with relatively defined paths. Despite this, note that each map has distinct frequency intervals due to their different Euclidean dimensions.

Maps A, B, D and F had overall good partitioning results. For map C, the partitioning

quality was overly better for odd number of regions and not so good for even numbers. In the map E's case, better partitioning is coincident with the cases where there were better results, namely, for two, five and six regions.

General results show that visiting frequencies with the MSP Algorithm are better in graphs where there are high-quality partitions and/or the global cyclic path has too many edge repetitions, while the addition of the local patrolling paths in each region is globally less repetitive.

Note that the presented results are normalized and constant speed for each robot of 1m/s is considered, also each pixel in the Voronoi Graph is considered equivalent to a meter. Assuming this, results show that in 97,2% of cases, the MSP approach point frequencies are within the interval $\left[\frac{n_r}{\sum d_T}, \frac{4n_r}{\sum d_T}\right] \times 10^{-3}$, with $\sum d_T$ being the sum of all the distances travelled by robots in the MSP approach in meters, which is equivalent to the time (in seconds) that the robots would take to cover all the map, in separated time schedules, with a 1m/s constant speed and n_r being the number of robots patrolling the graph. Note that $\sum d_T$ hugely depends on graph complexity and vertices number. Frequencies not within the interval occurred only in some cases where the global complexity of the graph resulted in unbalanced partitions (generally with high number of regions) or not so satisfactory main paths.

The distance travelled by the mobile agents to cover the whole graph ($\sum d_T$) is almost always shorter for the MSP approach. This is mainly due to the non-redundant work as opposed to the cyclic strategy. On the other hand, the standard deviation (σ) is generally lower for the cyclic approach because the MSP has global frequency values less close to average values, mostly because of the inversed path followed when no representative cycles exist in the subgraph. Inverting the path results in better frequencies for points in the middle and worse for points near the path extremities.

7.4 Summary

In this chapter, results were presented and discussed. In the subsequent chapter, an overview of the algorithm is done. Its benefits and weaknesses are debated and finally this dissertation ends with conclusions and future work.

Chapter 8

Conclusions

The area of patrolling with autonomous mobile robots still has vast research possibilities and the work presented in this dissertation addresses one of the core subjects in this context: The multi-robot systems' problem of efficiently patrolling a given infrastructure environment.
In this last chapter, a global and self-critical overview of the proposed approach takes place in order to sum up the work. Also, final conclusions and possible related future work directions are discussed.

8.1 MSP Overview

In this document, a new multi-robot patrolling algorithm based on balanced graph partition was proposed. This approach decreases redundant work and consequent robot detritions when compared to a cyclic approach. It is also relatively simple, effective, scalable, distributed and robust. Fault-tolerance can be easily implemented, when a robot fails, by just rerunning the algorithm with fewer robots and send them to their new starting positions. Simulation results prove its good scalability and agents distribution in larger environments. Also, there is no need for a communication system or expensive sensors. The interference between robots is minimal; they may only be near to each other in the vertices of the boundaries of their regions. A collision avoidance mechanism to forbid two or more robots from being in the same border vertex at the same time was also developed. On the other hand, for a cyclic approach when there are no main cycles, it is much

harder to employ such a mechanism, since robots may be coming and going through the same path. Also, for an intelligent intruder, it is easier to attack an environment, which is patrolled using a cyclic approach because robots follow the same route over and over again, whereas in our approach, although being deterministic, each robot follows its own patrolling cycle and completes it in different periods of time, so it is much more difficult to track every robot's path and predict which areas of the environment are better for intrusion, because the global robot disposition is continuously changing. Possible disadvantages of this approach are the longer time that robots take to cover points on the extremities of the patrolling route when there are no cycles, the need to redefine the paths when an agent fails and, even though it is a difficult task for an observer, its deterministic nature makes it possible to calculate and predict the robots disposition in a given point in time. Also, the performance decreases in graphs, in which the partitions are not so balanced, when compared to the cyclic approach.
As a result of the developed work, a paper [PR09] was written and submitted to the 25th ACM Symposium on Applied Computing (SAC 2010), which will take place in Sierre, Switzerland during March 2010. The referred paper is available in the project's CD.

