Role-Based Multi-Robot Exploration

Julian de Hoog

Exeter College

Department of Computer Science
University of Oxford

Doctor of Philosophy

May 2011

© Copyright 2011
Julian de Hoog

All Rights Reserved

Abstract

The advent of robotic technologies means that teams of robots can be used for an ever wider range of tasks, such as exploration of unknown terrain, search and rescue in disaster scenarios, and inspection of hazardous areas. In many such applications, partial or full autonomy is a desirable characteristic for the robots, since this can reduce the load on human operators and improve the speed and quality of team coordination. However, in many scenarios the environments of interest may contain significant communication challenges due to their size or complexity, introducing the need for robust methods for communication-limited exploration by multiple robots.

This thesis proposes "Role-Based Exploration", a novel exploration algorithm for multi-robot systems that aims to efficiently gather information obtained by all members of the team in a single location. In Role-Based Exploration, some of the robots in the team explore the environment while others act as mobile relays, ferrying information back and forth within the team. By imposing a team hierarchy, choosing clever locations for robots to meet, and applying some simple rules that allow robots to exchange places within this hierarchy, a robust exploration system emerges that reactively adjusts to communication availability and to the shape of the environment.

Role-Based Exploration demonstrates several advantages over other methods, particularly as communication becomes less reliable. New information obtained by the team is brought to a single location quickly and in regular intervals, team members share information well and often, and the full team effort can be easily monitored and controlled. The approach has been implemented and compared to competing algorithms both in simulation, and on a team of Pioneer robots.

To my mother Regina, whose perseverance was an inspiration.

Acknowledgements

First and foremost, I am deeply grateful to my supervisor Stephen Cameron. Before I even became a member of the department, Stephen was kind enough to take me on as a volunteer research student, and that changed the course of my life. Without his encouragement I would likely not have begun the DPhil, which has proven enormously rewarding and fulfilling. Stephen provided invaluable guidance over the past four years, and his assistance and initiative with a number of scholarship applications and conference submissions made my graduate experience all the richer. I couldn't have asked for a better supervisor.

Second, I am very thankful to Arnoud Visser at the Intelligent Systems Laboratory of the University of Amsterdam. Arnoud agreed to let me join his RoboCup team early on in my DPhil, and it led to several years of fruitful collaboration. I have learnt much from Arnoud and his students not just about the various problems that the robotic search-and-rescue domain offers, but also about teamwork and leadership. It was a pleasure to visit Arnoud in Amsterdam and to host him in Oxford, and I hope we continue to see more of one another in the future.

Third, I am deeply indebted to Adrian Jiménez-González at the University of Sevilla, along with his supervisors Anibal Ollero and José Ramiro Martínez de-Dios. Adrian spent many long days, nights, and weekends helping me with implementation issues involving the robots, and I would never have achieved the results in this thesis without his assistance. Ramiro's suggestions throughout the experiments were very helpful, and I would not have had the opportunity in the first place without Anibal's kind invitation and support through funding from the CONET Network of Excellence.

In the Computing Laboratory, I am grateful to many members of the department that played a role in my DPhil experience. During my Transfer of Status and Confirmation assessments, I received invaluable advice from Niki Trigoni, Paul Newman, and Alexandru Baltag, and their suggestions guided my ensuing research in the right direction. I will never forget joint

academic travels with Jie Ma, who taught me to see the world from a Chinese perspective. I also learnt much from my many colleagues in Skylab: Ricklef Wohlers, Muzammil Hussain, Andrew Markham, Sonia Waharte, Andrew Symington, Sarfraz Nawaz, Hongkai Wen and Robert Patrick.

Outside of the Comlab, conversations with my friend Allan Dafoe, in particular, helped me to look at my research from different angles. When it was time for a break from work, my very good friends Mauro, Rebecca, Sian, Babak, Igor and Rod (and the rest of the team) were always there to help me out – thanks to you all!

A thesis can be as trying for a student's partner as for the student himself, and my case was no exception! I can't even begin to express my gratitude to my wonderful fiancee Keira, who was endlessly patient, supportive and accommodating, even when my many delays affected her own life plans. During the final months of writing the thesis I spent much time in Australia, and my new family Kirsten, Stewart and Alex were very kind to me even as I spent my days burrowed away in seclusion.

To my father Adriaan, my sister Ariane, and my brother-in-law John, I will never forget the journey we've all been through together over the past year, and the way in which we all supported one another. You provided the solid foundation to my life without which I would not have been able to return to the thesis. Thank you.

Finally, Mum, thank you for everything, I have hidden six pixels in this thesis for you!

Julian de Hoog

Oxford, 23 May 2011

Contents

List of Figures

List of Tables

List of Acronyms

AMCL	Adaptive Monte Carlo Localisation (Player driver)
LF	Leader-Follower exploration
G	Greedy exploration
GP	Greedy exploration with Periodic return
MRESim	Multi-Robot Exploration Simulator
MRICP	Map Reference Iterative Closest Point (Player driver)
RB1	Role-Based exploration without role swaps
RB2	Role-Based exploration with role swaps
SLAM	Simultaneous Localisation And Mapping
UAV	Unmanned Aerial Vehicle
VFH	Vector Field Histogram (Player driver)
YARP	Yet Another Robot Platform

Chapter 1

Introduction

1.1 Background

Historically robots have been limited to industry, but as technologies mature, robots are being used more and more for other purposes. These include exploration of unknown or remote terrain (such as the robotic rovers on Mars [107]), search and rescue in disaster zones (such as the robots used to find human remains after the collapse of the World Trade Centre in September 2001 [23]), inspection of hazardous areas (such as monitoring of nuclear waste facilities [143], or bomb disposal [154]), in urban settings (*e.g.* for surveillance [64]), and even in the home (for example, the popular Roomba vacuum cleaner [46]). In particular the miniaturisation of hardware means that now teams of small robots may be used for many tasks.

Today, most robots are controlled remotely by human operators, and control of a robot typically requires the operator's full concentration. However, the amount of incoming information can be overwhelming, there may be many other factors competing for the operator's attention, and in the case of very remote environments, control delay can be significant. As a result, there is much incentive to offload some of the work to the robots. In other words, partial or full autonomy is a desirable capability for robots in many scenarios.

For a team of robots to successfully explore autonomously, robust performance of a number of functionalities is necessary: the robots must be able to navigate without being hindered by obstacles; they must keep track of where they are and maintain information about the environment; and they must communicate with one another in order to coordinate their actions and improve team performance.

1.2 Motivation

In many applications, however, communication is likely to be unreliable. Environments may extend beyond the team's communication range, or they may be complex and contain significant interference, and often there is no existing communication infrastructure. Thus, communication links between members of the team are often liable to drop-out or failure.

While there has been much work on multi-robot exploration and mapping, most existing approaches either assume perfect communication, or apply a strategy that aims to keep team members within range of one another. In large and complicated environments, this means that some parts of the environment may never be reached.

The main aim of this thesis is to provide an early approach towards autonomous exploration of previously unknown environments, even beyond team communication range limits. In short, the motivation of this thesis was to investigate the following two questions: How can a team of robots be coordinated to explore a previously unknown and communication-limited environment as efficiently as possible; and how can new information obtained by this team be gathered at a single location as quickly and as reliably as possible?

1.3 Approach

To solve this problem, a novel exploration algorithm for teams of robots is proposed: *Role-Based Exploration.* In Role-Based Exploration, some of the team members explore the environment, while others act as mobile relays, ferrying information back and forth between exploring robots and a central "BaseStation". By maintaining a team hierarchy, choosing clever locations for robots to meet, and applying some simple rules that allow robots to exchange places within the hierarchy, a robust exploration system emerges that reactively adjusts to communication availability and the shape of the environment. Information is shared well throughout the team, and the regular return of information updates to the BaseStation means that the full team effort can be easily monitored and controlled from a single location.

1.4 Progression of Research

Early versions of Role-Based Exploration were implemented for use by the "Amsterdam Oxford Joint Rescue Forces" [45], a joint team between the Universities of Amsterdam and Oxford that competes in the Virtual Robots rescue competition, which forms one of the events endorsed by the RoboCup initiative [6, 109]. The Virtual Robots competition uses a 3D simulator, USARSim [153], and human competitors have to direct simulated robots through various virtual disaster scenarios that have been created by the competition organisers.

USARSim is a useful and widely used simulator, but certain characteristics (such as the time it takes to build new environments, and the fact that communication is simulated by a separate application) mean that for extensive simulations it can be time-consuming to use. For the purposes of this thesis a new simulation tool, the Multi-Robot Exploration Simulator (MRESim), was developed. This simulator has been tailored specifically for quick and simple comparison of various multi-robot exploration algorithms, and is described in greater detail in Appendix A.

The behaviour of Role-Based Exploration is demonstrated initially in MRESim, where it is compared to several competing approaches. Its performance relative to these other approaches is examined for each of four parameters: sensor range; communication range; number of robots in the team; and type of environment. Several further tests are conducted in simulation that emulate, to some degree, scenarios that might be found in reality.

To examine Role-Based Exploration in reality, it is also demonstrated on a team of Pioneer robots as part of a series of experiments that were undertaken at the University of Seville under the auspices of the CONET project[1]. Its performance on a real multi-robot system is compared to other methods first within a controlled, purpose-built environment; and subsequently in an uncontrolled, fully realistic domain as the robots explore the halls of a large building.

1.5 Thesis Organisation

Figure 1.1 demonstrates how this thesis fits into contemporary robotics research. The literature review (Chapter 2) has been divided into two parts: the first provides a brief overview of some of the applications the research in this thesis is most relevant to; and the second describes in greater depth the work that has been done in multi-robot exploration elsewhere.

In Chapter 3 the exact nature of the problem is defined, including the assumptions that are made, the exact goals of the research, and the performance metrics that are used to compare competing approaches.

Chapter 4 describes the main contribution of this thesis, Role-Based Exploration, and details two improvements that were made over the course of the research which

[1] CONET: Cooperating Objects Network of Excellence (INFSO-ICT-224053) funded by the European Commission under ICT, Framework 7.

led to gains in exploration speed and connectivity of the team.

The performance of Role-Based Exploration in simulation is demonstrated in Chapter 5, while the implementation of Role-Based Exploration on a real multi-robot system is described in Chapter 6.

Finally, Chapter 7 discusses various characteristics of the approach and examines in greater detail its strengths and weaknesses, before Chapter 8 concludes the thesis.

Robot Applications	**Robot Capabilities**
Industry	Motion Control
Mining	Navigation
Agriculture	Path Planning
Bomb Disposal	Obstacle Avoidance
Volumetric Mapping	Sensing
Environmental Monitoring	World Modeling
Reconnaissance, Surveillance	Vision
Inspection of Hazardous Areas	Mapping
Search and Rescue	Localisation
Remote Exploration	Exploration
Military	Communication
Intelligent Vehicles	Autonomy
Medicine	Cooperation
Domestic Uses	Learning
Entertainment	Reasoning

Table 1.1: Non-comprehensive lists of the applications and capabilities that attract much research today. This thesis deals primarily with those robot capabilities highlighted in blue on the right, and the resulting multi-robot exploration approach is most relevant to those applications highlighted in blue on the left.

Chapter 2

Literature Review

The literature review has been divided into two halves.

In the first, four robotics application domains are presented to which the research discussed in this thesis is most relevant: Reconnaissance & Surveillance; Inspection of Hazardous Areas; Search and Rescue; and Remote Exploration. For each of these, a brief overview of the state of the art is presented along with a couple of selected examples of recent research, followed by a discussion of the main challenges involved.

In the second half, the specific task of multi-robot exploration is examined in greater detail. This part of the literature review is organised by the classes of methods that have been attempted, and several recent examples are provided for each. Finally, several of the approaches most relevant to this thesis are compared in greater detail.

2.1 Relevant Applications

2.1.1 Why Robots?

The potential advantages of robots over humans for certain tasks have been expounded by roboticists for many years now:

- Robots can be more mobile. Robots can be built to fit into small places or to reach areas otherwise inaccessible to humans or search dogs.
- Robots are expendable.
- Robots can have greater sensual perception. Depending on the sensor payload, robots can learn considerably more about their environment than a human could (*e.g.* using chemical or infrared sensors, or computer vision techniques).
- Robots can be more robust. They can be designed to cope with heat, radiation and toxic chemicals, and can operate in many environments too dangerous for humans.
- Robots can be stronger. Depending on the task and environment, robots can be designed to exert greater force or move faster than humans could.
- Robots can be very intelligent. In certain situations robots can make autonomous decisions faster, more efficiently and more intelligently than humans[19].

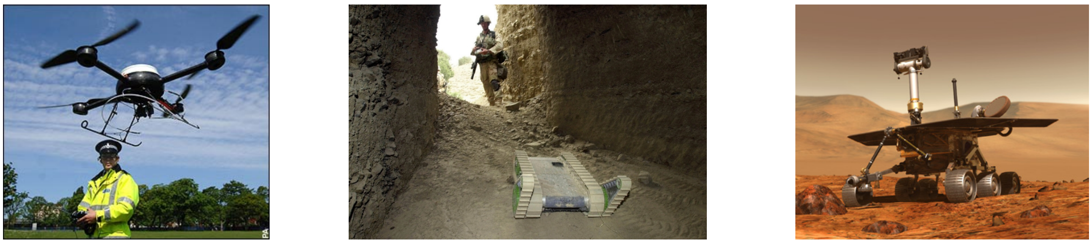


Figure 2.1: Robots are already being used for surveillance (left), inspection of hazardous environments (centre), and planetary exploration (right)

2.1.2 Reconnaissance and Surveillance

Reconnaissance and surveillance are useful not just for the military and security forces, but in various civilian applications as well. Possible scenarios that are commonly cited include hostage situations, urban conflicts, intruder detection, and tracking [124, 89, 8, 66]. Reconnaissance involves the use of robots to find new information about environments of interest that are too dangerous, large, or remote for humans to do the job. Surveillance involves monitoring an environment in order to learn about changes or movement within.

An early multi-robot system developed by Rybski *et al.* uses larger "Ranger" robots to transport a number of smaller "Scout" robots [124]. The ranger can cover large distances to reach a location of interest, and then launch the scouts into further territory, *e.g.* through windows of a building. The scouts carry a wide array of sensors and can both roll and jump, returning information to the ranger for processing.

In a similar approach, a suite of "ThrowBots" have been described by Barnes, Everett and Rudakevych [8]. Developed primarily for military purposes, the ThrowBots can be tossed down a corridor, up a stairwell, or into a window, and thereafter can be remotely operated and provide camera feedback on the interior. Several models have been tested, including a 4-wheeled platform and a 6-wheeled platform, and future development of a track and flipper based model is planned.

In an unprecedented study, Howard, Parker, and Sukhatme deployed 80 robots into an unexplored building to map it, detect and track intruders, and transmit this knowledge to a remote operator [66]. The team is divided into a small number of "highly capable" robots and a large number of "simple" robots. A follow-the-leader paradigm is used to deploy the team into the environment, and an acoustic sensor network deployed by the team detects targets with 100% accuracy.

Recently, unmanned aerial vehicles (UAVs) have received more and more attention as tools for surveillance and reconnaissance. UAVs have already been used by police for crowd control and to make arrests [69]. Research is also being conducted into multi-UAV target tracking – for example the SUAAVE (Sensing, Unmanned, Autonomous Aerial VEhicles) consortium, of which Oxford University Computing Laboratory is a member, focusses on the use of UAV teams to search for a given target, such as a lost hiker in open countryside [139].

The main challenges of reconnaissance and surveillance depend on the task at hand, and the platforms used. However, typically the challenges involve accurate sensing (which can depend on localisation, mapping, and possibly computer vision), robust design and navigation (particularly where unorthodox entry such as being thrown is involved), and strong communication (since information that doesn't reach human responders is useless). When teams of robots are involved, autonomy becomes more and more important and communication and coordination of the team must be managed carefully.

2.1.3 Inspection of Hazardous Areas

There is much overlap between the reconnaissance task and inspection of hazardous areas. However, whereas targets in the previous section were typically human (intruders, criminals), this section focusses more on environmental hazards. There has been much work in this area, as robots have already been used for landmine detection, explosive ordnance disposal, fire-fighting, and inspection of nuclear power plants and waste storage pools [137, 146, 12, 102], to name but a few. The typical task for robots in such scenarios is to inspect, and often act on, an environment that is too dangerous or toxic for humans.

The use of robots in hazardous environments has been discussed for decades. Already in 1992, Stone and Edmonds describe the development of a robot that can identify hazardous material (such as toxic canisters) and mitigate instances involving chemical releases [137]. The robot can further navigate to locations of interest and unlock and open doors when necessary.

Robots have also been used extensively for explosive ordnance disposal, de-mining and detection of landmines. For example, Tojo *et al.* have developed a wheeled buggy that has high mobility and can be taken to places that are hard for other de-mining equipment to reach [146]. The buggy has a special field arm mounted that is counter balanced and allows detection of mines from a distance, even on sloping terrain.

Bengel *et al.* have demonstrated the potential use of a mobile robot on an offshore oil platform [12]. The robot can be used for a variety of remote inspection tasks, such as monitoring of gauges and meters, inspection of valves, and inspection of leakage. Given that offshore environments can be dangerous and (in severe weather) inaccessible, the authors argue that there is great scope for offloading work onto robots and thereby improving operational efficiency and safety.

Nawaz and colleagues are working on a robotic system for monitoring of nuclear waste storage pools [102]. Such environments can contain significant challenges given that the environment is quite cluttered, making localisation difficult. Nevertheless, a successful system would provide helpful information about the state of the spent fuel rods, and the system as a whole is applicable to other domains, such as inspection of settling ponds of chemical plants and inspection of sewage treatment facilities.

Recently the world has become acutely aware of the potential of robots in hazardous zones due to the March 2011 Tohoku Earthquake, and the ensuing collapse of the Fukushima nuclear reactors. Japan came under criticism for being the world's most advanced robotics nation, yet not having a single robot available for use at the Fukushima plant [62][1]. It seems likely that there will be increased interest and spending on robots for such scenarios in the near future.

Summing up, robots used for inspection of hazardous areas typically need to be robust and well designed for their specific task. Autonomy is a desirable quality,

[1] although, as Dr. Robin Murphy points out, this is an unfair criticism since most other nations would not be prepared for a catastrophe of that scale either [98]

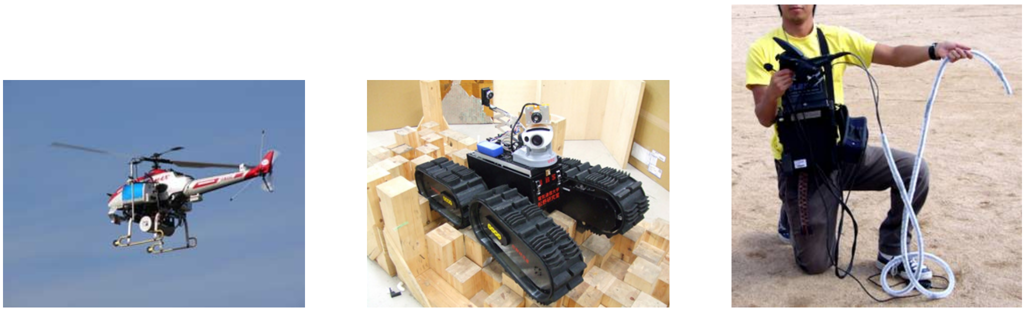
particularly for mundane tasks, navigation, and obstacle avoidance. The return of information to human responders is paramount, so communication is of great importance.

2.1.4 Search and Rescue

Search and rescue robots are used to aid human responders in the search for victims of disasters. Such disasters could include buildings collapsing after an earthquake, mines becoming inaccessible after a landslide, or urban disaster zones after terrorist attacks. The field of rescue robotics contains a rich set of challenges [84, 132]. Tasks can include search, reconnaissance, mapping, rubble removal, structural inspection and communication relay. Environments are typically unknown and complex and communication is typically poor and unpredictable.

Rescue robotics has now attracted research on an international scale, but the earliest efforts were made in Japan and the United States. In Japan, the response to the Great Hanshin Earthquake of 1995 included provisions for the "Development of Advanced Robots and Information Systems for Disaster Response" [140, 88]. The project led to the creation of the International Rescue Systems Institute, and widespread development of robotic platforms for search and rescue (Fig. 2.2).

In the United States, the development of rescue robots was kick-started by the

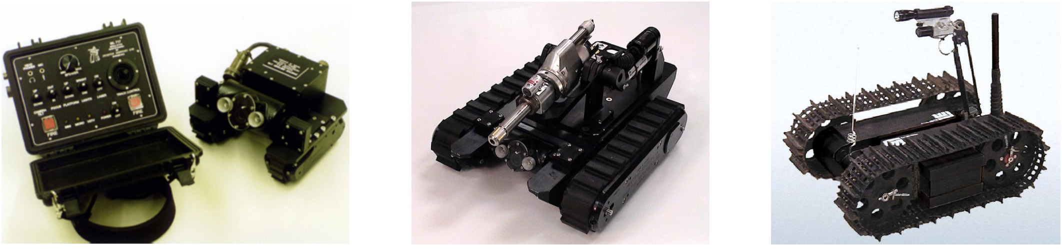


(a) Intelligent Aero-robot (Kyoto University)

(b) Kohga 2 (University of Electro-Communications)

(c) Active Scope Camera (Tohoku University)

Figure 2.2: Robots developed in Japan for search and rescue



(a) Inuktun Microtracs (b) Inuktun Minitracs (c) Foster-Miller SOLEM

Figure 2.3: Some of the robot models taken to the site of the 2001 World Trade Centre collapse by CRASAR. Images extracted from [93] and used with permission.

Oklahoma City bombing of 1995, which occurred only four months after the Great Hanshin Earthquake [32, 126]. Robots used at the disaster site proved to be too large, heavy, slow and clumsy to deal with anything other than bomb disposal. In response, the University of South Florida's CRASAR (Centre for Robot Assisted Search and Rescue) Institute was created, which is now considered a world leader in the field of rescue robotics.

The first opportunity for CRASAR to deploy its robots in a real urban search and rescue situation (and indeed, the first use of rescue robots ever) came on 11 September 2001 when the World Trade Centre collapsed in New York City [93, 23]. A variety of robots were deployed in eight separate drops (see Fig. 2.3). While the robots did not find surviving victims they did help with locating ten bodies. The exercise further provided valuable insight into the skills required by both robots and humans, and pointed out shortcomings that require most urgent attention.

There has also been much literature recently on the use of teams of unmanned aerial vehicles (UAVs) for the creation of an adhoc communication network infrastructure [122, 59]. UAVs are fast, quick to deploy, and can benefit from line-of-sight communication. Such networks could be used not only for coordination of a robot team, but also for all rescue responders to communicate and coordinate their efforts.

An overview of some recent rescue robot deployments at real disaster scenarios is presented in Table 2.1.

Rescue robots have been used at a variety of sites, in various different ways. The following is a non-comprehensive list of recent deployments:

September 2001	New York City: Following terrorist attacks, the World Trade Centre collapses. Ground robots are used to enter voids in the wreckage to search for survivors. No survivors are found, but ten bodies and a number of entry routes are located [23].
October 2004	Niigata, Japan: After the Niigata Chuetsu Earthquake (magnitude 6.9), the Souryo snake robot is tested in the resulting wreckage [132].
January 2005	La Conchita, US: A mudslide destroys 18 homes and kills 10 residents. Inuktun VGTV Extreme robots are deployed in two buildings but fail due to the density of the material in one case and the difficulty of mobility in the other [101].
September 2005	New Orleans, US: In the aftermath of hurricane Katrina, small camera-equipped ground based robots and helicopters are used to examine structures at risk of collapse [71, 58].
August 2007	Utah, US: A mine at Crandall Canyon collapses, trapping six miners. A customised Inuktun robot is lowered 430 metres through a bore hole to inspect the condition of the mine [100].
March 2009	Cologne, Germany: Following poor underground tunnel work a modern multi-storey building collapses and two people are trapped. Rescue robots are taken to the site but the voids are too small even for the smallest robots [83].
January 2010	Haiti: Following a major earthquake, Global Hawk UAVs (Unmanned Aerial Vehicles) are used to provide aerial imagery in support of humanitarian efforts on the ground [116].
October 2010	Croatia: An underwater robot is used to search for remains of balloonists suspected to have crashed in the ocean [91].
November 2010	Pike River, New Zealand: An explosion in a mine leaves 29 miners unaccounted for, but there is too much poisonous gas for rescue responders to enter the mine. Several military robots enter the mine, taking footage of a dropped helmet [61].
March 2011	Tohoku, Japan: Following the greatest earthquake in Japan's recent history, a number of coastal towns are washed away by a tsunami and a nuclear power station collapses. Robots are used to find victims underwater, and American "PackBots" developed by iRobot are used to detect radiation levels at the power station [99, 57].

Table 2.1: Recent rescue robot deployments

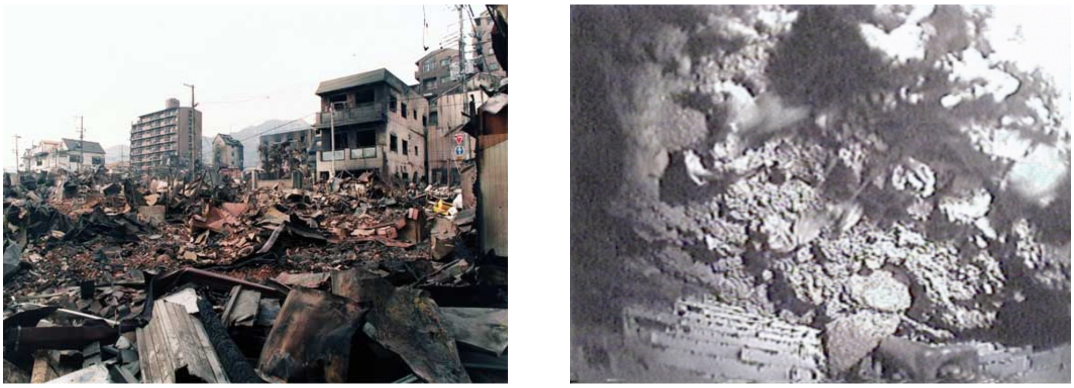


Figure 2.4: Devastation resulting from the Great Hanshin Earthquake (left) and a void into which robots were dropped at the scene of the World Trade Centre collapse (right). Finding victims in such environments is extremely difficult.

The rescue robotics task is clearly a very difficult one. Rescue robots need to be small so that they can be carried into complex sites. At the same time, they need to be highly mobile to traverse a variety of obstacles, since rescue environments are typically complex (Fig. 2.4). A variety of prototype platforms have been developed for this purpose, from snake-like robots to tracked vehicles with flippers (Fig. 2.5).

Further desirable features include advanced image processing systems, tether management systems, wireless relay systems, localisation and mapping, and assisted navigation [93]. In other words, a degree of autonomy is of great interest.

2.1.5 Remote Exploration

A final application domain worth discussing is that of Remote Exploration: using robots to explore environments that are too far or too difficult for humans to reach.

Perhaps today's most famous robots are the robotic rovers that have been used with great success on Mars. At latest reading, the rover Spirit has traversed more than 7.7km, while Opportunity has traversed more than 29.3km of terrain [108]. Since sending humans to other planets or asteroids is enormously expensive and risky, the great value of space robotics has been recognised, most recently in the Obama

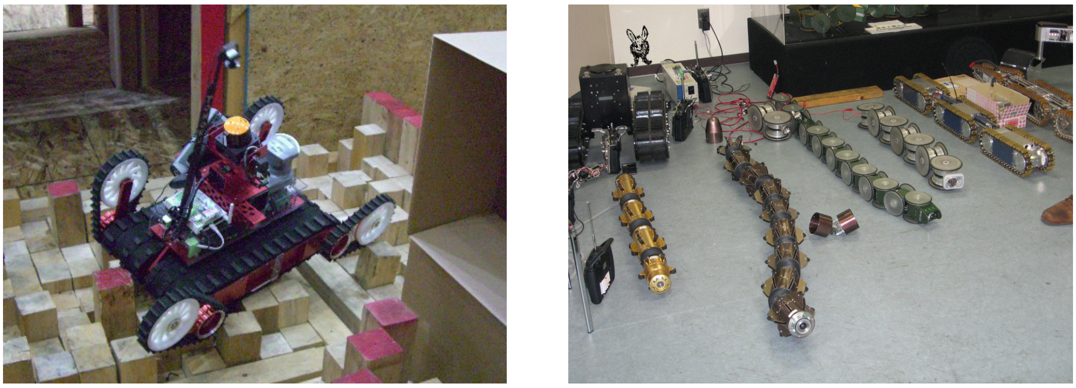


Figure 2.5: Recent rescue robot designs. The Kenaf robot (left), developed in part at the University of Tohoku, Japan, has two main tracks and four tracked “flippers” which allow it to climb stairs and traverse difficult terrain. At Prof. Matsuno’s laboratory in Osaka University, there is a strong focus on snake-like robots that are becoming ever smaller and more mobile (right).

administration’s new space budget [106, 25], and several future rover missions are in planning stages. The goals of Martian exploration are to determine whether life ever arose on Mars, characterise the climate of Mars, characterise the geology of Mars, and prepare for human exploration [107].

Remote environments exist on Earth as well. Less is known about the deep oceans than about the surface of the moon, and autonomous underwater vehicles (AUVs) have been used to examine underwater environments already for decades [159]. The purpose of such exploration extends beyond simple information gathering, being useful also for monitoring health of marine environments, dispersion of pollutants, and surveillance of ports or underwater engineering projects. In one recent example, a team of two different robots and a number of sensor network modules were deployed underwater, demonstrating docking, communication, and coupled motion over a large number of experiments [40]. Robots were also used with mixed success to fix a leak in an oil well in the disaster on one of BP’s oil rigs in April 2010 [54].

Other remote environments that can be explored by robots are caves and mines. Mines in particular can contain gases dangerous to humans or may be at risk of collapse. Several robots have already been used in mines [143, 95] and the potential use of robots for volumetric mapping has been demonstrated [145].

The main challenges in the domain of remote exploration are autonomy, navigation, mapping, and information relay. As with the other domains discussed previously, when teams of robots are used, coordination and communication management become more and more important.

2.1.6 Common Requirements

Clearly there is much overlap between all of the above application domains, and they share many common requirements, including the following:

- Information gathered by the robots must often be relayed to human responders or to a base station where it can be analysed.
- Communication is not ubiquitous and can drop out or be unpredictable.
- Autonomy of the robots can help take the load off human operators.
- When multiple robots are used, they need to be coordinated in order to prevent redundant task completion.

The remainder of this thesis deals with exploration algorithms for teams of robots in communication-limited environments and it is hoped that the methods proposed are applicable, to some degree, to each of the applications discussed here.

2.2 Multi-Robot Exploration

While the previous section dealt with robotics *applications*, this section provides a literature review of a particular robotics *capability*: exploration. In particular, it examines current approaches to the problem of multi-robot exploration, *i.e.* using multiple robots to explore an unknown environment. Following a brief history of the general development of robotic exploration, the review is organised into several categories that current approaches can be classified by, and multiple examples are provided for each category. Finally, a handful of approaches most relevant to this thesis are compared in greater detail.

2.2.1 A Brief History

Robotic exploration has been an active area of research since the beginning of robotics itself, from the mid-20th century onwards. Early approaches in robotics attempted to create internal models of the world, but since the mid-1980's there has been greater focus on behaviour-based and reactive robotics (see e.g. [15, 7]). The latter approaches allow robots to adjust with greater speed to complicated and dynamic environments. In both cases, however, exploration is closely tied to mapping, and in many studies these two fields are closely intertwined. It is hard for a robot to explore well without a useful map of its environment, and it is likewise difficult for a robot to map an environment without effective exploration.

2.2.2 The Advantages of Multiple Robots

Initial efforts in robotic exploration involved only single robots (e.g. [39, 78, 160, 144, 157]). Due in large part to an increased availability and functionality of communication hardware and techniques, interest shifted in the early 1990's to multi-robot exploration. While there are disadvantages to using multiple robots (such as collisions, interference, and complexity of coordination), numerous advantages to using a team, instead of single robots, have been identified throughout the literature [20, 117, 4, 110, 37, 55, 163]:

- several robots can cover more area in less time than a single robot
- use of multiple robots can lead to improved localisation through mutual observation
- if properly implemented, a team of robots may be more robust than a single robot due to redundancy and elimination of single points of failure
- heterogeneity can be of advantage – different robots within the team may be particularly suitable for certain subtasks of the mission
- multiple robots can exchange sensor information and sense the environment from multiple viewpoints
- depending on the task, multiple robots may help each other overcome obstacles or collaborate to manipulate objects in the environment
- it can be easier to build and maintain many simple robots instead of a single complex robot

Several early studies in multi-robot exploration focused on motion planning and collision avoidance (for a review see [81], more recent examples include [51, 13]). More recently, however, the emphasis in multi-robot exploration has been on coordination and cooperation, which ties this field to distributed artificial intelligence, multi-agent systems, and networking.

In the remainder of this chapter, only those approaches that involve multiple robots exploring together are considered.

2.2.3 Line-of-Sight and Leader-Follower Approaches

The term "line-of-sight" may refer either to visual detection (used in earlier systems) or wireless line-of-sight, *i.e.* being within direct communication range. Leader-follower approaches involve one or multiple robots directly following a lead robot.

In an early approach, Rekleitis, Dudek and Milios propose an exploration algorithm in which one moving robot is observed by two stationary ones, with the resulting triangle between them considered empty for mapping purposes [117]. While this behaviour isn't governed by a central agent, the robots still behave according to a centrally agreed plan. The robots detect one another using a range sensor. Exploration is faster than it would be with a single robot, but the algorithm requires at least half of the robots in the team to remain stationary at any point in time, limiting efficiency. In a later study, the same authors propose a system whereby pairs of robots track the environment and one another, for improved mutual localisation [118]. Robots can detect markers on one another, and only one moves at a time, with the area between the two considered clear space (much like their previous study). If an obstacle impedes the line-of-sight, the moving robot backtracks and navigates around the obstacle. While this approach was a leader in its time, the advent of accurate laser range scanners means that much better methods for mapping and exploration now exist.

As part of an investigation into how a team of robots can self-organise for exploration of an environment, Arkin and Diaz propose three different behavioural strategies, all of which maintain line-of-sight communications between robots of the team [3]. The first, "anchored wander", involves a single robot acting as a communication anchor, and the rest of the team entering the environment one at a time. In the second, "quadrant-biased anchored wander", a bias is introduced which makes robots wander towards an area of particular interest. In the third, "informed exploration", the team of robots relies upon map knowledge to disperse themselves along an optimal path. This is one of the first studies to explicitly make team connectivity a priority; however the exploration approach is not highly efficient since many robots remain stationary during the exploration effort.

Sgorbissa and Arkin assume a dynamic, complex, communication-limited environment for their experiments [129]. In their approach, each robot is assigned a set of locations that must be reached. Robots may help one another using line-of-sight communication by either sharing goals (*e.g.* if one robot reaches another's goal) or by sharing state (*e.g.* a robot in trouble is attracted by teammates that are not in trouble). Extensive simulation experiments indicate that a role-defined approach such as this works, particularly if 'support robots' are deployed to assist with the effort.

Powers and Balch use "motor-schemas" (based on those proposed in [7]) to preserve line-of-sight communication between members of a team of robots [115]. Their approach, "value-based communication preservation", reactively chooses a direction in which to move at each time step that is most likely to keep the robot in communication range of the rest of the team. Robots make decisions in a distributed manner, but stay in contact at all times. Extensive experiments with simulation and real robots validate the approach.

A practical system is developed by Nguyen *et al.* [105]. Using short-range, high-throughput radios, a lead robot enters an environment followed by several "slave" robots that relay the leader's sensor input to a remote control / monitoring station. The slave robots automatically stop when necessary to maintain an ad-hoc network, and those no longer needed in the network can navigate back to the lead robot to be redeployed.

A similar approach (already briefly mentioned in section 2.1.2), using the leader-follower paradigm, has been developed by Howard, Parker and Sukhatme [66]. Eighty robots are used to map the interior of a building, detect and track intruders, and transmit all of the acquired information to a remote operator. Using a heterogeneous team in an environment built and monitored by external human supervisors, Howard *et al.* employ a decentralised frontier-based approach with local occupancy grids and minimal communication between robots. Although there is much redundancy in the exploration (as robots explore the same location multiple times), Howard *et al.*'s large robot team manages to deploy robots to intended locations with a 60% - 90% success rate, while detecting intruders with 100% success.

Stump *et al.* also aim to keep direct contact from a base station to an exploring robot via multi-hop communication over a set of relay robots [138]. In their approach, the Fiedler value of the weighted Laplacian describing the communication interactions of all robots in the system is used to determine movements of the robots such that a connection is maintained.

The multi-robot routing problem (maintaining multiple robot routers between a base station and an exploring robot) is further explored from a theoretical perspective by Tekdas *et al.* [141]. Algorithms are developed to compute the minimum number of robots required for either communication with an exploring robot, or for coverage of the environment.

In summary, line-of-sight and leader-follower approaches are very good at ensuring reliable communication. Robots are typically deployed in a manner that ensures their communication links to teammates or a basestation remain strong. Having a fully connected team enables full, instantaneous control for any human operators.

However, a drawback is that there are limitations to the extent of the environment that may be explored by such a team. If no robot is allowed to go beyond its parent's communication range, there is an inherent limit to the distance that such a team can be stretched across. In very large or complex environments, some areas may never be reached by teams using leader-follower or line-of-sight paradigms.

Furthermore, in most leader-follower approaches, only a single robot is exploring at any given time (while follower or "slave" robots relay information). There may be many situations in which it is much more efficient to have multiple robots exploring; one way to achieve this is by using frontier or utility based methods, as detailed in the next section.

2.2.4 Frontier and Utility Based Approaches

In frontier and utility based approaches, areas of interest are typically evaluated based on certain criteria (such as expected information gain, path cost, or likelihood of communication availability), and assigned a value. For every robot-frontier pair, there may be a different such value. As a result it is straightforward to assign robots to different areas of interest, and thereby have multiple robots contributing to the exploration effort at the same time.

A landmark paper by Yamauchi in 1998 laid the foundations for frontier-based exploration, with the first definition of "frontiers" [157]. Frontiers are the boundaries between explored space and unexplored space, where explored space has been sensed with a range sensor. Robots can navigate towards unexplored frontiers. When teams of robots are involved, they share information each time they arrive at a new frontier, and teammates' observations are incorporated into each robot's individually maintained map.

In an extension of Yamauchi's frontier-based exploration, Simmons, Burgard and colleagues propose a method in which robots construct 'bids' based on estimates of expected information gain and travel cost to locations of interest (using frontier cells to calculate each of these values) [134]. The bids are sent to a central agent which evaluates the bids from the whole team and assigns tasks with the goal of maximising utility. Individual robots make their own maps and these are joined together by the central agent. Maximum likelihood estimates are used for both localisation of self and of objects in the environment, and the approach is fully tested with a team of three heterogeneous robots. In an extension of the approach, robots no longer send bids to a central agent but use a robot-to-frontier assignment algorithm that finds the best set of frontier-robot pairings for improved exploration [17].

A similar idea is used in a much later version of Burgard *et al.*'s work [18]. Value iteration, a dynamic programming algorithm, is used to calculate costs to frontiers, with penalties applied to narrow passages or obstacles. Frontier cell utilities decrease when other robots are near to them, which leads to better coordination of the team to different frontiers. To take communication drop out into account, the robot-to-frontier assignment is not calculated centrally, but by any connected subgroup of robots. Extensive experiments both in simulation and with real robots indicate that the approach works well. A further extension takes into account communication limitations [90]. To decrease bandwidth requirements, robots share only approximations of their respective maps, using bounding polygons. Targets are assigned to robots in a distributed manner, and extensive simulation tests reveal that the approach significantly decreases communication volume while maintaining strong exploration.

Fox *et al.* also use frontier-based exploration to drive their multi-robot system [48]. Robots are not assumed to know one another's locations initially, and may start from entirely different locations in the environment. Once two robots encounter one another, they share all sensor data. Since repeated encounters allow the robots to confirm hypotheses about one another's locations, they aim to repeatedly rendezvous. Once hypotheses are confirmed, robots form an exploration cluster and continue exploring together. Robots may either explore new areas or attempt to confirm hypotheses – for each of these tasks a utility is calculated that takes into account frontier size and path cost in the former case, and probability of hypothesis and path cost in the latter.

Rooker and Birk propose a similar idea in which frontier-based exploration is applied to multiple robots, but there is no central base station, resulting in "robot pack" behaviour [119]. A communicative exploration algorithm is applied that keeps the network structure of a group of robots intact, allowing this group to explore freely. Robot behaviour is guided by a state heuristic that involves rewards for frontier cells and penalties for locations that are impossible to reach or out of range. The communication component of this heuristic depends on a list of neighbours (connected teammates) for each robot, and consequently there is an incentive for each robot in the group to stay near to other robots in the group while searching out frontiers. The result is roaming packs of robots, where every member of a pack is connected to every other member of the same pack. In a later paper the use of an immobile base station is discussed. This may be used as an anchor for an initial length of time (and likelihood of communication is factored into frontier utilities), before the robots split off from the base station to explore in packs [120].