8.2 Future Work

Some issues are still left open and correspond to future guidelines that can be followed to improve current work. An even more thorough study with more maps, considering the graphs complexity and comparisons can lead to a criterion to define the more efficient main path, either a non-Hamiltonian cycle or the longest path. Also the fault-tolerance approach mentioned before can be implemented as well as simulating an intruder attack and calculate the time necessary to detect it and the best possible location for attacks. An alarm system could also be discussed when an anomaly is detected by any robot. An interesting feature would also be an automatic method of adequate agents' team size estimation considering the given environment. Finally, it would be interesting to take this approach further than simulation and test it with mobile robots in real world scenarios.
It is the author's opinion that the work done represented a very interesting challenge and it is his intention to proceed with academic studies (Ph.D.) in this research area.

Bibliography

[AKK08] N. Agmon, S. Kraus, G. Kaminka, *Multi-Robot Perimeter Patrol in Adversarial Settings*, IEEE International Conference on Robotics and Automation (ICRA 08), 2339-2345, Passadena, California, U.S.A., May 19-23, 2008. ISBN 978-1-4244-1646-2.

[Alt03] P. Althaus, *Indoor Navigation for Mobile Robots: Control and Representations*, Doctoral Dissertation, Royal Institute of Technology, Universitet Stockholms, Universitetsservice US-AB, Stockholm, Sweden, October, 2003. ISBN 91-7283-605-9.

[ARS+02] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, B. Vöcking, *Randomized Pursuit-Evasion in Graphs*, Proceedings of the 29th International Colloquium on Automata, 901-912, Languages and Programming, Málaga, Spain, July 8-13, 2002. ISBN 3-540-43864-5.

[ARS+04] A. Almeida, G. Ramalho, H. Sanana, Y. Chaveleyre et al, *Recent Advances on Multi-Agent Patrolling*, Brazilian Symposium on Artificial Intelligence (SBIA 2004), 474-483, Vol. 3171, São Luís, Brazil, September 29 - October 1, 2004. ISBN 3-540-23237-0.

[AV79] D. Angluin, L. Valiant. *Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings*. Journal of Computer and System Sciences, 155-193, 18(2), April 1979.

[BJK05] P. Beeson, N. Jong, B. Kuipers. *Towards Autonomous Topological Place Detection Using the Extended Voronoi Graph*. International Conf. on Robotics and Automation, 4373-4379, Barcelona, Spain, April 18-22, 2005.

[BK91] J. Borenstein, Y. Koren, *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*, IEEE Journal of Robotics and Automation, 278-288, 7(3), June, 1991.

[BM76] J. Bondy, U. Monty, *Graph Theory with Applications*, The Macmillan Press Ltd., 5th Edition, Elsevier Science Publishing Co. Inc, U.S.A., 1976. ISBN 0-444-19451-7.

[BS02] M. Batalin, G. Sukhatme, *Multi-Robot Dynamic Coverage of a Planar Bounded Environment*, Technical Report, Robotic Embedded Systems Laboratory, University of Southern California, 2002. Available: `http://www.cens.ucla.edu/~maxim/Publications/papers/331.pdf`

[Che04] Y. Chevaleyre, *Theoretical Analysis of the Multi-agent Patrolling Problem*, Proceedings of the Intelligent Agent Technology: IAT '04, IEEE/WIC/ACM International Conference, 302-308, Beijing, China, September 20-24, 2004. ISBN 0-7695-2101-0.

[Chv75] V. Chvatal, *A combinatorial theorem in plane geometry.* Journal of Combinatorial Theory Series B, 39-41, Vol. 18, 1975.

[DA99] R. Dollarhide, A. Agah, *Simulation and Control of Distributed Robot Search Teams*, Computers & Electrical Engineering, 625-642, 29(5), Elsevier Science Ltd., 1999.

[Dij59] E. W. Dijkstra. *A Note on Two Problems in Connection with Graphs.* Numerische Math, 269-271, Vol. 1, 1959.

[DK07] D. Dimarogonas, K. Kyriakopoulos, *Decentralized Navigation Functions for Multiple Robotic Agents with Limited Sensing Capabilities*, Journal of Intelligent and Robotic Systems, 411-433, 48(3), Springer Netherlands, March, 2007, ISSN 0921-0296.