Trade-offs in the calculation of frontier utilities have been explored by Visser and Slamet [152]. When more value is placed on information gain, robots are more likely to explore hallways; when more value is placed on reducing path cost, robots are more likely to explore rooms. In this way the team behaviour can be tuned by a simple parameter. Visser and Slamet further take communication into account when calculating the information gain [148]. The degree of a robot's interest in a particular frontier is based not only on potential area to be discovered and path cost, but on likelihood of communication success from that area. Strength of communication with the base station is measured at regular intervals and stored in a quad-tree. Signal strength at frontiers can be estimated using a nearest-neighbour technique, and consequently robots are unlikely to stray far from the team's communication range.

Recently, the frontier-based paradigm has evolved to include more specific environmental information. Stachniss *et al.* use semantic place labelling to determine features of the environment that can be used instead of frontiers [136]. An AdaBoost algorithm is used to train a classifier to determine whether a given scan belongs to a corridor or not. Unknown space is further partitioned into "segments" by Wurm *et al.* [155]. Segments are determined using a Voronoi graph of the free space and determining critical points (typically in doorways). An adaptation of the Hungarian method [77] is then used to optimally assign robots to segments.

In summary, frontier and utility based approaches typically make it straightforward to guide robots, either centrally or in a distributed manner, to new, unexplored areas. Each robot can make a significant contribution to the exploration effort, and typically exploration proceeds faster than in leader-follower type approaches.

The drawbacks of frontier-based methods are an inability to deal with limited or failing communication. Robots may choose to explore in opposite directions, and may soon be out of one another's range. Thus there can be a risk of redundant work, as some team members may end up re-exploring rooms that their teammates have already discovered. Variants of frontier and utility based approaches (such as [119, 120, 90, 148]) attempt to take this into account by factoring communication into utilities. However, typically this means that robots either (i) only explore frontiers that are within range of teammates or a basestation, meaning that some areas are never explored; or (ii) explore together as a group, staying within range of one another, but not maintaining a link to any central basestation.

In utility-based approaches, robots are assigned to frontiers by a deterministic mechanism. This has also been achieved in a different way, however, using market principle based approaches, as the next section explains.

2.2.5 Market Principle Based Approaches

In recent years, multi-robot systems research has drawn inspiration from economics. This has been achieved both in terms of competition (when individual robots try to maximise their personal gain, the team effort as a whole can benefit) or in terms of auction-like behaviour (where robots can bid on subtasks based on their desirability).

Dias and Stentz propose a free market architecture for distributed control of a multi-robot system [37]. In a paper that draws examples from international market systems, Dias and Stentz suggest that an approach in which individual agents try to maximise their personal gain will lead to optimal global behaviour. Their agents try to maximise profit by using revenue and cost functions to determine potential profit of individual tasks. In a simulation where robots have full prior knowledge of the environment, robots bid on cities (or points of interest) in the environment to visit. The emergent cooperative behaviour is an efficient exploration of the environment.

Gerkey and Matarić suggest an auction-based task allocation system, MURDOCH, which is built upon a principled, resource centric, "publish/subscribe" communication model [53]. Based on the much earlier proposed Contract Net Protocol [33], this system uses a novel communication protocol in which messages are passed by subject rather than by intended recipient. The system as a whole is tested in two situations: a tightly coupled multi-robot physical manipulation task, and a loosely coupled multi-robot experiment in long-term autonomy. In both cases the approach turns out to be reactive to changes in the environment, including abrupt failures of robots and random introduction of new tasks.

Zlot *et al.* also use a market-based approach to multi-robot exploration that does not rely on perfect communication, and is still functional with zero communication (apart from initial deployment) [163]. Robots explore by visiting a set of goal points in unknown regions. Each robot produces a tour containing several of these points, and subsequently the tours are refined through continuous inter-robot negotiation. A revenue function involving information gain determines robots' profits, as does a cost associated with the resources required by the individual robot to obtain that information. The revenue is paid out by a central coordinating agent, but otherwise the agents act in a completely distributed manner and may be removed or added to the effort in a dynamic manner. Experiments with P2DX robots validate the approach, and both random and quadtree approaches outperform a greedy exploration approach.

Unlike the above approaches, Sheng *et al.* propose an approach in which there is no central planning agent [130, 131]. Individual robots' bids are broadcast to all team members using a novel communication mechanism that reduces the exchanged data volume. Bids are placed on frontier cells and evaluated in discretely timed, distributed auctions. The bids themselves are based on information gain (which is itself a measure of the current local map, current positions of other sensing-and-mapping robots within the same subnetwork, and target cell positions of traveling robots within the same subnetwork) and path cost (calculated using Dijkstra's shortest distance algorithm on a connectivity graph derived from the map). Simulation results suggest that this approach does reduce communication overhead and can lead to a clustering behaviour, where groups of robots tend to stay together.

While market principle based approaches show promise regarding the management of robot teams, they do require careful coordination – either by a central agent, or by carefully timed distributed auction mechanisms. Perhaps for that reason, utility and frontier based approaches have been more popular in recent years, being both easier to implement and equally effective.

Another class of methods that typically involve more theoretical analysis comes from the domain of graph theory, as detailed in the next section.

2.2.6 Graph Theoretic Approaches

Graph theoretic approaches use methods and ideas from the mathematical domain of graph theory to coordinate teams of robots. The correlation between graphs and robot teams is straightforward: robots are usually represented as nodes in the graph, and edges typically represent communication links. There is a wealth of proofs and useful conclusions in the area of graph theory, making it an attractive source of ideas for multi-robot coordination algorithms.

Basu and Redi are among the first to model multi-robot systems as a graph [9]. They argue that biconnectivity is a desirable characteristic of any adhoc network established by a team of robots. If the team is biconnected, failure of one robot does not interrupt communication between any other pair of robots in the team. Several simple algorithms are proposed to direct team members in such a manner that biconnectivity is achieved.

Vazquez and Malcolm propose a completely distributed behaviour-based architecture, whose behaviours are meant to keep the robots in a mobile ad-hoc network [147]. Using graph theory fundamentals involving articulations and bridges, the authors impose constraints on robot motions to maintain a connected network. Each robot maintain its own "comfort zone" within which it is well-connected to the team, and the behaviour architecture ensures that robots do not leave their comfort zones.

Yao and Gupta apply a connectedness constraint to a multi-robot team, and extract the "backbone" of the team by using graph theoretic principles to classify the robots into 'clusterheads', 'doorways', and 'gateways' [158]. Non backbone robots use a leader-follower behaviour to remain connected to the backbone. The team remains fully connected at all times, even when obstacles are introduced into the environment.

A "sensor-based random graph" provides the basis for work done by Franchi *et al.* [49]. In such a graph, a node contains the local safe region while arcs represent safe paths between nodes. Arcs are established either by having been travelled by robots, or by joining of two adequately close existing nodes. Actual directions of travel are chosen by looking at frontier cells in the local node configuration, similar to [157]. In a similar approach, Juliá *et al.* develop a graph to represent safe zones and frontiers as the exploration effort unfolds [72]. A reactive method is used to travel from one safe zone to the next.

Mukhija *et al.* employ an approach that is in some respects similar to the ideas proposed later in this thesis [97]. A fixed base station is assumed, and robots act either as relay nodes or as explorer nodes within an exploration tree. Each robot can travel in one of a discrete number of directions, and new robots are added to the tree according to a baseline algorithm that behaves similar to depth-first tree traversal. However, nodes must remain static while behaving as relay nodes.

While many real-world factors can be difficult to model in a graph (*e.g.* exact path costs, or unexpected communication drop-outs), coordination of multiple robots seems a natural candidate application for graph like methods. Graph-based methods have been applied with great success to the problems of mapping (where each set of sensor readings can correspond to a node) and navigation (where graphs can be used for path planning), and it seems likely that there will be increased research in this direction for finding solutions to multi-robot exploration in the near future.

Approaching the multi-robot exploration problem from quite a different angle, several groups have proposed the use of miniature electronic tags during the exploration effort, as detailed in the next section.

2.2.7 Exploration with Environmental Tagging

In recent years miniature electronic tags that can store information have emerged as a highly useful technology, and there is great potential for exploring robots to drop such tags as they explore and thereby assist their teammates. Oft-cited advantages of tags are that they are cheap, small, can help with localisation, and can be used by human responders to find a path in an explored environment. As a result, there is a wealth of research regarding the use of tags for assistance with multi-robot exploration.

Kleiner, Prediger and Nebel propose an approach in which radio frequency identification (RFID) tags are dropped at suitable locations during the exploration process (nodes corresponding to unique sensor measurements in the common scan-matching algorithm originally proposed by Lu and Milios [85]) [73]. Tags store the relative locations of frontier cells and visited cells, which allows robots passing later on to update their information and explore more effectively.

Ziparo *et al.* simulate a team of robots deploying RFID tags that can be used to significantly reduce the size of the search space [162]. Robots deploy these tags autonomously, and once deployed the tags act as coordination points. The overall approach is hybrid centralised/decentralised: robots plan their paths locally, but a global coordination mechanism periodically moves robots to new interesting locations to prevent them from being trapped in local minima.

Ferranti, Trigoni and Levene propose "Brick & Mortar", a fully distributed approach in which agents tag the environment as they explore [44]. By reading information left on tags by teammates (e.g. whether a cell is *unexplored*, *explored* or *visited*), agents know which direction to proceed in to make the exploration as efficient as possible. In extensive simulation the authors demonstrate that this approach outperforms both multiple depth first search and Ants [76]. An extension of this approach proposes evacuation route discovery methods that can proceed in step with the exploration effort [43].

While the use of such tags brings with it additional challenges (such as the engineering of a tag dropping mechanism, and the ability to find and read tags), it seems that there is great scope for such methods to improve the efficiency of exploration, and it seems likely that there will be more research in this direction in the near future.

2.2.8 Coverage

Various authors approach the multi-robot exploration problem from a theoretical angle, in particular regarding solutions to the so-called *coverage problem* [27]. This problem can be defined as “visiting each location in known terrain to perform a task” [14]. While solutions to this problem are important for such applications as robotic lawn mowing, vacuuming, harvesting or mine clearing, they are not as relevant to robotic exploration, where robots need to cover terrain sensorially but not physically. Low-cost high-quality sensors such as laser range finders and cameras mean that robots do not need to sweep entire terrains to fully investigate them. Nevertheless some of the approaches to the coverage problem may be of relevance so it is briefly discussed here.

Extensive research into the coverage problem has been conducted by Koenig, Szymanski and Liu, using a paradigm based on ants, or ant robots [76]. Stating that ants “are simple to design, easy to program, and cheap to build. This makes it easy to deploy groups of ants and take advantage of the resulting fault tolerance and parallelism” [76], Koenig *et al.* study and implement several algorithms, including learning real-time A* and Node Counting. The former, guaranteed to be polynomial in the number of locations, outperforms the latter which can be exponential in the square root of the number of locations.

Building on the idea that terrain coverage is an instance of multi-robot spanning tree coverage [60], Zheng et al. prove that the multi-robot coverage problem is NP-complete and propose a Multi-Robot Forest Coverage algorithm [14]. In this algorithm, the environment is represented by a graph, and a rooted tree cover is found where the roots are vertices of the graph containing robots. Each robot must then circumnavigate its tree. The polynomial computation time of this approach is an improvement on previous approaches.

Agmon, Hazon and Kaminka extend this field with an extensive discussion of the nature of such spanning trees and how they may be constructed in order to minimise the time required for coverage [1]. According to their simulation results, non-backtracking multi-robot spanning tree coverage usually outperform other algorithms. In all cases, relatively simple, grid-based environments are assumed, as is prior knowledge of the environment.

A different sort of coverage is examined by another set of authors: communication coverage. In this problem domain, the goal is to arrange a team of robots in such a manner that they are within communication range of the largest possible space.

Poduri and Sukhatme aim to deploy a team of robots in an unknown environment such that each robot has a fixed number of neighbours, and propose an approach based on artificial potential fields [114]. Interaction between individual members of the team is governed by a repellent force that causes team dispersion, and an attractive force that prevents communication loss.

Esposito and Dunbar establish a set of control laws for swarms of robots to navigate through obstacle-filled environments while always maintaining connectivity [42]. In their approach, all robots of the team must pass on the same side of a given obstacle to maintain connectivity. The behaviour is shown to work both in simulation and on a team of Koala robots.

Further control laws to drive robots toward desirable sensing configurations are developed by Schwager *et al.* [128]. Given knowledge of relative sensory importance of regions of the environment, robots navigate towards estimated centroids of their respective Voronoi regions. The approach is demonstrated in a physical system [127].

The Coverage problem is a well studied field of research and many of these solutions show great promise. However, for many applications, such as robotic search-and-rescue, the methods are unlikely to apply: typical robot platforms may not be able to leave a large number of traces, which ants-like methods require; the environment may be too complex or fractured for forest coverage algorithms to be effective; or the environment may be too extensive for potential field like methods to be useful. Thus a need remains for robust multi-robot exploration methods that may not solve the coverage problem, but are useful in other application areas.

2.2.9 Other Approaches

This section details approaches that do not fit into any of the above categories, but have relevance to the research discussed in this thesis.

Pimentel and Campos model a multi-robot exploration team as a mobile ad-hoc network in which all team members try to stay in range of one another and the base [125]. The authors propose a prediction model that receives location and velocity data from all robots in the team and uses this to estimate future network topology. This is accomplished using local so-called "energy functionals" which incorporate aspects relative to network connectivity, environment and task completion. For each robot's local space the goal is to select immediate actions that minimise local energy functionals.

Several authors propose the use of features, rather than occupancy grids or frontier cells, for exploration. Newman, Bosse and Leonard develop an action selection algorithm that steers robots towards open, new areas [103]. Trajectories are planned based on perceived features (such as walls). The location of such features is assumed to be uncertain, and represented by a probability distribution function. Utilities can then be determined for each possible goal using criteria such as trying to visit new, open areas and staying away from previously visited areas. The approach is demonstrated on a B21 robot that accurately explores an indoor environment.

Baxter *et al.* propose a potential field sharing multi-robot system in which individual robots perform no reasoning but a cooperative exploration behaviour nevertheless emerges [10]. Individual robots calculate potential field forces at eight separate locations around themselves, and move in the direction of greatest attraction. Using two separate potential field sharing methodologies (which perform similarly in simulation), robots indirectly communicate locations of obstacles to one another.

Mosteo, Montano and Lagoudakis propose four algorithms based on the same reactive framework [96]. Robot routes are built incrementally, one target at a time, and an underlying motion control mechanism based on the idea of virtual 'spring' forces between robots ensures that the team remains interconnected, much like a mobile ad-hoc network. The four proposed algorithms include a greedy allocation, where robots are assigned to closest targets; a traveling salesman problem-based algorithm, where clusters of goals are assigned to robots and traversed using a travelling salesman problem (TSP) solver; a clock allocation algorithm, which induces a sweeping behaviour using pre-computed polar coordinate-based task allocations; and an auction allocation, which draws on previous methods [79] to preplan the order of tasks according to bidding rules. Simulation experiments demonstrate that the TSP-based and clock algorithms exhibit the smallest sensitivity to actual communication range, making them suitable for situations where the communication range is not known in advance.

Brüggemann *et al.* develop a model for accurate signal strength prediction in yet unvisited areas [16]. The prediction model adjusts online to different environment types and robot systems. The authors argue that knowing future expected communication quality can be used for an improved exploration strategy, though an actual implementation of such is left as future work.

2.2.10 Comparison of Key Approaches

This section provides a comparison of the key approaches that are most relevant to the ideas discussed later in this thesis. Each of these approaches is evaluated along the following five axes:

1. **Type:** Which category of method does the approach primarily fall into? (See the earlier sections of this chapter for descriptions of each.)

    - Leader-follower / Line-of-sight **(LF)**
    - Frontier / Utility based **(FB)**
    - Market-principle based **(MP)**
    - Graph theoretic **(GT)**
    - Tagging-based **(TB)**
    - Coverage-based **(CB)**
    - Other

2. **Communication:** Does the approach assume perfect, flexible, environmental or no communication?

    - **Perfect:** all robots can communicate with all other robots at all times.
    - **Flexible:** robots may come in and out of range of one another.
    - **Environmental:** robots may communicate by leaving messages in the environment.

3. **Coordination:** Is the team coordinated in a central, distributed, or hybrid manner?

    - **Central:** a single control agent governs the behaviour of the team. This agent may be a robot, a command centre, or a human.
    - **Distributed:** agents make decisions on their own based on local information.
    - **Hybrid:** the approach contains both centralised and distributed elements.

4. **Complete**: Does the approach guarantee a complete exploration of the full environment, however large (ignoring power limitations)?

5. **Central BaseStation ("BS"):** Does the approach maintain a link to a base station, or involve relaying information back to some central location?

The full comparison is presented in Table 2.2.

Approach	Type	Comm.	Coordination	Comp.	BS
Arkin & Diaz, 2002 [3]	LF	Perfect	Centralised	No	Yes
Nguyen *et al.*, 2004 [105]	LF	Perfect	Distributed	No	Yes
Howard *et al.*, 2006 [66]	LF	Flexible	Distributed	Yes	Yes
Yamauchi, 1998 [157]	FB	Perfect	Distributed	Yes	No
Simmons *et al.*, 2000 [134]	FB	Perfect	Centralised	No	No
Burgard *et al.*, 2005 [18]	FB	Flexible	Hybrid	Yes	No
Fox *et al.*, 2006 [48]	FB	Flexible	Hybrid	Yes	No
Rooker & Birk, 2006 [119]	FB	Perfect	Centralised	Yes	No
Rooker & Birk, 2007 [120]	FB	Flexible	Centralised	Yes	Yes
Visser & Slamet, 2008 [148]	FB	Perfect	Distributed	Yes	Yes
Dias & Stentz, 2000 [37]	MP	Perfect	Centralised	Yes	No
Zlot *et al.*, 2002 [163]	MP	Perfect	Centralised	Yes	No
Franchi *et al.*, 2009 [49]	GT	Perfect	Distributed	Yes	No
Ziparo *et al.*, 2007 [162]	TB	Env.	Distributed	Yes	No
Poduri & Sukhatme, 2004 [114]	CB	Flexible	Distributed	No	No
Pimentel & Campos, 2002 [125]	Other	Perfect	Centralised	No	Yes
Role-Based Exploration (Chapter 4)	**FB**	**Flexible**	**Hybrid**	**Yes**	**Yes**

Table 2.2: A comparison of the key approaches most relevant to this thesis. Categories and abbreviations are explained in Section 2.2.10.

This concludes the literature review. In the next chapter, the exact nature of the problem tackled by this thesis is defined. Thereafter, Chapter 4 presents a solution.

Chapter 3

Problem Statement

This chapter serves as a blueprint for the experiments conducted in the remainder of this thesis. It sets out definitions and assumptions, lists the exact goals of the exploration effort, and defines five performance metrics that are used to compare and evaluate competing exploration algorithms later in the thesis.

3.1 Definitions and Assumptions

The following definitions and assumptions are made in this thesis:

1. *"Robots", as referred to throughout this thesis, are physically embodied, mobile vehicles with an onboard computing capacity.*

   The robots could be ground-based or aerial – the algorithms apply equally to both. All experiments (simulation and real) involve robots acting and sensing in a plane; the possibility of extending to 3-dimensional environments is discussed later, in Section 8.3.

2. *All robots have the ability to perform simultaneous localisation and mapping (SLAM).*

   In other words, every robot is capable of creating a map of its surroundings and localising itself within this map.

   This capability may be optical or sonar-based, but would more likely involve laser range-finder data, which is generally more accurate. Laser range-finders are very common on ground-based platforms and have recently been demonstrated on a unmanned aerial vehicle (UAV) [56].

   The localisation and mapping problem is not a simple one. In flat, 2-dimensional environments, much success has been achieved, typically using particle filters [74, 48] or scan-matching [85, 113]. In simple 3-dimensional environments, there is already much promise [104, 94, 56, 111]. For complex 3-dimensional environments, such as those encountered in rescue scenarios, much work still needs to be done. However, a team of robots cooperatively exploring will need to know where they are and where to go next, so it is realistic to expect that this is a problem that will see much attention and progress in coming years.

   For the algorithms proposed later in this thesis, localisation does not need to be perfect and there is some room for error. However, robots need to be able to find their way to within communication range of agreed rendezvous points. Frequent rendezvous between team members can be helpful for mutual localisation and map correction [47].

Maps created by the robots must keep track of explored, free space. In the remainder of this thesis occupancy-grid based maps are assumed (as originally proposed by [41]), but the approaches could be tailored to topological maps as well[1]. The notion of free space is essential for calculation of rendezvous points.

3. *At the point of entry there is a "BaseStation", which is immobile and can communicate with the robots.*

   In robotic search-and-rescue, this corresponds to the human responders' point of entry, while in reconnaissance or surveillance this corresponds to the command centre where information is gathered and analysed. Not all multi-robot exploration applications require a central base station, but many do.

4. *All robots begin the exploration effort with a common frame of reference.*

   Robots may enter the environment at different locations or at different times, but for purposes of map sharing they must know their starting location in a common global frame of reference.

   For some applications, it is conceivable that robots would start with separate frames of reference (*e.g.* two robots exploring a mine from opposite entrances). Several map-merging algorithms for such applications exist [48, 22]. However, these applications are not considered in this thesis, although a possible extension to incorporate such scenarios is briefly discussed in Section 8.3.

[1] Occupancy-grid based maps typically maintain spatial information of the environment in a grid, where every cell of the grid corresponds to a specific space in the environment. Topological maps typically keep track of objects or features of the environment, and their relations to one another.

5. *There is no prior knowledge of the environment.*

   Robots only know what they have sensed themselves and what their teammates communicate to them.

   In some applications (such as inspection of hazardous areas) this may not be true, as prior floor plans or the like may exist. Incorporating prior knowledge into the exploration effort is left as future work, as discussed in Section 8.3.

6. *Agents may only communicate when they are within communication range of one another.*

   Communication ranges are simulated in different ways in the experiments reported later in this thesis. These are described in the relevant sections detailing experimental results.

7. *Each robot has the ability to calculate "path costs" of potential trajectories.*

   The notion of path cost is essential for several aspects of the algorithms proposed and compared in this thesis. Unless otherwise indicated, the path cost is the distance of the path between two points of interest. While the real "cost" of traversing this path may not correspond exactly to its distance (for example, a path having many changes of direction could be more difficult to follow than a straight path of equal length), using the distance as a quantitative measure is a useful abstraction that makes calculation of several key heuristics much simpler.

8. *Robots do not act as obstacles for other robots.*

   In the simulation experiments, robot–robot collisions are ignored for purposes of simplicity. Clearly this was not possible in the experiments using real robots; for these, collision avoidance methods were used to prevent robots from running into one another. These are explained in greater detail in Chapter 6.

3.2 Goals

The main goals of the exploration effort are:

1. *To explore the environment as efficiently as possible.*

   Efficiency here is meant in terms of maximising the use of available resources, *i.e.* sensing the maximum amount of free space in the shortest possible time. A related problem therefore is preventing robots from exploring areas that have already been explored by teammates, a common problem in multi-robot exploration.

2. *To relay new information to the BaseStation as quickly and as often as possible.*

   Information is useless if it doesn't reach human responders.

3. *To minimise the time that team members spend out of range of the BaseStation.*

   The more often robots are within range of the BaseStation, the more often updates are received and the easier it is to control the robots manually. Control over the full robot team is highly desirable (for example, if an environment is deemed to have become either highly risky or of minimal interest, and the full team needs to be pulled out).

4. *To fully explore the environment.*

   This may seem an obvious goal, but many exploration algorithms will fail to achieve this (for example those which do not allow robots to explore beyond the team's communication range).

3.3 Performance Metrics

For the evaluation and comparison of individual exploration algorithms, it is helpful to have a common set of easily measurable, objective performance values that indicate the relative success of each approach.

Several metrics have been proposed elsewhere. Mosteo *et al.* use (i) total distance traveled by robots; (ii) mission timespan; and (iii) average task completion time [96]. Zlot *et al.* use a metric that takes into account both area explored and the sum of all robot distances traveled [163]. This thesis builds on the above metrics and proposes some additional ones. The metrics are specifically tailored to investigate to what degree the goals have been fulfilled (efficient exploration, relay of information to BaseStation, team connectivity, and full exploration).

Specifically, the following five metrics are used for evaluation of algorithms in later chapters:

- Total Area:

  $$M_1 = A_{Total}$$

  where A_{Total} is the total area explored (to be maximised). The total area is the union of the areas explored by all robots of the team. “Explored area” refers to the area that has been sensed by a robot using the range scanner.

- Knowledge at BaseStation:

  $$M_2 = A_{BS}$$

  where A_{BS} is the total area known at the BaseStation (to be maximised). Knowledge gathered by members of the team is only useful if it reaches the human responders. While some robots may know about far reaches of the environment, they may not have an opportunity to communicate this knowledge back to the command centre. For applications like search-and-rescue, this metric is considerably more important than M_1.

- Information Sharing:

$$M_3 = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i}{A_{Total}} \times 100\%$$

where A_i is the explored area known to robot i, and n is the number of robots (to be maximised). In other words, this metric measures the percentage of the full current exploration effort known to each robot, on average. The purpose of this metric is to examine how well members of the team share information: when a robot decides where to explore next, it is helpful to know what its teammates are doing and what other parts of the environment have already been explored. The more knowledge is shared between robots, the easier it is to efficiently coordinate the team effort and prevent repeated exploration of the same area.

- Responsiveness:

$$M_4 = \frac{1}{n}\sum_{i=1}^{n} c_i$$

where c_i is the time since the most recent message received from the BaseStation by robot i was originally sent (to be minimised). In other words, this metric measures the delay of commands from BaseStation to robots, averaged over all robots in the team. Human responders will want to have control over the robot team and it is not desirable to have robots out of the range of the command centre for a long time. Note that the last message from BaseStation need not be received directly – it may arrive over multiple hops or via a mobile relay.

- Completion:

$$M_5 = t_{Total}$$

  where t_{Total} is the time required for knowledge of the *full environment* to be attained by the BaseStation (to be minimised). This metric, while closely related to M_2, specifically addresses the goal of fully exploring the environment. (In several results presented in Chapters 5 and 6, the time taken to attain knowledge of 50% of the environment is also presented).

~

This completes the presentation of the Problem Statement. In the next Chapter Role-Based Exploration is introduced and described, and thereafter its performance is evaluated using the metrics presented here, both in simulation (Chapter 5) and in reality (Chapter 6).

Chapter 4

Role-Based Exploration

This chapter presents the main contribution of this thesis: *Role-Based Exploration*, a new exploration algorithm for teams of robots exploring unknown, communication-limited environments. A description of the basic approach is followed by sections detailing two significant improvements. The first involves an improved method for determining rendezvous locations. The second involves the introduction of role swaps within the team, leading to a dynamic team hierarchy that evolves as the exploration effort unfolds. Both improvements lead to gains in exploration speed, team connectivity and information sharing; the latter additionally results in a useful emergent behaviour that adapts well to variable communication availability and to any shape of environment.

4.1 The Basic Approach

4.1.1 Roles and Team Hierarchy

In basic role-based exploration, each robot in the team is assigned one of two possible roles:

1. *Explorer.* Explorers have the task of exploring the farthest reaches of the environment. To communicate their findings, they return periodically to previously agreed rendezvous points where they pass their knowledge to a Relay.

2. *Relay.* Relays act as mobile links between Explorers and the BaseStation, ferrying information back and forth. The primary purpose of a Relay is to communicate Explorers' findings up the communication chain, and to communicate control commands from the Basestation down the communication chain. If a Relay discovers information about the environment while relaying, this contributes to team knowledge, but exploration is only an incidental byproduct of the Relay's movement.

Roles are assigned prior to the start of exploration. In the basic approach, the roles do not change (at least not until a dynamic hierarchy is introduced in an advanced version of Role-Based Exploration in section 4.3). The team hierarchy may be represented by a tree, as demonstrated in Fig. 4.1.

4.1.2 Localisation, Mapping and Occupancy Grid

Each robot keeps track of its own pose (location in the x–y plane and yaw) and has a range sensor. Range measurements are synchronised with pose measurements to enable an updating of the map.

For experiments reported in this thesis, an occupancy-grid based map was used [41], although the approach could be tailored to topological or feature based maps

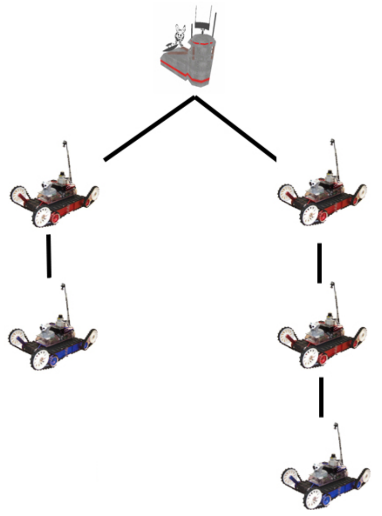


Figure 4.1: A possible hierarchy for Role-Based Exploration. The BaseStation (top) is the root of the hierarchy tree, Explorers (blue) are leaves, and there may be one or more Relays (red) in a branch.

if required. In the occupancy grid, each cell represents a space in the world (for example, 5cm×5cm). For purposes of navigation and planning, it is necessary to know for each cell whether it is *unknown*, *free*, or an *obstacle*.

Since data volume can pose difficulties for large maps, both in terms of communication and in terms of computation, a compact storage method for the occupancy grid was desirable. While two bits would suffice to store the cell's state (*unknown*/*free*/*obstacle*), there are a number of additional pieces of information associated with each cell, as detailed in later sections. Therefore, each cell of the grid was stored as a single byte, with each bit representing a single piece of information as detailed in Table 4.1. Individual bits can be toggled on and off as new information arrives. There are possibilities to compress information even further (for example, in the experiments on real robots, maps were compressed using the png image format – see section 6.3.3).

Given a range scan and a pose, the occupancy grid is updated as demonstrated in Fig. 4.2: First, the points of the scan (provided by the range sensing device as a distance measurement for each angle scanned) are converted to Cartesian coordinates

Bit	Information
0	free space
1	safe space
2	obstacle
3	boundary of frontier
4	belongs to skeleton
5	child rendezvous point
6	parent rendezvous point
7	path

Table 4.1: Information stored in one byte at each cell in the occupancy grid. Each bit is toggled on or off, indicating whether the related information is true or not.

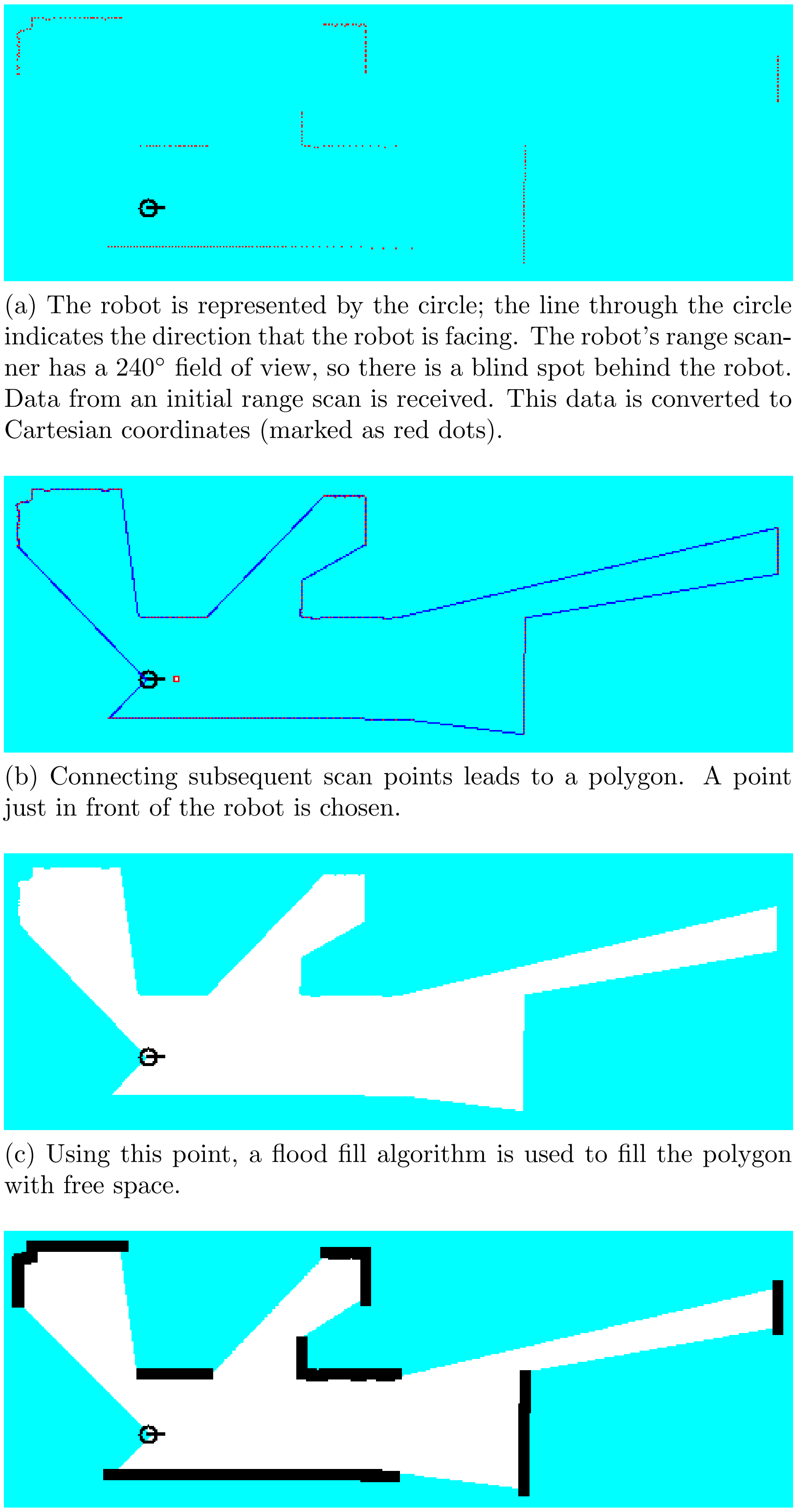
that match the coordinate system of the map. It is easy then to create a polygon of the points found by the scan.

Second, a flood-fill algorithm is used to fill this polygon in the map as *free*. An efficient way to do this is to choose a point known to be free (such as the cell immediately in front of the robot), and recursively designate as free all neighbours of this cell that are within the polygon but not already *obstacle* cells.

Third, where subsequent scan points are very close together, it is reasonable to assume that they are part of a connected obstacle, so the corresponding cells in the grid are labeled as *obstacle* and connected.

For the experiments conducted in simulation (Chapter 5), perfect localisation and scan data was assumed. This was a strong assumption since in reality odometry can be noisy and unreliable, and range scan data can contain spurious or noisy measurements. However, all exploration algorithms suffer from these problems equally and the purpose of the experiments was to compare Role-Based Exploration to existing techniques in a straightforward, simple, and repeatable manner. Thus this assumption was considered acceptable.

For the experiments conducted on a real system (Chapter 6), the problem of noisy data had to be addressed, and both scan matching and particle filter techniques were used for improved localisation.



(a) The robot is represented by the circle; the line through the circle indicates the direction that the robot is facing. The robot's range scanner has a 240° field of view, so there is a blind spot behind the robot. Data from an initial range scan is received. This data is converted to Cartesian coordinates (marked as red dots).

(b) Connecting subsequent scan points leads to a polygon. A point just in front of the robot is chosen.

(c) Using this point, a flood fill algorithm is used to fill the polygon with free space.

(d) Scan points sufficiently close together are connected as obstacles.

Figure 4.2: Turning range scan data into an occupancy grid based map

4.1.3 Frontier Polygons and their Utilities

In many existing approaches, autonomous exploration is achieved by the use of "frontiers" (following Yamauchi's work [156, 157]). Frontiers are the *boundaries* between known space and unknown space. For the experiments described in this thesis however a slightly different approach was applied that uses frontier *polygons.* This approach is based on a system that is used by the "Amsterdam Oxford Joint Rescue Forces", a joint team between the Universities of Amsterdam and Oxford [151, 149, 150] that competes annually in RoboCup's Virtual Robots competition [6].

Frontier polygons are determined as shown in Fig. 4.3. An essential tool is the concept of *safe space.* Safe space is the free space within a "safe radius" of the robot's location (in other words, open areas near the robot). The safe radius is typically about half the distance of the range sensor's maximum range. When the safe space is artificially superimposed on the free space, the free areas that remain form the "frontier polygons". Frontier polygons are easy to calculate in a map using standard image processing techniques – the method chosen in the system presented here uses contour tracing[1].

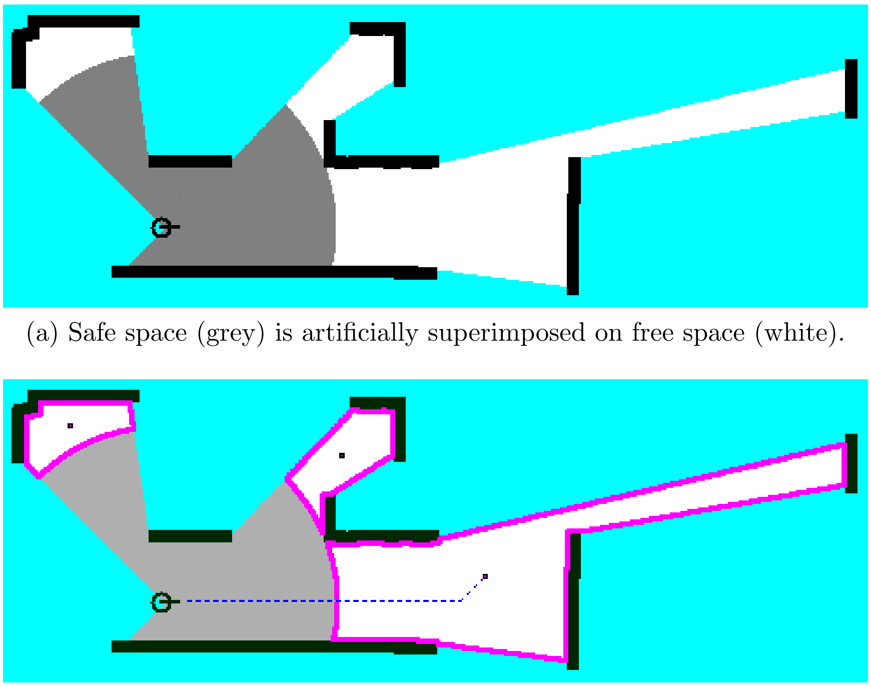
Defining frontiers in this manner means that some useful values can be calculated for each frontier polygon, for example its area (which we treat as the potential information gain) and the path cost of reaching it. For the latter, the *centre* of the frontier polygon is used as a goal for the path planner.

These values can be used to assign a specific *utility* to each frontier polygon p_i:

$$U(p_i) = A(p_i)/C^n(p_i)$$

where $A(p_i)$ is the area of frontier polygon p_i, $C(p_i)$ is the length of the path to that

[1] Some good examples of how to perform this are available at `http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/alg.html`



(a) Safe space (grey) is artificially superimposed on free space (white).

(b) Frontier polygons (outlined in magenta) are the areas of free space beyond safe space. For each frontier polygon it is possible to calculate its area and the path cost to its centre.

Figure 4.3: Using safe space to find frontier polygons

frontier polygon's centre (the path cost), and exponent n determines the exploration behaviour. High values of n lead to exploration of nearby frontier polygons (such as rooms) whereas low values mean that robots are more likely to pursue larger frontier polygons (such as hallways or open spaces) [152]. For most experiments reported later in this thesis, $n = 2$ was used, since in practice this provided a good balance between trying to explore open space quickly, while not changing direction too frequently.

4.1.4 Frontier Selection and Team Coordination

When there are multiple explorers active, there is a risk that two of them may choose to explore the same frontier polygon. This is likely to be an inefficient use of resources, and it makes more sense for teammates to coordinate in such a manner that they explore different areas. This is a common robotics problem; elsewhere a robot-to-frontier assignment algorithm has been proposed by Burgard *et al.* that coordinates

```
**Input**: Set $R$ of explorers within range; Set $F$ of frontier polygons
**Output**: List $L$ of $\{r_i, f_j\}$ robot to frontier polygon assignments for $\forall r_i \in R$
**Data**: $Q$ is a priority queue of all $\{r_i, f_j\}$ pairings, ordered by utility $U_{i,j}$
foreach $r_i \in R$ do
    foreach $f_j \in F$ do
        $U_{i,j} = Area(f_j)/StraightLineDistance(r_i, f_j)^n$
        $Q.add(\{r_i, f_j\})$
    end
end
while not $Q.isEmpty()$ do
    $\{r_a, f_b\} = Q.pop()$
    $U_{a,b} = Area(f_b)/PathCost(r_a, f_b)^n$
    if $U_{a,b} > U(Q.peek())$ then
        $L.add(\{r_a, f_b\})$
        foreach $\{r_i, f_j\} \in Q$ where $i == a$ or $j == b$ do
            $Q.remove(\{r_i, f_j\})$
        end
    else
        $Q.add(\{r_a, f_b\}$
    end
end
```

Algorithm 1: The robot to frontier polygon assignment algorithm, as first proposed in [148]. Typically the bottleneck in robot-frontier assignment calculations is the time it takes to calculate paths from robots to frontier polygons (especially in large environments). Using straight line distances as an initial estimate instead of full path costs means that fewer accurate paths need to be calculated.

the team centrally [18]; in a later related work by Wurm *et al.*, the robot-to-frontier assignment problem has been shown to have an optimal solution, using the Hungarian method [155, 77].