[DJM+93] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, *Map Validation and Robot Self-Location in a Graph-Like World*, Thirteenth International Conference on Artificial Intelligence, 1648-1653, Chamberry, France, August, 1993.

[DJM+97] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, *Map Validation and Robot Self-Location in a Graph-Like World*, Robotics and Autonomous Systems, 159-178, 22(2), Elsevier Ltd., November, 1997.

[DJM+98] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, *Topological Exploration with Multiple Robots*, Proceedings of the 7th International Symposium on Robotics with Applications (ISORA'98), Anchorage, Alaska, May 10-14, 1998.

[EAK07] Y. Elmaliach, N. Agmon, G. Kaminka, *Multi-Robot Area Patrol under Frequency Constraints*, IEEE International Conference on Robotics and Automation (ICRA 07), 385-390, Rome, Italy, April 10-14, 2007. ISBN 1-4244-0601-3.

[Fle83] Fleury. *Deux problemes de geometrie de situation.* Journal de mathematiques elementaires, 257-261, 1883.

[Ger03] V. Gervasi, *Robotic Cops: The Intruder Problem*, IEEE Conference on Systems, Man and Cybernetics (SMC 2003), 2284-2289, Vol. 3, Washington D.C., U.S.A., October 5-8, 2003. ISBN 0-7803-7952-7.

[GLL+97] L. Guibas, J. Latombe, S. Lavalle, D. Lin, R. Motwani, *Visibility-Based Pursuit-Evasion in a Polygonal Environment*, Algorithms and Data Structures, 5th International Workshop, WADS '97, 17-30, Vol. 1272, Halifax, Canada, August 6-8, 1997. ISBN 3-540-63307-3.

[GPM07] Y. Guo, L. Parker, R. Madhavan, *9 Collaborative Robots for Infrastructure Security Applications*, Studies in Computational Intelligence (SCI), Springer-Verlag Berlin Heidelberg, 185-200, Vol. 50, April 22, 2007. ISSN 1860-9503.

[GTG04] B. Gerkey, S. Thrun, G. Gordon, *Visibility-Based Pursuit-Evasion with Limited Field of View*, Proceedings of the National Conference on Artificial Intelligence (AAAI), 20-27, San Jose, California, U.S.A., July 25-29, 2004.

[Hie73] C. Hierholzer. *Uber die Moglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechnung zu umfahren.* Mathematische Annalen 6, 30-32, 1873.

[HKS09] G. Hollinger, A. Kehagias, S. Singh. *Efficient, Guaranteed search with multi-agent teams*. In Proceedings of Robotics: Science and Systems V, Seattle, Washington, USA., Jun. 28 - Jul. 1, 2009.

[KC07] A. Kolling, S. Carpin, *The GRAPH-CLEAR Problem: Definition, Theoretical Properties and its Connections to Multirobot Aided Surveillance*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007), 1003-1008, San Diego, California, U.S.A., October 29 - November 2, 2007. ISBN 978-1-4244-0912-9.

[KC08] A. Kolling, S. Carpin, *Multi-robot Surveillance: an Improved Algorithm for the GRAPH-CLEAR Problem*, IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008), 2360-2365, Pasadena, California, U.S.A., May 19-23, 2008. ISBN 978-1-4244-1646-2.

[KC08a] A. Kolling, S. Carpin, *Extracting Surveillance Graphs from Robot Maps*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), 2323-2328, Nice, France, September 22-26, 2008. ISBN 978-1-4244-2057-5.

[KK98] G. Karypis and V. Kumar. *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. Society for Industrial and Applied Mathematics, Journal of Scientific Computing, 359-392, 20(1), 1998.

[Kon03] K. Konolige et al, *Centibots: Large Scale Robot Teams*, Multi-Robot Systems: From Swarms to Intelligent Automata, Vol. 2, Kluwer, 2003.