Calculating the utility of a frontier polygon for a particular robot involves calculating the cost of the path from that robot to that frontier polygon's centre. However, paths can be expensive to compute in large or complex environments. To speed up the process of assigning robots to frontier polygons, an approximation is therefore initially applied: instead of computing the full path, the straight-line distance is used (which ignores obstacles). This method was originally proposed by Visser and Slamet in [148] and has been shown to be quick and effective at computing the robot to frontier polygon assignment.

The full algorithm is presented in Alg. 1, and can be summarised as follows: each explorer calculates, for each robot-frontier pair $\{r_i, f_j\}$, the utility $U_{i,j}$ (where the set of robots is all explorers within range including itself) using as a path cost the *straight line distance* from r_i to f_j. The pairings are then inserted into a priority queue in order of their utility, with the pairing having highest utility at the front. A pairing is retrieved from the front, and its true utility (using the *path cost*, instead of straight line distance) is calculated. If it remains the pairing having the highest utility, then the robot from that pairing is assigned to the frontier polygon from that pairing, and all remaining pairings involving that robot and that frontier polygon are discarded. If it does not, that pairing is reinserted in the queue at the correct position using the new exact utility value. Iteratively, all robots are assigned to frontier polygons in this manner.

While this algorithm may not be optimal in rare cases, it nevertheless performs well, is easy to implement, and can be calculated quickly. The straight-line estimation step in particular means that far fewer robot-frontier polygon paths need to be calculated.

It must be emphasised that each robot determines this assignment on its own, in real-time, using its onboard computing capacity; there is no necessity for a central coordination mechanism. Since explorers within range of one another share the same map (and will therefore compute the same explorer-frontier pairing), no two robots will be assigned to the same frontier polygon. An exception occurs when there are fewer frontier polygons than robots; in this case more than one robot may pursue the same frontier polygon. In practice (both in simulation and in reality), such cases are rare, as new frontiers tend to open quickly in most environment types.

4.1.5 Communication and Map Sharing

Almost all robot teams today use some form of wireless communication. Depending on the hardware and medium used, bandwidth can be an issue and there has been much work on compressing or compactly communicating full or partial maps [90, 112]. In the implementation used here, with an occupancy grid containing a single byte for each cell (see section 4.1.2), data requirements are not huge. A grid of 800 × 600 cells, for example, requires 480KB. For robots using standard 802.11 wireless to communicate, this is a manageable load, and in the implementation on a team of real robots (see chapter 6) sharing maps of this size was not a problem and occurred at a rate of multiple times per second, when desired. Possibilities for further compression certainly exist: in a later implementation on real robots that used only network cards on the robots themselves to communicate (and no central router), PNG image compression reduced the same maps to 4KB or less (this is described in greater detail in section 6.3.3).

As a result, it was assumed for the purposes of this thesis that robots, when in range, can fully communicate both their own position and their own map to one another. Knowing teammates' positions means that robots can avoid one another and erase erroneous obstacle measurements in their maps that are actually their teammates. Knowing teammates' maps means that robots can increase their knowledge of the environment and coordinate their exploration so that areas already explored are not visited again.

The use of ad-hoc wireless networks today means that there does not need to be a central network infrastructure. Robots can detect and recognise one another once in range, and connect. In simulation it was therefore assumed that robots could fully share their information whenever they were in range according to the communication model used; on the system of real robots this was successfully implemented, verifying the assumption.

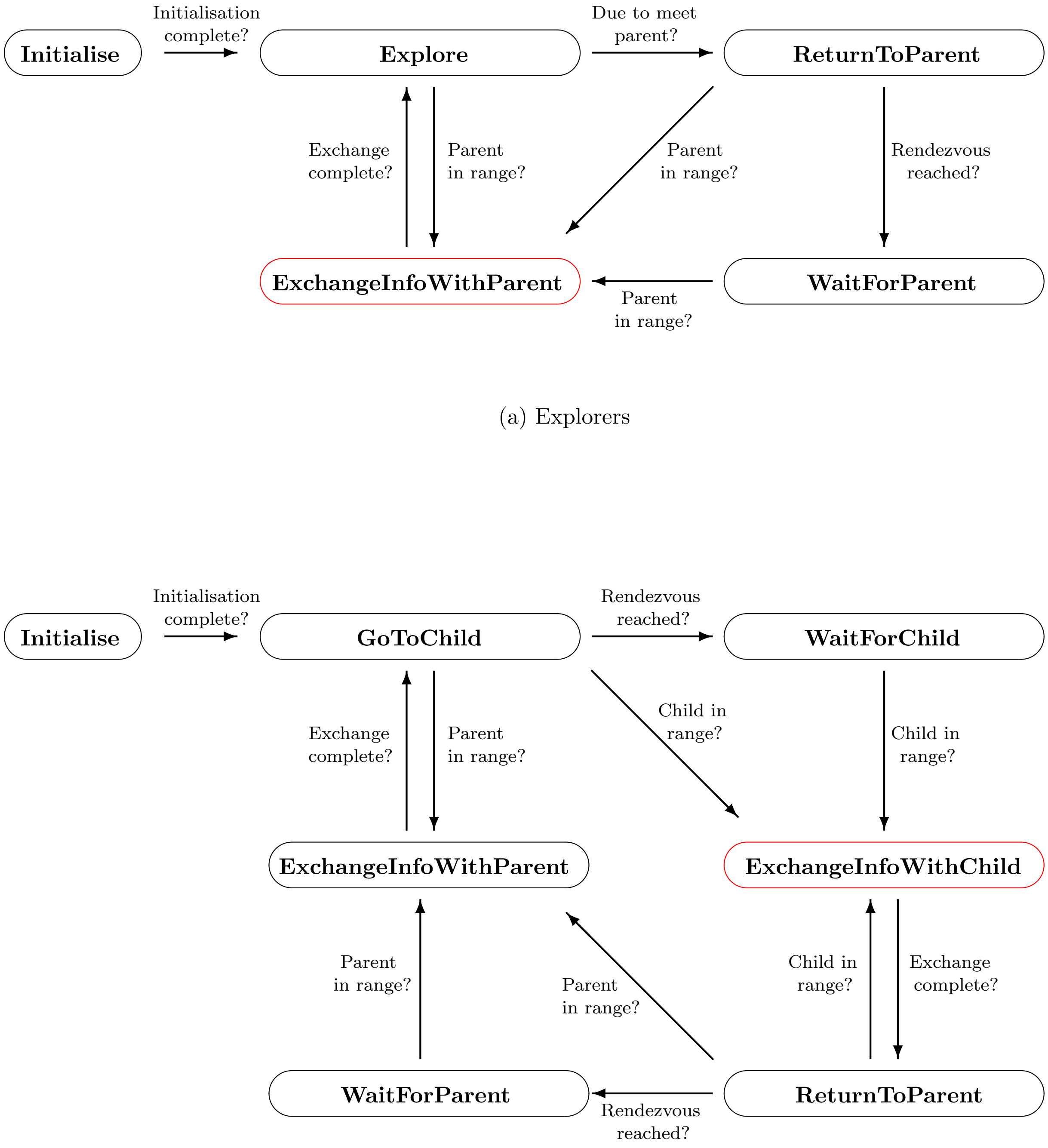
4.1.6 State Transitions and Rendezvous

A Relay must: meet an Explorer; exchange information with this Explorer; return to the Basestation; exchange information with the Basestation; and go to meet the Explorer again. An Explorer must: explore for a time; return to meet its parent Relay; exchange information with this Relay; and then return to exploring. These state transition for Explorers and Relays are presented in Fig. 4.4. (Naturally there are potential points of failure in this approach, *e.g.* if a Relay fails, its child Explorer may end up stationary, waiting for it. These are discussed further in section 7.4.1.)

The periodic meeting for information exchange ("rendezvous") is clearly a crucial element of the approach. It is essential that Relay and Explorer agree on a specific location for rendezvous, so that they can find one another. The rendezvous location is thus a specific place in the map, chosen by the Explorer and communicated to the Relay.

Relay and Explorer share the same map when in range. Therefore, the Explorer can predict the Relay's path, and determine how long it will take the Relay to return to the BaseStation, turn around, and make its way to the next rendezvous point. At any point in time the Explorer can check whether the Relay is soon to reach rendezvous, and whether the Explorer itself should stop exploring and make its way to rendezvous as well. In other words, the Explorer can determine the ideal moment of transition from state *Explore* to state *ReturnToParent*. In this manner the rendezvous is carefully timed, with both the Relay and the Explorer approaching it at approximately the same time. This is important for purposes of efficiency: well-timed rendezvous means that no robot wastes time waiting for a teammate to appear.

In the basic version of Role-Based Exploration, the most forward point of progress is chosen as the subsequent rendezvous. In other words, when an Explorer makes its way back to a rendezvous, it already knows where it will tell its parent Relay to meet it the next time. Choosing locations for rendezvous in this manner means that



(a) Explorers

(b) Relays

Figure 4.4: State transition diagrams for Explorers and Relays. The states outlined in red are synchronous, during rendezvous.

subsequent meetings between the robots are set deeper and deeper in the environment, pushing the exploration effort further and further into new areas.

Note that chance encounters are taken into account. If an Explorer happens to meet its parent Relay by chance prior to their scheduled rendezvous (this can happen for example in environments containing loops), then the chance encounter is treated like a planned rendezvous; both robots exchange information, and replan.

There is an inherent trade-off involved in the rendezvous process. If the chosen rendezvous location is deep in the environment, the Relay must travel farther to reach it and updates at the BaseStation are received less frequently. If it is chosen closer to the BaseStation then the Explorer spends much time returning to rendezvous and less time discovering new information.

4.1.7 Preventing Redundant Relaying

When two Relays are in state *ReturnToParent* and they are both taking a similar route back to BaseStation at the same time, they unnecessarily duplicate one another's work. It would be more efficient to share information between the two, and have only a single Relay make the trek back to BaseStation.

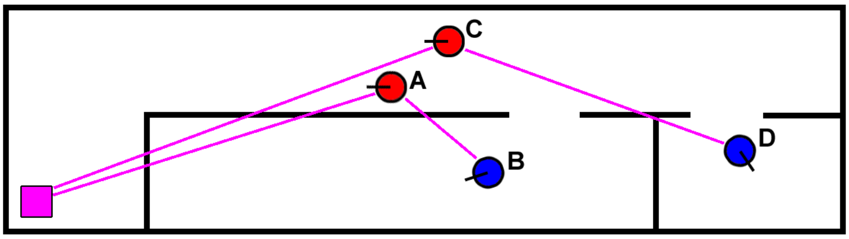


Figure 4.5: An example of redundant relaying

An example is presented in Fig. 4.5. *A* is a relay for explorer *B*; *C* is a relay for explorer *D*. Both Relays have recently rendevoused with their child and are now on their way back to the BaseStation. It is unnecessary for them both to make the trip along the long hallway; it would make more sense for them to share information, for

A to travel back on its own, and for C to return to the next rendezvous with its child.

This redundant relaying is straightforward to prevent by introducing the "Redundancy Rule": see Table 4.2.

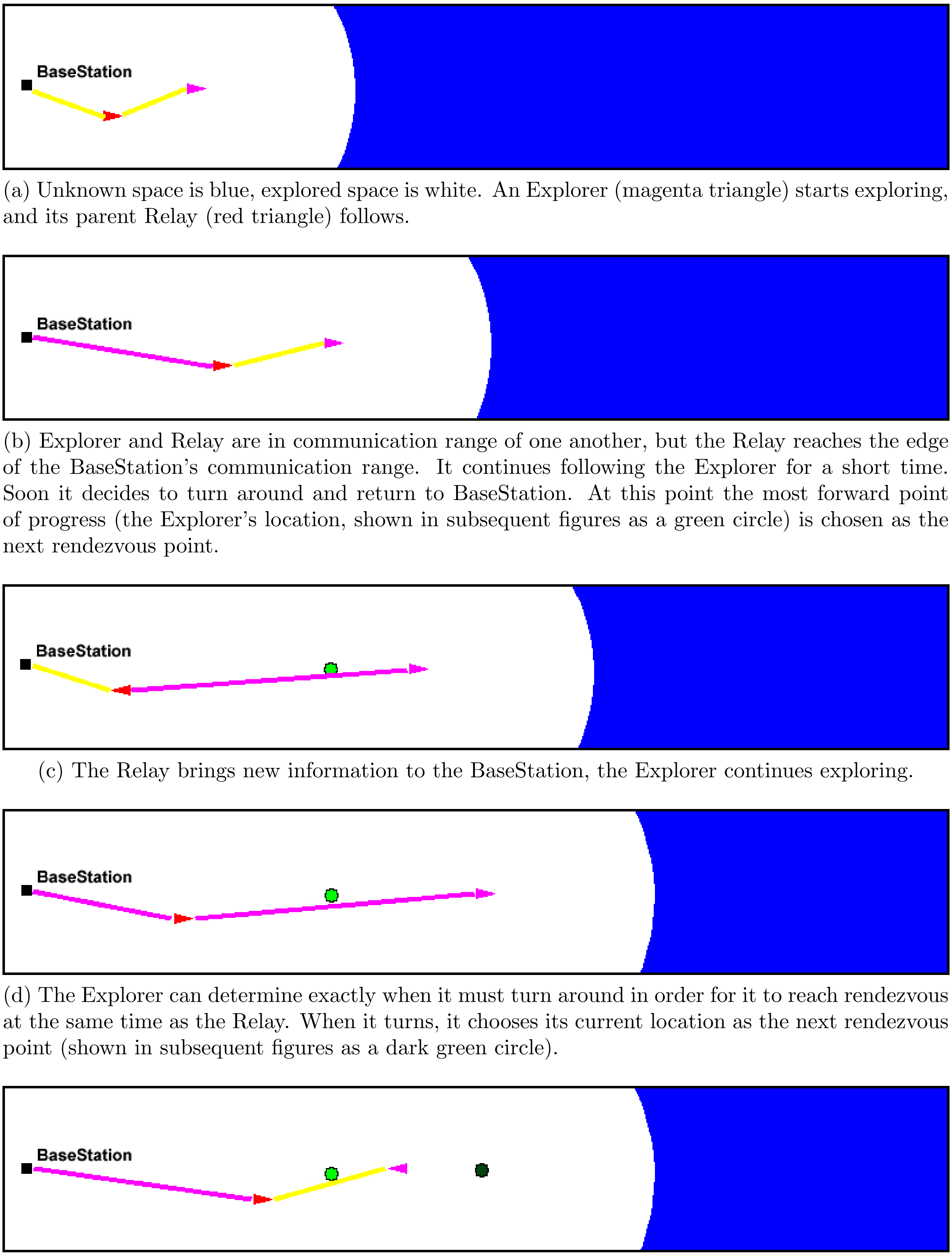
> Consider two Relays, R_A and R_B that have encountered one another and established a communication link. If the following statements are all true:
>
> - R_A is in state *ReturnToParent* and R_B is in state *ReturnToParent*
> - R_A and R_B have the same parent (usually the BaseStation)
> - R_A is closer to the parent than R_B (without loss of generality)
>
> then let R_A shoulder the burden for both Relays in its return to the parent, and let R_B change state to *GoToChild*.

Table 4.2: The Redundancy Rule

This rule is only applied to pairs of Relays in different branches of the tree, and does not affect Explorers. The rule is a very minor, simple adjustment to basic Role-Based exploration, and ensures that two Relays don't waste time moving down hallways side by side.

~

This completes the description of Role-Based Exploration in its most basic form; an example showing a full cycle of the states each type of role undergoes is presented in Fig. 4.6. Role-Based Exploration has been compared extensively to several existing algorithms [34]; simulation results are presented in Chapter 5 and results of implementation on a real system are detailed in Chapter 6. Over the course of the research, however, several key improvements were made to the basic algorithm; these are detailed in the following sections.



(a) Unknown space is blue, explored space is white. An Explorer (magenta triangle) starts exploring, and its parent Relay (red triangle) follows.

(b) Explorer and Relay are in communication range of one another, but the Relay reaches the edge of the BaseStation's communication range. It continues following the Explorer for a short time. Soon it decides to turn around and return to BaseStation. At this point the most forward point of progress (the Explorer's location, shown in subsequent figures as a green circle) is chosen as the next rendezvous point.

(c) The Relay brings new information to the BaseStation, the Explorer continues exploring.

(d) The Explorer can determine exactly when it must turn around in order for it to reach rendezvous at the same time as the Relay. When it turns, it chooses its current location as the next rendezvous point (shown in subsequent figures as a dark green circle).

(e) Relay and Explorer meet at the first rendezvous point. The Explorer communicates its updated map to the Relay, and tells it where the next rendezvous (dark green circle) will be.

Figure 4.6: A demonstration of basic Role-Based Exploration

4.2 Improving Rendezvous

4.2.1 Common Problems

While the rendezvous point selection outlined in the previous section was successful at pushing the exploration effort deeper into the environment, it was not always optimal. There are certain situations where it leads to inefficient behaviour, for example:

1. **Poor environmental choices for rendezvous**

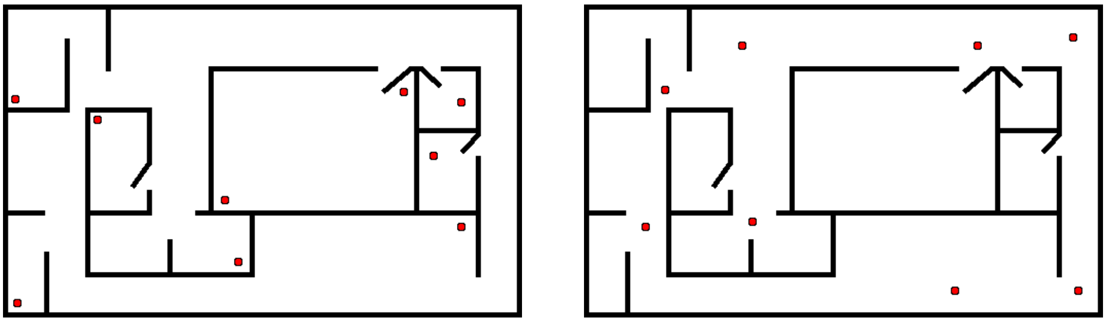
   If at the moment the Explorer decides to return to rendezvous it is at a poor location, then the subsequent rendezvous will have to occur at that poor location. Poor locations are points in a dead end, behind an obstacle, or generally close to an obstacle. Some examples are presented in Fig. 4.7a.

   Exploration proceeds fastest when Explorer and Relay meet as quickly as possible; to meet as quickly as possible they need to enter one another's communication range as quickly as possible; for them to enter one another's communication range quickly, it is desirable to arrange rendezvous in open space, or at junctions. Some desirable rendezvous locations are presented in Fig. 4.7b.

2. **Poor planning choices for rendezvous**

   If the Explorer has chosen the next rendezvous location in advance, and then changes its mind regarding where to explore next, there is a chance that it will have to travel a long distance to the next rendezvous. This means less time for exploration and a loss of efficiency. It makes much more sense to choose for next rendezvous a location that is easy to reach, thereby maximising exploration time.

In short, it is desirable to choose rendezvous points in open space or at junctions, and near the Explorer's next intended exploration area.



(a) Poor choices

(b) Good choices

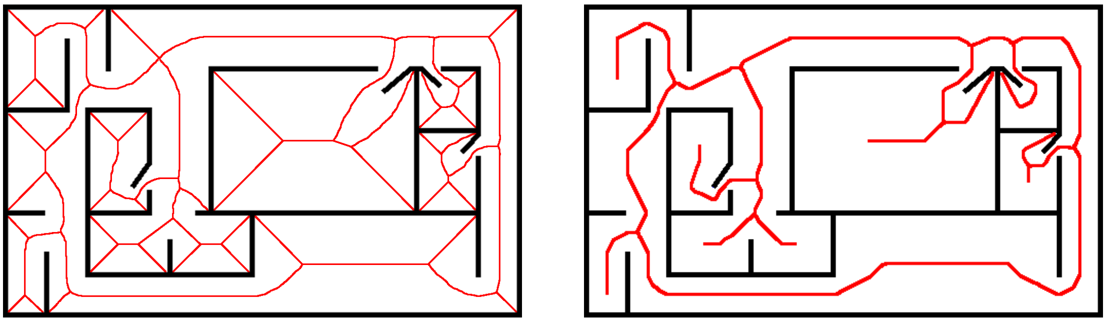
Figure 4.7: Comparing choices for rendezvous: some locations are much more suitable than others

4.2.2 Skeletonisation

How can we find points that are in open space or at a junction? Two classes of candidate methods exist for this task:

- **Voronoi graphs**. In Voronoi graphs, lines are constructed between a set of points S in such a manner that each line is equidistant to its two nearest points. Voronoi graphs have been used for topological map building, navigation, and place detection [28, 29, 11]. When applied to a map, points in the walls are added to the set S in order for standard Voronoi graph calculation algorithms to work.

- **Medial axis transform (also known as "thinning")**. A medial axis transform takes as input a binary image, and outputs a transformed image with distance measurements to the closest obstacle at each cell. This is performed using a distance transform with multiple passes over the image. A wide range of techniques have been proposed to achieve this since the 1960's, each having various advantages or disadvantages; a review is presented in [80]. When only the local maxima of the medial axis transform are considered, a skeleton emerges. A common analogy for how thinning is performed is that of a prairie fire: if one were to set fire to all edges of a shape simultaneously and extract



(a) A Voronoi graph of an environment

(b) Skeletonisation using thinning applied to the same environment

Figure 4.8: Comparing Voronoi graphs with thinning

the locations where fires from two sides met, this would be the equivalent of the shape's skeleton.

For the purposes of experiments in this thesis, thinning algorithms were chosen as slightly preferable to Voronoi graphs, since they provide a cleaner skeleton and are easier and faster to compute (see Fig. 4.8 for an example of each). Specifically, Hilditch's algorithm was chosen for an initial implementation [63]. It proved to be easy and fast to compute, and effective for finding a connected skeleton[2].

4.2.3 Choosing a Rendezvous Point

Every point in the skeleton as calculated above is certain to be in open space, and the skeleton contains every junction. There are many points in the skeleton however, so it is necessary to reduce it further to a small set S of rendezvous candidates. This is performed in three steps:

1. **Finding junctions**. Junctions are the most suitable rendezvous locations of all, since they provide line of sight (also in terms of communication) in multiple directions. Hilditch's algorithm requires the calculation of a *neighbour traversal*

[2] A useful tutorial on how to implement Hilditch's algorithm can be found at `http://cgm.cs.mcgill.ca/~godfried/teaching/projects97/azar/skeleton.html`.

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Table 4.3: To check whether a cell p_i in the occupancy grid is a junction, it is necessary to examine the cells immediately surrounding it. The traversal function $T(p_1)$ is the number of 0,1 patterns in the sequence p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_2. When $T(p_i) \geq 3$, p_i is a junction.

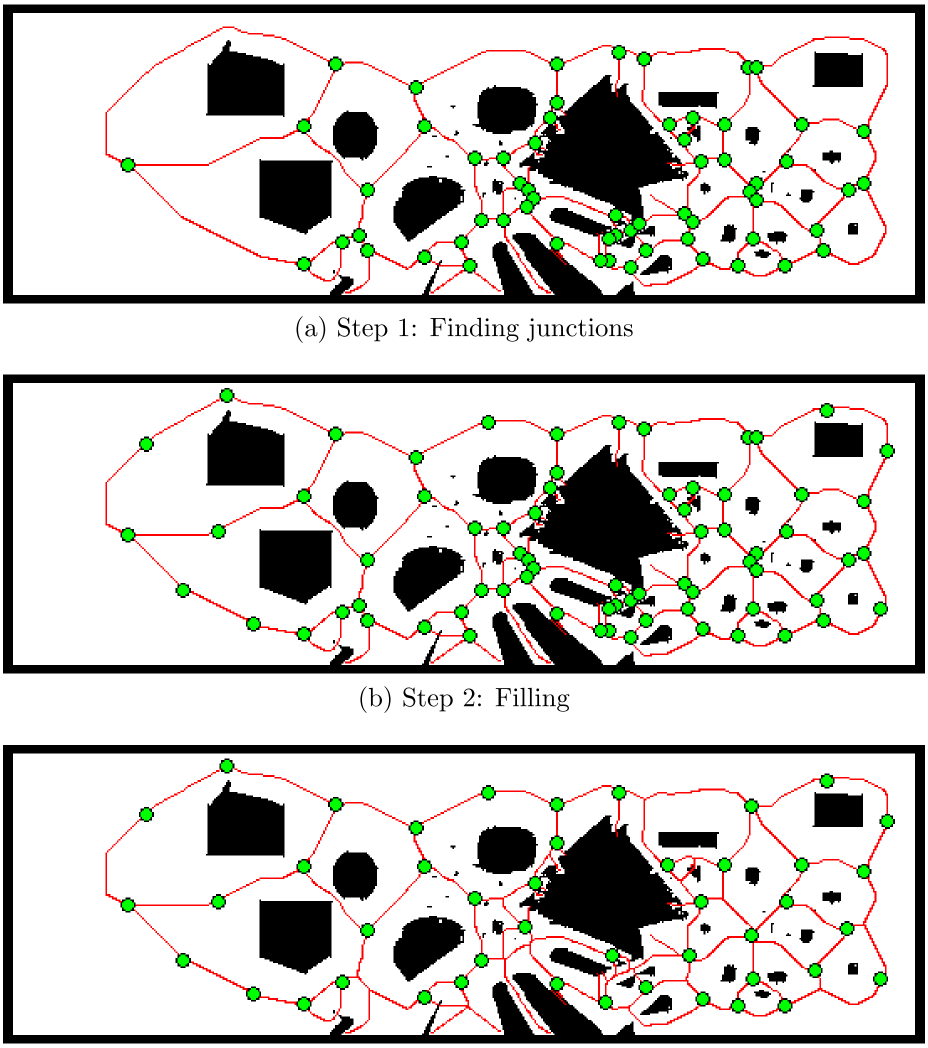
function $T(p_1)$, described in Table 4.3. This function can also be used to find junction points in the skeleton: any point p_1 that is a junction in the skeleton will have $T(p_1) \geq 3$. All junctions are added to the set S.

2. **Filling**. A skeleton may contain long stretches without junction points, for example along a hallway, so it is necessary to fill some extra points in. This is performed via iteration over all points in the skeleton; those points that are a minimum distance from all existing rendezvous points are added to the set S.

3. **Pruning**. On the other hand, complex parts of the environment may contain a large number of junction points in a small area – to simplify calculations, only one point per given density is added. This is performed by iterating over the current set of potential rendezvous locations; those points too close to another point in S are removed.

These steps for choosing a set of candidate rendezvous points are demonstrated in Fig. 4.9. What remains is a small set of possible rendezvous points, distributed fairly evenly over the known environment and including all junctions. Since an Explorer, after the current rendezvous, will move towards its preferred frontier, it makes most sense to plan the next rendezvous in the vicinity of that frontier (as discussed also in section 4.2.1). Thus the exact location for the next rendezvous is that point within S that is *closest* to the centre of the Explorer's next choice of frontier.

The full algorithm for rendezvous point calculation is presented in Alg. 2, and several further examples of skeletonisation and rendezvous point candidates are pre-



(a) Step 1: Finding junctions

(b) Step 2: Filling

(c) Step 3: Pruning

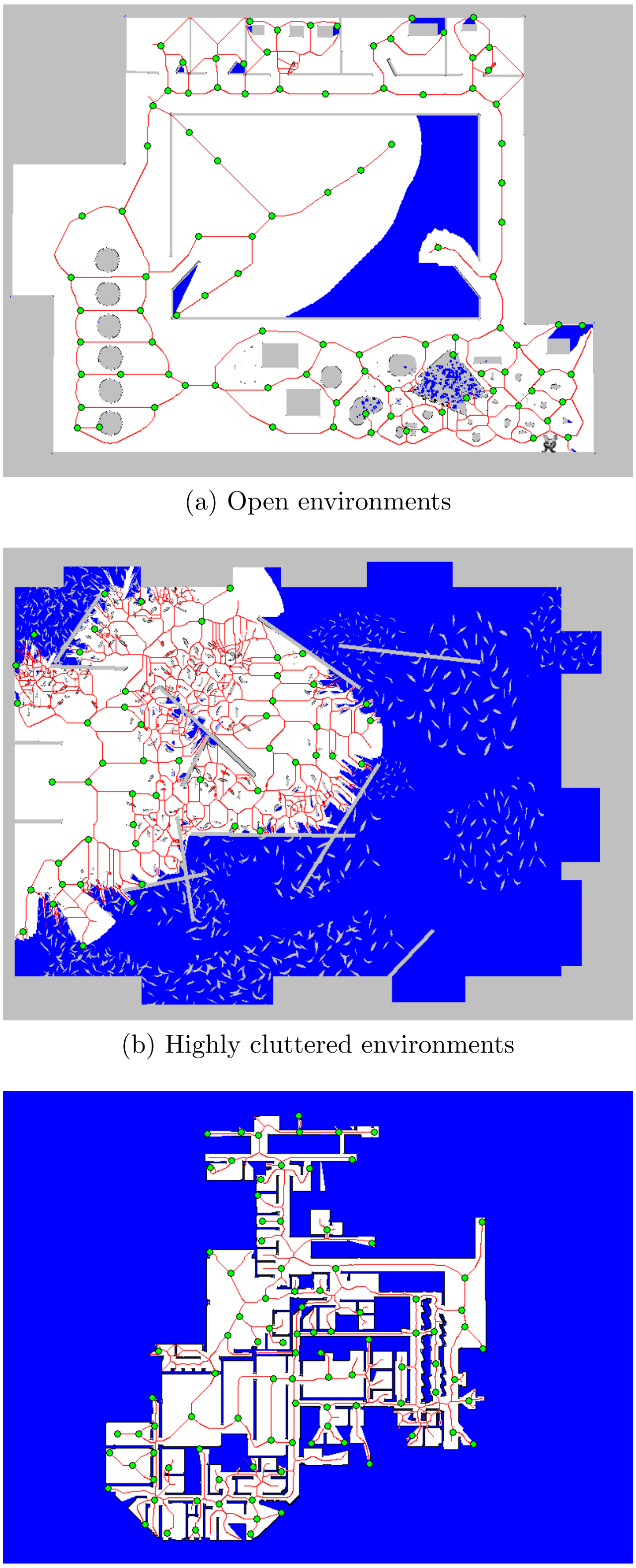
Figure 4.9: Finding a set of rendezvous candidates

sented in Fig. 4.10.

Using such a method for determining locations for rendezvous led to significant improvements in the efficiency of exploration [36]; the results were so compelling that only this new method for rendezvous point calculation was used in subsequent implementations.

```
**Input**: The Explorer's map $M$; the Explorer's next frontier $F$
**Output**: The next rendezvous point $r_{next}$
**Data**: List of points $S$ = hilditchThinning($M$)
**Data**: List of points $R$ (the list of candidate rendezvous points)
// Add junction points
foreach $s_i \in S$ do
    if $neighbourTraversal(s_i) \geq 3$ then
        $R.add(s_i)$
    end
end
// Fill in extra points where distances are too far
foreach $s_i \in S$ do
    boolean $addToList = true$
    foreach $r_j \in R$ do
        if $distance(s_i, r_j) <$ $threshold$ $T_1$ then
            $addToList = false$
            $break$
        end
    end
    if $addToList$ then
        $R.add(s_i)$
    end
end
// Prune points where distances are too close
foreach $r_i \in R$ do
    foreach $r_j \in R, i \neq j$ do
        if $distance(r_i, r_j) <$ $threshold$ $T_2$ then
            $R.remove(r_i)$
            $break$
        end
    end
end
// Choose the point closest to the Explorer's next frontier
$r_{next} = R.pop()$
$d_{min} = distance(r_{next}, F)$
while not $R.isEmpty()$ do
    $r_{curr} = R.pop()$
    if $distance(r_{curr}, F) < d_{min}$ then
        $r_{next} = r_{curr}$
        $d_{min} = distance(r_{curr}, F)$
    end
end
$return(r_{next})$
```

Algorithm 2: Choosing a rendezvous point.



(a) Open environments

(b) Highly cluttered environments

(c) Office-like environments

Figure 4.10: Some further examples of rendezvous point calculation

4.3 Introducing a Dynamic Hierarchy

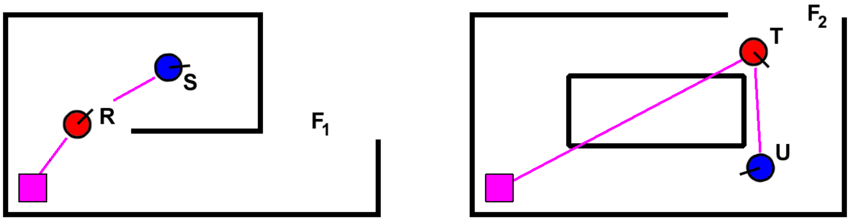
4.3.1 Common Problems

Even with an improved rendezvous point calculation leading to more efficient exploration, several instances were observed where the behaviour of the team was clearly not as efficient as it could be:

1. **Dead Ends**. When an Explorer finishes exploring a room (or any dead end), and its parent Relay is positioned at or near the entry, this Relay may remain stationary while the Explorer retraces its path and starts making its way to a new frontier. It would be more efficient for the two to switch roles; for the Relay to become an Explorer (since it is closer to the next frontier) and for the Explorer to now become the Relay.

   An example is presented in Fig. 4.11a: Explorer S explores the room, but reaches a dead end. The only open frontier is now at F_1. Since relay R is closer to F_1, it is of advantage for the two robots to swap roles.

2. **Loops**. When an Explorer completes a loop, it can happen that now the Relay is closer to the next frontier than the Explorer. Rather than stop and wait for the Explorer to turn around and pass it, it would make more sense for the Relay to become an Explorer and for the Explorer to become its Relay.



(a) Dead ends

(b) Loops

Figure 4.11: Example scenarios where team behaviour is suboptimal. The magenta square is the base station, magenta lines indicate team hierarchy.

An example is presented in Fig. 4.11b: Explorer U has traveled clockwise around the obstacle in the middle of the environment while relay T has followed. Once the loop is closed, the only remaining frontier is at F_2. T is now closer to F_2 than U, so it makes sense for the two to swap roles. T becomes the explorer while U becomes the relay.

Multiple other situations arise that are variations on these examples.

4.3.2 Swapping Roles

To solve the problem of inefficient motion in dead-ends and loops, robots need to swap roles and exchange places within the team hierarchy. Several different attempts were made to solve this problem, and the swapping of roles is discussed in greater detail in section 7.1. Here, only the eventually chosen solution is presented, namely the "Role Swap Rule" (see Table 4.4).

Consider two robots A and B, each having destinations D_A and D_B, respectively. Let $\gamma(u, v)$ represent the path cost from location u to location v in a given map. When u and v are known, this value is easy to calculate using standard path planners (such as A*) on the map. Suppose A and B have encountered one another and established a communication link. If

$$max\{\gamma(A, D_A), \gamma(B, D_B)\} > max\{\gamma(A, D_B), \gamma(B, D_A)\}$$

then let A assume B's role, state, and location in the tree, and let B assume A's role, state, and location in the tree.

Table 4.4: The Role Swap Rule

This rule is applied equally to relays and explorers, both within the same branch and across branches.

In short, the role swap means that two robots analyse their current goals and check whether it would be faster if they switched roles. Therefore, the longest path among the four paths computed is always eliminated. If a role swap occurs within a single branch, this means that new information will travel faster from Explorers to the BaseStation. If it occurs across two separate branches, the role swap generally has a load balancing effect (as described in greater detail in section 7.1).

Perhaps the best way to demonstrate this dynamic behaviour, and how it improves the exploration effort, is by example: see Fig. 4.12 and the accompanying explanation in Table 4.5. In this simple demonstration, three separate applications of the role swap are applied: the swap in Stage III involves a relay and an explorer in the same branch, the swap in Stage IV involves a relay and an explorer from separate branches, and the swap in Stage V involves two relays in separate branches. Additional possible applications of the rule exist, and the rule is applied in the same way to larger hierarchies and longer branches.

4.3.3 Implementation in Practice

Implementing the role swap rule in practice is fairly straightforward, as long as several key implementation issues are kept in mind. This section explains why each robot needs to maintain two separate identification numbers, lists the information that must be swapped, describes the actual role swap protocol, discusses how to deal with oscillation, and lists all relevant parameters.

Two types of ID: static and dynamic

To allow role swaps while still maintaining a consistent hierarchy, each robot must maintain both a *static* ID and a *dynamic* ID.

The static ID is assigned to each robot at the start of exploration, and never changes, even if the robot swaps roles with a teammate. All robots maintain information about their teammates, such as location and whether they are in range. This information is related to the physical robot itself, and does not have any relation to the robot's position in the hierarchy. Thus it must be stored using the robot's static ID.

Stage I: Four robots set out to explore an unknown environment. Initially, A and C are relays, B and D are explorers. Following initial range scans, two frontiers are discovered (F_1 and F_2). By joint utility maximisation, B chooses to explore F_1 and D chooses to explore F_2.

Stage II: B explores F_1 and A follows B. The frontier F_1 opens up into a bigger frontier at F_3. In the meantime D explores F_2 and C follows D. D reaches the end of the room, and decides to rendezvous with C to relay new information back to base station. The members of the team are still fully connected, although the connection to base station is lost.

Stage III: B and A reach the limits of the team communication chain, and break from it – B continues to explore, A continues to follow. Two frontiers open up (F_4 and F_5), and B chooses to explore F_4. In the meantime, D and C have rendezvoused, and C has new information to relay back to base station. However, D's only frontier of interest is at F_3 (D is not aware of A and B's latest exploration knowledge since these have been out of range). Since

$$max\{\gamma(D, F_3), \gamma(C, Base)\} > max\{\gamma(D, Base), \gamma(C, F_3)\}$$

the role swap rule is applied, and C and D trade positions in the tree. C is now an explorer with F_3 as its goal, and D is now its parent relay with the base station as its goal. C and D agree on R_1 as the next rendezvous point.

Stage IV: Enough new information has been gained by B, so A and B rendezvous and A turns around to relay new information to the base station while B continues to explore. However, A encounters C, on its way to explore F_3. Since

$$max\{\gamma(C, F_3), \gamma(A, Base)\} > max\{\gamma(C, Base), \gamma(A, F_3)\}$$

the role swap rule is applied, and C and A trade positions in the tree. C becomes a relay for B, while A becomes an explorer with D as parent relay.

Stage V: Now an explorer, A chooses the nearest frontier at F_6 and starts to explore. D, having relayed information to the base station, is on the way to rendezvous with its child (now A) at R_1. C is on its way to the base station to relay new information. Since

$$max\{\gamma(D, R_1), \gamma(C, Base)\} > max\{\gamma(D, Base), \gamma(C, R_1)\}$$

the role swap rule is applied, and D and C trade positions in the tree. D becomes a relay for B, and C becomes a relay for A. D turns around to complete the task of relaying information, while C turns around to rendezvous with its child A at R_1.

Stage VI: In the meantime, A and B have continued exploration of open frontiers and fully explored the environment. Eventually all robots return to the base station.

Table 4.5: A description of the events in Fig. 4.12

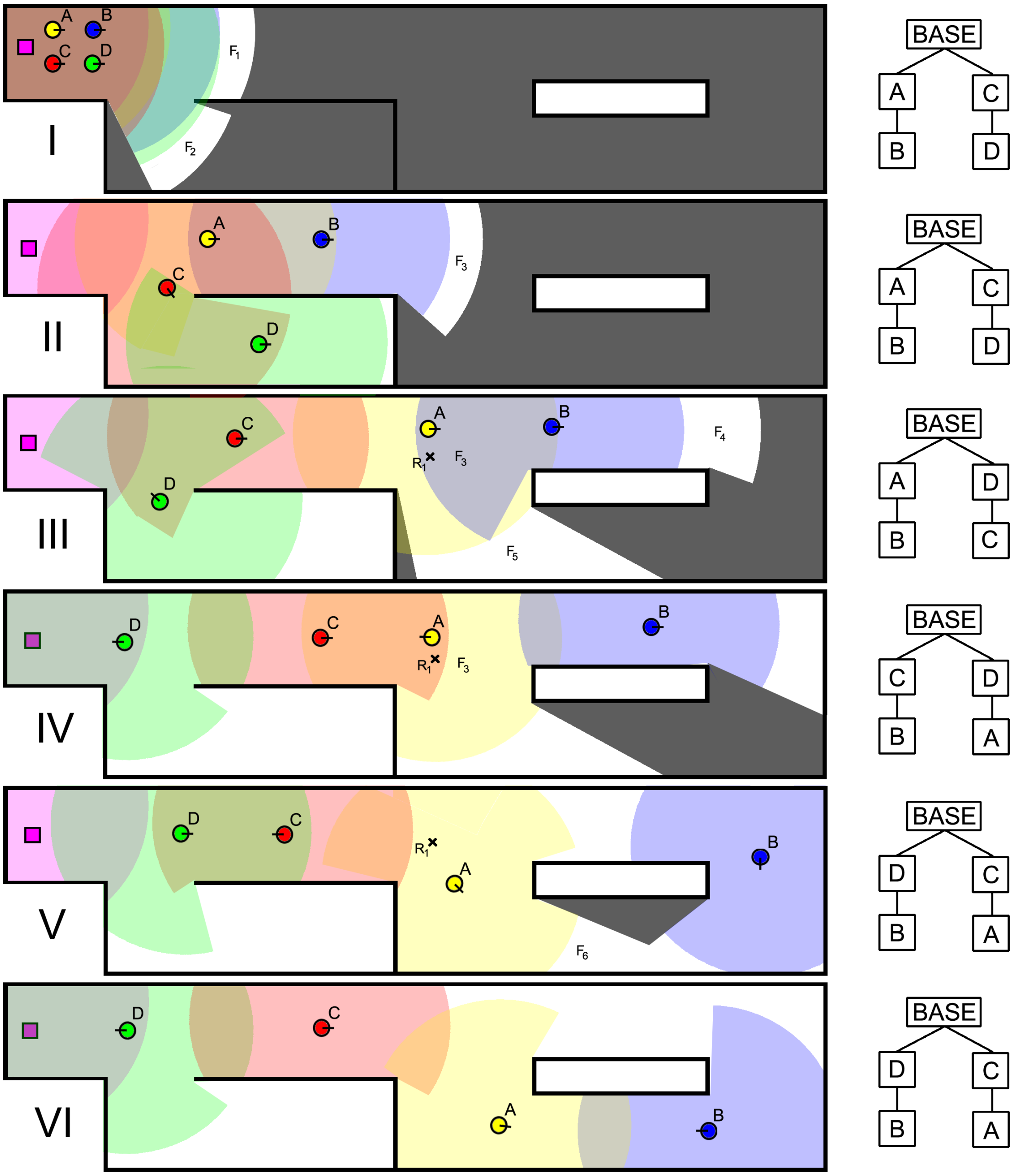


Figure 4.12: A demonstration of how a team hierarchy may change during an exploration effort. The team hierarchy is presented to the right of each stage of exploration. Dark parts of the map are unexplored, white parts have been sensed using range finders. The base station is the purple square on the left. Coloured circles indicate agents' respective communication ranges. A full explanation of the stages in this figure is provided in Table 4.5.