[LGB+97] S. Lavalle, H. González-baños, C. Becker, J. Latombe, *Motion Strategies for Maintaining Visibility of a Moving Target*, IEEE International Conference on Robotics & Automation (ICRA'97), 731-736, Vol. 1, Albuquerque, New Mexico, U.S.A., April 10-25, 1997. ISBN 0-7803-3612-7.

[LS08] M. Likhachev, A. Stentz, *Information Value-Driven Approach to Path Clearance with Multiple Scout Robots*, IEEE International Conference on Robotics and Automation (ICRA'08), 2651-2656, Pasadena, California, U.S.A., May 19-23, 2008. ISBN 978-1-4244-1646-2.

[Mia07] M. Miazaki, *Sistema de controle multi-robô baseado em colônia de formigas artificiais*, Master Thesis, Instituto de Ciências Matemáticas e de Computação - USP, Brazil, February, 2007.

[Min07] D. Miner, *Swarm Robotics Algorithms: A Survey*, May 17, 2007, Available: `http://www.coral-lab.org/~don/projects/swarmrobotics.pdf`

[Por09] D. Portugal, *Multiple Autonomous Mobile Robots Patrolling Related Issues*, Report, Faculty of Science and Technology, University of Coimbra, Portugal, September, 2009.

[PR09] D. Portugal and R. Rocha. *MSP Algorithm: Multi-Robot Patrolling based on Territory Allocation using Balanced Graph Partitioning*. Submitted to the 25th ACM Symp. on Applied Computing (SAC 2010), special track on Intelligent Robotic Systems (ROBOT), Sierre, Switzerland, Mar. 22-26. 2010.

[Roc05] R. Rocha, *Building Volumetric Maps with Cooperative Mobile Robots and Useful Information Sharing: a Distributed Control Approach based on Entropy*, Ph.D. Thesis, Faculty of Engineering of University of Porto, Portugal, October, 2005.

[Rub74] F. Rubin, *A Search Procedure for Hamilton Paths and Circuits*, Journal of the Association for Computing Machinery, 576-580, 21(4), ACM, New York, U.S.A., October, 1974. ISSN 0004-5411.

[SA03] A. Sgorbissa, R. Arkin, *Local Navigation Strategies for a Team of Robots*, Robotica, 461-473, 21(5), Cambridge University Press, October, 2003. ISSN 0263-5747.

[She92] T. Shermer, *Recent Results In Art Galleries*, Proceedings of the IEEE, 1384-1399, 80(9), IEEE Press, New York, NY, USA, September, 1992. ISSN 0018-9219.

[SRL04] S. Sachs, S. Rajko, S. Lavalle, *Visibility Based Pursuit-Evasion in an Unknown Planar Environment*, The International Journal of Robotics Research, 3-26, 23(1), MIT Press - Massachusetts Institute Of Technology, ISSN 0278-3649, January, 2004.

[SUP+07] D. Saitov, U. Umirov, J. Park, J. Choi, S. Lee, *Effective Map Building Using a Wave Algorithm in a Multi-Robot System*, International Journal of Precision Engineering and Manufacturing, 69-74, 9(2), KSPE, 2007.

[SYX04] W. Sheng, Q. Yang, N. Xi, *A Distributed Bidding Algorithm for Multi-Robot Exploration with Safety Concerns*, International Conference on Complex Systems 2004 (ICCS2004), 633-640, Boston, MA, U.S.A., May 16-21, 2004.

[Sza04] R. Szabo, *Topological Navigation of Simulated Robots using Occupancy Grid*, International Journal of Advanced Robotic Systems, 201-206, 1(3), 1729-8806, September, 2004.

[Thr98] S. Thrun, *Learning Maps for Indoor Mobile Robot Navigation*, Artificial Intelligence, 21-71, Vol. 99, Elsevier Science Ltd., 1998.

[WSB08] K. Wurm, C. Stachniss, W. Burgard, *Coordinated Multi-Robot Exploration using a Segmentation of the Environment*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), 1160-1165, Nice, France, September 22-26, 2008. ISBN 978-1-4244-2057-5.

[ZFV94] U. Zimmer, C. Fischer, E. Von Puttkamer, *Navigation on Topologic Feature-Maps*, 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, 131-132, Iizuka, Fukuoka, Japan, August 1-7, 1994.