The dynamic ID specifies the exact location of a robot within the team hierarchy. Dynamic IDs are assigned at the start of the exploration effort, but change every time a robot swaps roles with a teammate. A robot looking for its parent or child will thus be looking for a robot having a given *dynamic ID*, since its parent or child may have switched roles unbeknownst to it.

Consider for example the case where robot A has to rendezvous with its parent B. However, unknown to A, B has swapped roles with D. D now has B's dynamic ID number. Since A is looking for the dynamic ID only, and not a specific robot, A finds D as its parent, and role-based exploration may proceed as expected.

Role swap message

The actual role swap itself involves the exchange of a *role swap message* that contains all relevant information necessary for a role exchange. This message contains the following information:

- dynamic ID (so that the robot knows its location in the hierarchy and can be found by teammates);
- role and current state;
- child's ID and child rendezvous (so that it can find its new child);
- parent's ID and parent rendezvous (so that it can find its new parent); and
- current goal (so this does not need to be recalculated)

No other information needs to be exchanged; the above is sufficient for exploration to proceed as normal after role swap.

Role swap protocol

It is very important that two teammates swapping roles are on the same page; if one robot decides to swap its role but the other does not, clearly confusion would ensue. In practice this is achieved by one robot *commanding* the other to swap roles. In the experiments presented in Chapters 5 and 6 this was achieved by having the robot with the *lower* static ID give the command to the robot with the *higher* static ID; it could be done the other way around as well, as long as one of the two teammates is authorised to make the decision.

The full role swap process can be described as follows:

1. Robots A and B enter one another's communication range

2. Robots A and B receive status and map updates from one another. The status update includes for each robot its current position and its current goal.

3. Without loss of generality, assume that robot A has a lower static ID than robot B (static IDs are unique). Robot A calculates the four path costs required by the role swap rule, and determines whether a role swap would be advantageous.

4. If yes, robot A sends a role swap message (see above) to robot B. If it receives a role swap message as a reply, the role swap occurs.

Avoiding oscillation

In rare cases, the role swap rule can lead to oscillatory behaviour by two teammates. For example, consider the case of two robots deciding on whether to exchange roles while moving past one another. It is possible that at the moment they start evaluating the path costs involved, it is advantageous to exchange roles. However, if the calculation takes time or if they are moving fast (or both), it is possible that once the decision is made, the role swap is no longer advantageous. As a result, the teammates may move past one another, decide to swap roles, turn around, move past one another, decide to swap roles again, and so on.

There are two straightforward ways to prevent such oscillations. In the first, oscillations can be circumvented by introducing a timeout on the rate of role swaps a robot may undergo. As long as the timeout is *at least as long* as the replanning interval used by the robots, oscillation is prevented. Consider robots A and B, who have decided to swap roles while passing one another. If the rare case occurs where the role swap turns out to be disadvantageous, they will both turn around and pass one another again. However, since there is a timeout on role-swapping that is longer than the timeout on replanning, they will replan *before* they consider swapping roles again. Once they replan, they each have new goals and new path costs, and will realise that they should not swap roles again, preventing oscillation.

A second, perhaps tidier, way to prevent oscillation is to introduce hysteresis into the role swap mechanism. This can be achieved with a simple adjustment to the Role Swap Rule:

Consider two robots A and B, each having destinations D_A and D_B, respectively. Let $\gamma(u, v)$ represent the path cost from location u to location v in a given map. If

$$max\{\gamma(A, D_A), \gamma(B, D_B)\} \;>\; \alpha \;\times\; max\{\gamma(A, D_B), \gamma(B, D_A)\}$$

then let A assume B's role, state, and location in the tree, and let B assume A's role, state, and location in the tree.

Table 4.6: A safer Role Swap Rule using hysteresis

In this safer Role Swap Rule, constant α is a small value greater than 1 (*e.g.* 1.1). Multiplying the maximum path cost of the scenario that would arise after a role swap means that role swaps will occur *only* when there is a significant advantage to switching roles, thus preventing oscillation.

Note that in this example a factor is multiplied to the right hand side of the equation; a similar result could be achieved by adding a small constant. For example, if robot speeds are known, then the addition to the right hand side of the equation of the maximum distance a robot might travel between replanning intervals could achieve the same result.

Parameter space

Most planning, mapping and exploration algorithms have a large number of parameters that can be adjusted to nudge the resulting behaviour in one direction or another (depending on robot platforms involved, environmental factors, desirability of certain behaviours, etc.).

The Role-Based approach presented in this chapter only depends on a small number of parameters:

- n: where n is the exponent in the equation $U(p_i) = A(p_i)/C^n(p_i)$ (which is used to calculate frontier polygon utilities, see Section 4.1.3). High values of n lead to exploration of larger spaces (such as hallways), lower values of n lead to exploration of nearby spaces (such as a series of rooms). For most experiments, both in simulation and in reality, a value of 2 was chosen for n, as this seemed to provide the ideal trade-off between exploring new, large areas while not changing direction too often.

- t_r: where t_r represents the time interval for replanning. Deciding on how often to replan is a common robotics problem. Replanning can be resource intensive and can take time, especially when multiple frontier polygons need to be found, paths need to be calculated to each, a skeleton of free space needs to be determined, and so on. On the other hand, environments can change or sudden factors can influence next decisions, so it is important not to wait too long. A typical solution is to introduce a timeout on how often a robot may replan. In the experiments conducted in simulation, replanning only occurred every ten simulation cycles, or when there were no more waypoints in the robot's current path (whichever came first). In the experiments conducted with real robots, this timeout was set to five seconds.

- T_1 and T_2: where T_1 and T_2 are thresholds used in the rendezvous point calculation process (Algorithm 2). T_1 is used to add potential rendezvous points when there are large gaps in the skeleton; T_2 is used to remove potential rendezvous points that are too close to other rendezvous points. Actual values for these thresholds depend on how thick the distribution of potential rendezvous points is desired, and on the resolution of the map.

- α: where α introduces greater or lesser degrees of hysteresis into the Role Swap Rule, as described earlier in this section.

Various aspects of Role-Based Exploration are not described by specific parameters but could nevertheless be "tuned" in one direction or another, as desired. For example, the current implementation chooses a rendezvous point that is deep in the explored environment, near the Explorer's next intended area of exploration. As discussed at the end of Section 4.1.6, there is an inherent trade-off in the rendezvous point selection. Choosing rendezvous points close to the BaseStation will mean that new information is returned more frequently, but only small amounts of information are gained each time. Choosing rendezvous points that are deep in the environment means that updates are received infrequently, but when they arrive they contain a significant amount of new information.

There are further external parameters influencing the behaviour of Role-Based Exploration (such as parameters that govern the mapping system, or parameters like sensor range, communication range, and number of robots in the team) but these are not intrinsic to the approach and are explored in greater depth in the results detailed in Chapters 5 and 6.

4.3.4 Emergent Behaviour

This section details the most surprising and perhaps the most interesting result of this thesis.

Once the role swap rule and a dynamic hierarchy had been implemented and verified in experiments, attention was turned to the effect of various hierarchy structures on the exploration effort. For example, given a team of six robots, is it better to have two branches of length 3, or three branches of length 2 (*i.e.* two Relays per Explorer, or one)? It turned out that longer chains of relays do not lead to an improvement, and can actually introduce difficulties due to the increased number of required rendezvous. Additionally, a smaller number of exploring robots means that exploration proceeds slower.

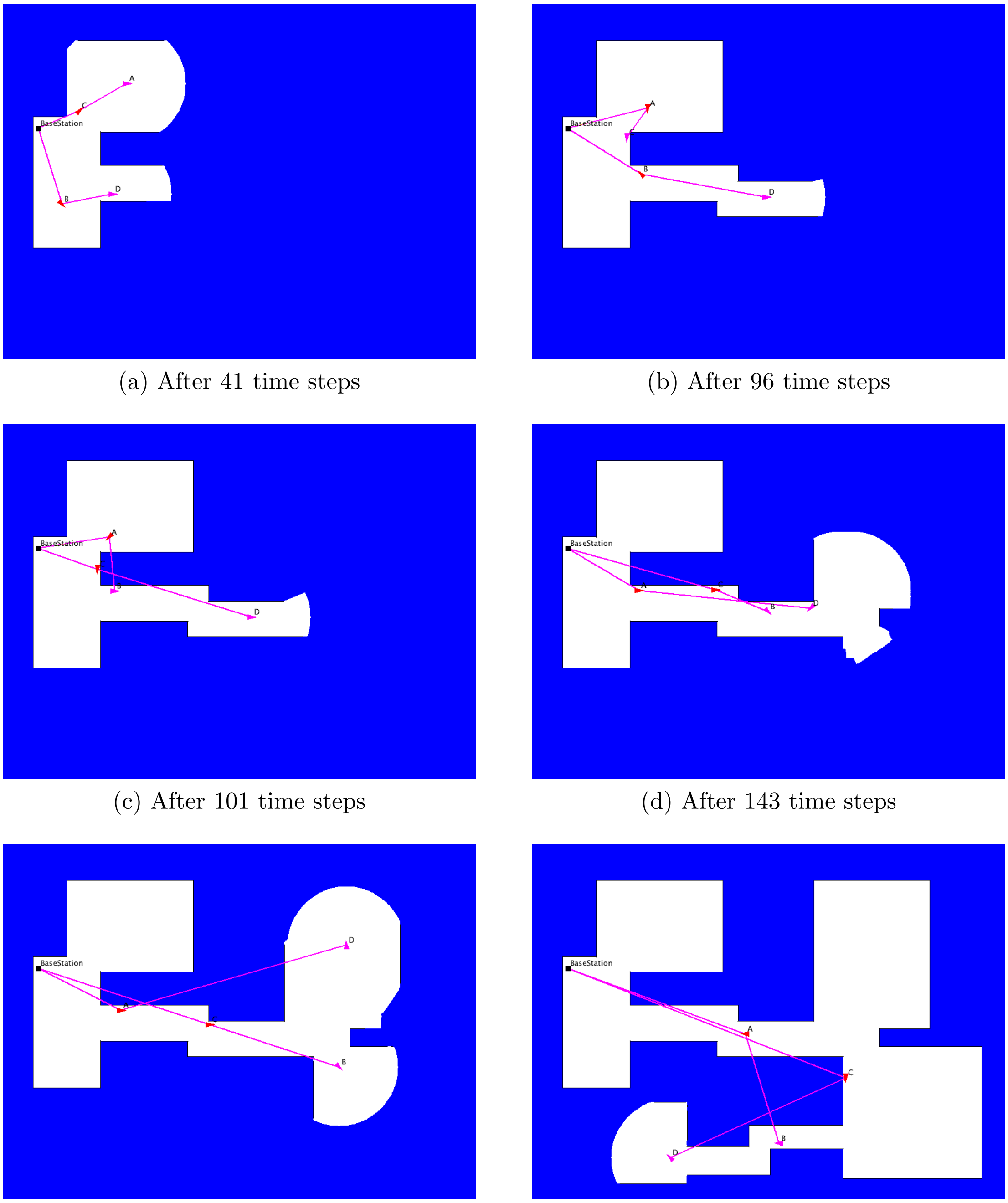
However, analysis of the runs involving many short branches, *i.e.* only a single Relay for each Explorer, revealed that when the Role Swap Rule is implemented and the hierarchy changes dynamically, *many short chains of robots can behave much like one long chain of robots.*

The example in Fig. 4.12 demonstrates this well: In Stage VI, the hierarchy is still composed of two branches of length 2. But the behaviour of the robots, and the routing of information from Explorer B to Basestation (via A, C and D) is much like what one would expect from a hierarchy having a single branch of length 4. This behaviour emerged consistently over a wide range of environments and team sizes; an additional example is presented in Fig. 4.13.

This is a significant result for coordination of any large team of robots: a simple hierarchy and a simple role exchange mechanism can lead to highly coordinated behaviour that adapts dynamically both to communication availability and to the shape of the environment.

~

This concludes the description of Role-Based Exploration. An improved method for rendezvous point calculation and the use of a Role Swap Rule led to significant improvements in the efficiency of exploration in many scenarios [36, 35]; simulation results are presented in Chapter 5 and the implementation on a real system is detailed in Chapter 6. The Role Swap Rule also led to an interesting and useful emergent behaviour that adapts to the shape of the environment. However, theoretical analysis of its efficiency is tricky, and it displays some unexpected characteristics. These are discussed in more detail in Chapter 7.



(a) After 41 time steps

(b) After 96 time steps

(c) After 101 time steps

(d) After 143 time steps

(e) After 178 time steps

(f) After 392 time steps

Figure 4.13: Screenshots from a run demonstrating the interesting behaviour that emerges from teams having hierarchies of short branches, using the role swap rule. Magenta lines indicate team hierarchy. Initially exploration proceeds as normal, with A and D as explorers, and C and B their respective relays (4.13a). After 96 time steps, A completes exploration of the top room and switches roles with C (4.13b). Soon after, C switches roles with B (4.13c). The exploration phase now involves traversal of a long hallway to the only open frontiers. The team dynamically adjusts, via role switches, in such a manner that A, C, and B act as a multi-hop link for explorer D (4.13d). As new frontiers open after 178 time steps, the team again restructures so that two robots actively explore (4.13e). When another long hall is found much later, again the team behaves like a 4-robot multi-hop chain (4.13f).

Chapter 5

Simulation Results

This chapter presents the results of an extensive series of experiments performed in simulation to examine and evaluate the behaviour of Role-Based Exploration.

Following a justification for the use of simulations, it details the algorithms that were compared to Role-Based Exploration. It further describes the systematic approach that was used to examine the effects of changes in each of four parameters: sensor range; communication range; number of robots used; and type of environment. Finally some specific scenarios are chosen in which multiple parameters are changed at the same time, and the relative advantages and disadvantages of Role-Based Exploration are presented for each of them.

5.1 Motivation for Simulations

While any new method for robotic exploration must ultimately be proven to work on real robots, there are many good reasons to examine its behaviour in simulation first. Indeed, the number of existing simulators in the robotics community (*e.g.* Player-Stage-Gazebo [52, 75], Webots [92], USARSim [21], etc.) demonstrates that the use of simulators is widespread. Some of the advantages of simulation most commonly cited include:

- **Cost**. Experiments on real robots can take much time, energy and money. It can be much cheaper to show that algorithms work in simulation before moving to reality.
- **Simplicity**. Complications experienced in the real world can be abstracted away and it is possible to focus on specific aspects of a particular part of an approach.
- **Control**. It is easy to examine performance of a method under a variety of conditions and in a variety of situations.
- **Repeatability**. Experiments can easily be performed multiple times and shown to work (or not work) consistently.

To examine the behaviour of Role-Based Exploration, a custom simulator, the Multi-Robot Exploration Simulator (MRESim), was developed. MRESim is described in greater detail in Appendix A.

MRESim simplifies real scenarios in a number of ways, as discussed in Appendix A. Nevertheless, it provides a useful platform for comparison of Role-Based Exploration to a number of existing algorithms.

5.2 Algorithms Used For Comparison

Several multi-robot exploration algorithms were implemented in order to conduct a comprehensive comparison:

1. **Leader-Follower Exploration (LF)**

   Having much in common with several of the approaches discussed in section 2.2.3, in this approach a lead robot explores using frontier-based exploration. When the leader is almost out of range, a second robot follows it to maintain a communication link. When the second robot is almost out of range, a third robot follows it (and so on, to the maximum number of robots in the team). The expected benefits of this approach are that all team members are expected to stay connected. However, the extent of exploration is limited by the team's communication range.

2. **Greedy Exploration (G)**

   Closely related to the approaches discussed in section 2.2.4, in this approach robots opportunistically seek to expand their knowledge by navigating to new, unexplored frontiers. When robots are within range of one another, they coordinate their exploration using the algorithm presented in section 4.1.4. The expected benefits of this approach are fast exploration since all robots in the team are exploring. However, there is no explicit effort to relay information to the BaseStation, or to maintain communication connections between team members.

3. **Greedy Exploration with Periodic Return (GP)**

   The same method as above, except that robots return periodically to relay their knowledge to the BaseStation. The purpose of this approach is to ensure that newly gained knowledge reaches the BaseStation. To allow for complete exploration of the environment (and prevent oscillation), the timeout for return to the BaseStation is increased with each iteration. In other words, if the robot initially explores for 2 minutes before returning, then it will subsequently explore for 4 minutes before returning; then 6 minutes; and so on.

4. **Role-Based Exploration *without* Role Swaps (RB1)**

   As described in chapter 4, with a static hierarchy. Teammates are assigned roles at the start of the effort, and *do not* swap roles.

5. **Role-Based Exploration *with* Role Swaps (RB2)**

   As described in chapter 4, with a dynamic hierarchy. Teammates *do* swap roles according to the role swap rule discussed in section 4.3.

5.3 Methodology

There are a number of factors that can influence the relative success or failure of a given algorithm. These include: sensor range; communication range; number of robots in the team; and type of environment (some exploration algorithms may perform better in open spaces, others when there are many obstacles). Many different sets of values for these parameters were experimented with over the course of this research, before the methodology described here was developed. The factors were systematically examined as follows:

First, to have a benchmark to compare changes in parameters against, a simple set of values was chosen for the parameters, and a set of runs for each algorithm was conducted in an empty room.

Second, the parameters were adjusted one by one and the resulting behaviours, still in an empty room, were compared against the benchmark. Section 5.5 explores the effect of changes in sensor range, section 5.6 explores changes in communication range, and section 5.7 discusses the effects of using a variable number of robots.

Third, all algorithms were tested in a variety of environments to examine whether some algorithms are better suited to certain types of environments than others. Section 5.8 presents the results of these runs.

Finally, section 5.9 looks at a few specific scenarios in which several parameters are changed at the same time with the goal of emulating, to some degree, scenarios that a team of robots might be used for in the real world.

Unless otherwise indicated, each set of results presented is based on an average of at least three runs for each algorithm in that scenario.

5.4 Benchmark

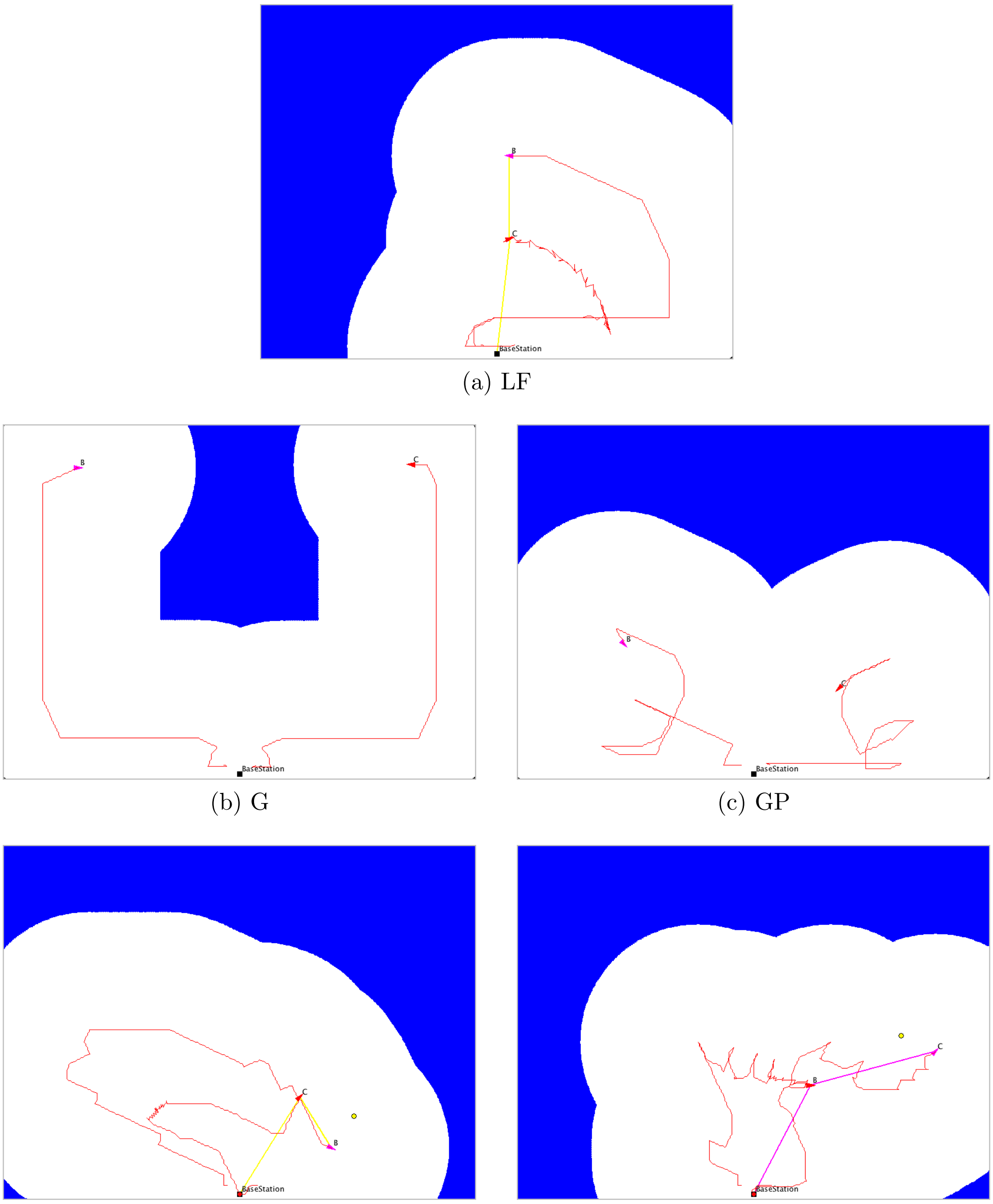
The initial, "benchmark" setup involves a team of two robots exploring an empty room. Each robot has a communication range of one quarter the width of the room, and a sensor range also of one quarter the width of the room (for an environment represented by an occupancy grid of 800×600 cells, this meant a sensor range of 200 and a communication range of 200). Choosing these values means that significant parts of the environment remain beyond sensor and communication range. While this may be a very simple and unlikely scenario, it is useful to use as a benchmark for evaluating changes in the various parameters affecting each approach. Results for the benchmark scenario, measured using each of the five performance metrics discussed in Section 3.3, are presented in Fig. 5.2. Screenshots showing the progress of each algorithm after 250 simulator cycles are presented in Fig. 5.1.

As was to be expected, Leader-Follower Exploration (LF) performs best regarding metrics 3 (information sharing) and 4 (responsiveness). In other words, all robots remain in communication range of one another and the BaseStation throughout the exploration effort. Information is perfectly shared throughout the team, and responsiveness to BaseStation commands is near instantaneous. Even exploration proceeds fairly quickly, outperforming the other algorithms in the early stages. This is due to the fact that in an empty space such as the one examined here, the exploring robot will reach the end of the communication chain and then search frontiers nearby. The effect is that its relaying robots act like a fan, sweeping all terrain between BaseStation and Explorer. In more complicated environments, it will be shown that this speed of exploration soon disappears. In addition, LF exploration is not able to fully explore the environment: some areas remain out of range, even when the team is stretched to its limit.

Greedy Exploration (G) is by far the fastest method regarding metric 1 (exploring the full environment). Intuitively this makes sense: if all robots are continuously exploring, then surely this should be faster than approaches where half (or more than half) of the robots are relaying. The drawbacks of G exploration become evident however when examining the other metrics. Information is returned to the BaseStation slower and less frequently, information sharing between teammates is significantly worse than any of the other approaches, and robots spend a long time out of range of the BaseStation.

Greedy Exploration with Periodic Return (GP) tries to improve on G's lack of responsiveness and information sharing by returning periodically to provide the BaseStation with the latest information update. While the responsiveness and information sharing are clearly improved, the speed of exploration suffers. Particularly as the efforts wears on, repeated long walks back and forth to the BaseStation mean that this approach becomes very slow, and takes the longest to reach completion.

Role-Based Exploration without role swaps (RB1) and with role swaps (RB2) perform quite similarly, and provide a trade-off between the speed of Greedy exploration and the responsiveness of Leader-Follower exploration. Indeed, the role-based approaches provide the best performance for metric 2 (return of information to the BaseStation) over the course of the full effort, and the team shares information well. These are highly desirable characteristics for applications such as search-and-rescue,



(a) LF

(b) G

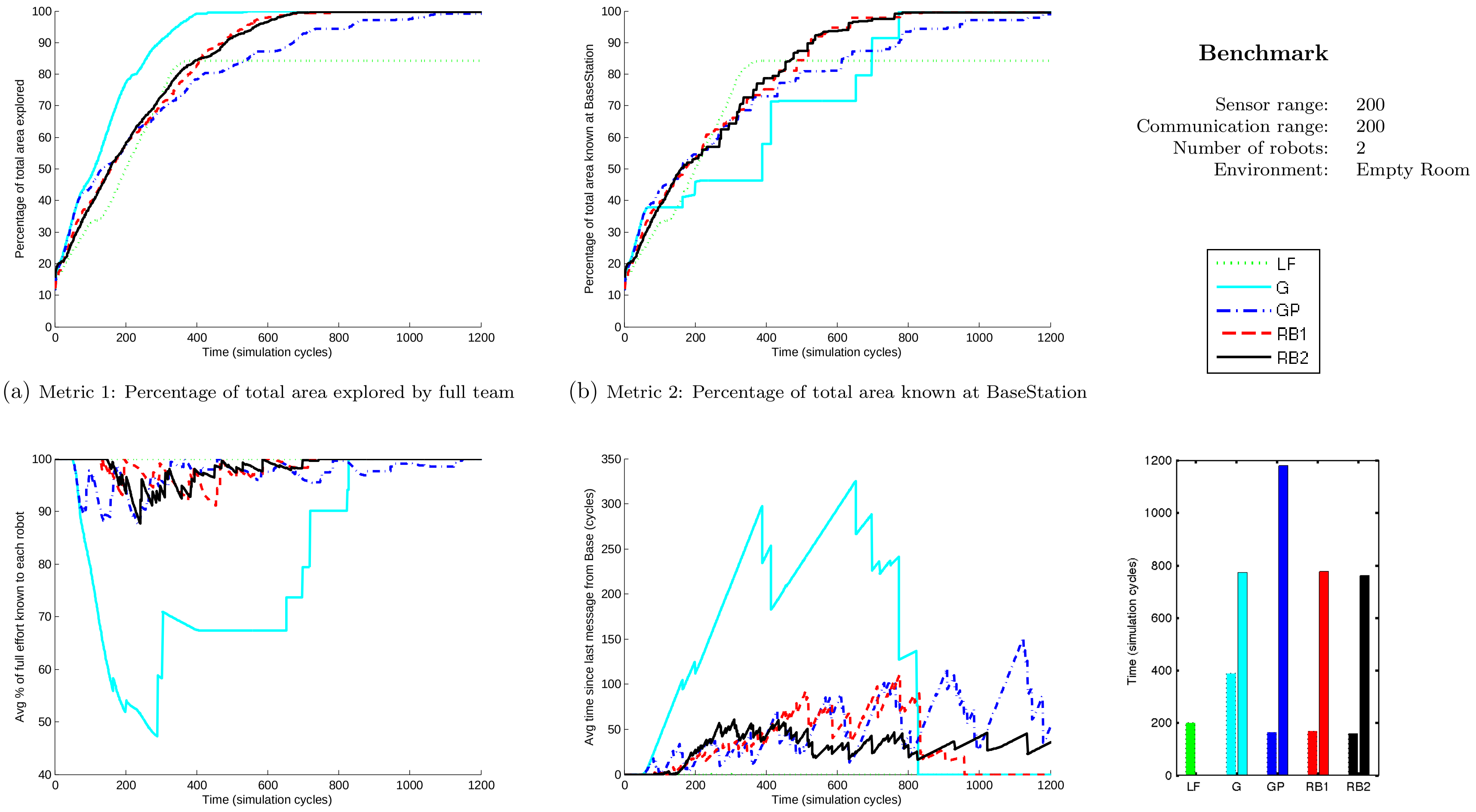
(c) GP

(d) RB1

(e) RB2

Figure 5.1: Screenshots of the progress of each algorithm in the “Benchmark” scenario after 250 simulator cycles

where human responders will want regular updates.



(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

Benchmark

Sensor range:	200
Communication range:	200
Number of robots:	2
Environment:	Empty Room

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.2: Results for the “Benchmark” scenario

The reason for the similarity in performance of both RB1 and RB2 is that an empty room does not introduce many reasons to switch roles – one robot will soon be deeper than the other, and role swaps are unlikely. The real advantages of swapping roles will become clearer in later sections.

5.5 Variable Sensor Range

This section explores the effect of how a relative increase or decrease in sensor range (all other factors remaining equal) can affect the performance of the exploration algorithms. Results for each of the five performance metrics in the empty room using a sensor range of 50 (instead of 200) are presented in Fig. 5.4; results using a sensor range of 300 are presented in Fig. 5.5. (A value of 50 corresponds to 1/16 the width of the room; a value of 300 corresponds to 3/8 the width of the room.)

As is to be expected, all algorithms explore faster with greater sensor range.

The behaviour of LF exploration remains the same: information sharing and responsiveness is perfect, but the environment is never fully explored.

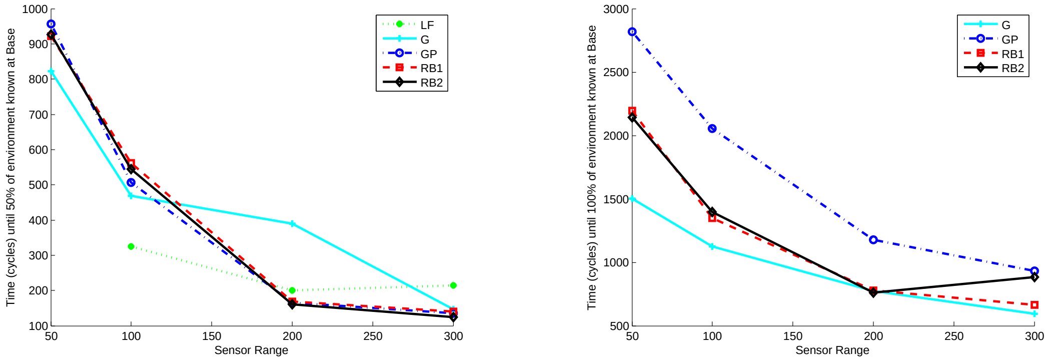
At very small sensor range, G exploration is by far the fastest at covering the environment; however the larger the sensor range becomes, the less difference there is between the algorithms regarding speed of exploration. The reasons for this are straightforward: given that the environment is a room, a single exploring robot (as in the LF, RB1 and RB2 approaches) will explore it almost as fast as two robots when the sensor range approaches the width of the room.

Relay of knowledge to the BaseStation is consistently performed better by the RB algorithms, at least in the early stages of exploration. Under short range, G exploration surprisingly shows better performance relative to the benchmark runs. This is an interesting side effect that occurs in G exploration when the sensor range is shorter than the communication range: greedy robots with a short sensor range are more likely to wander back and forth across the environment as they cover unknown space, and hence are more likely to wander into communication range of the BaseStation by chance.

Information sharing and responsiveness are not highly affected by sensor range (except for G exploration, for the same reason mentioned above). The RB algorithms outperform both of the G and GP algorithms in this respect, with RB2 showing slightly better responsiveness than RB1 in general.

Summing up, the effect of sensor range on the relative performance of the five algorithms is not hugely significant. In all cases, G exploration remains the fastest for total exploration, LF exploration remains the best for connectivity of the team, and RB exploration presents the best trade-off between the speed of exploration and team connectivity. Again the differences between RB1 and RB2 are not significant as the nature of the environment is such that role swaps do not occur often.

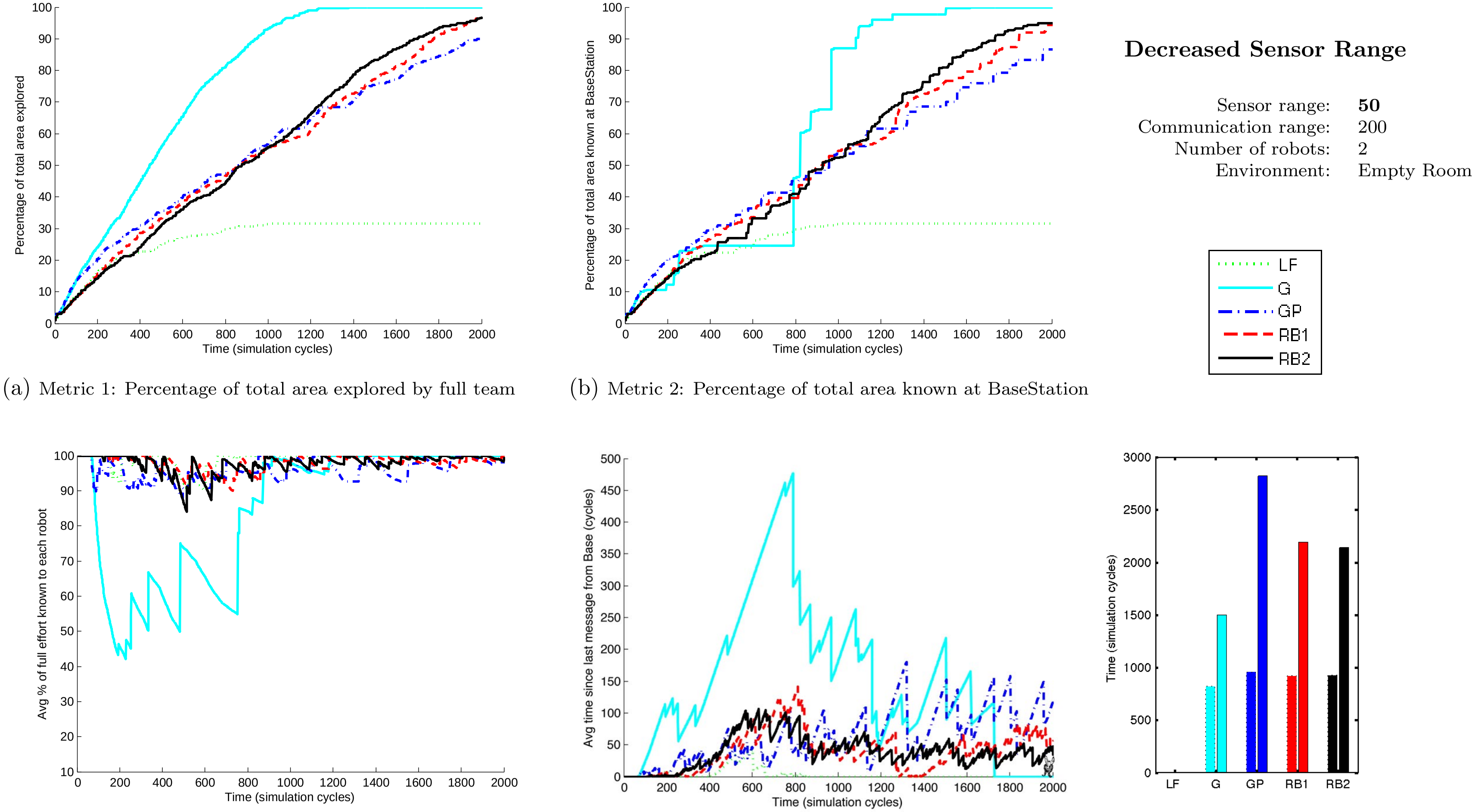
This is confirmed by Fig 5.3: larger sensor range leads to faster exploration, but relative to one another the algorithms are not hugely affected.



(a) The effect of sensor range on discovering 50% of the environment at BaseStation

(b) The effect of sensor range on discovering 100% of the environment at BaseStation

Figure 5.3: The effect of sensor range



Decreased Sensor Range

Sensor range:	**50**
Communication range:	200
Number of robots:	2
Environment:	Empty Room

(a) Metric 1: Percentage of total area explored by full team

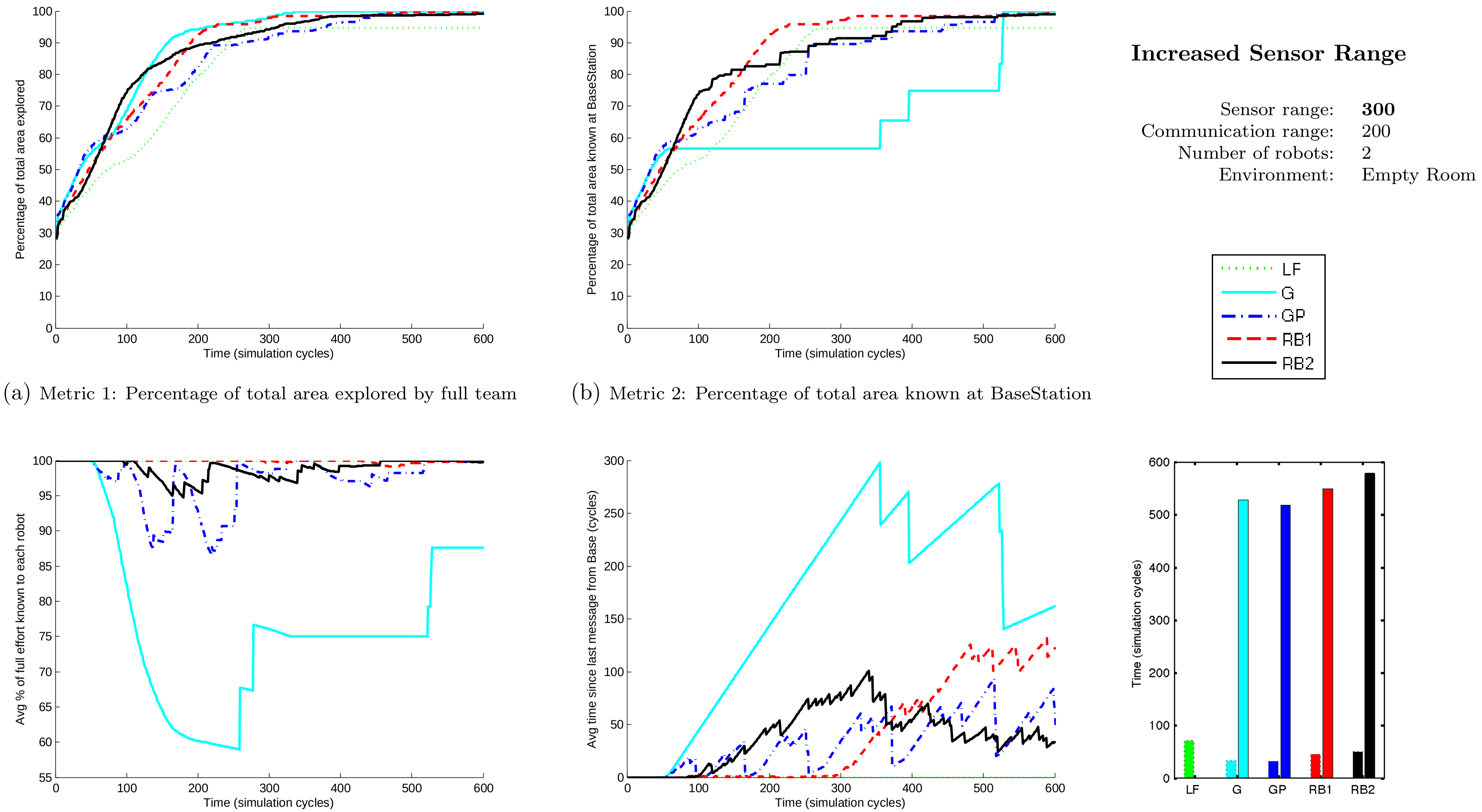
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.4: Results for decreased sensor range



Increased Sensor Range

Sensor range:	**300**
Communication range:	200
Number of robots:	2
Environment:	Empty Room

(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.5: Results for increased sensor range

5.6 Variable Communication Range

This section explores the effect of how a relative increase or decrease in communication range (all other factors remaining equal) can affect the performance of the exploration algorithms. Results for each of the five performance metrics in the empty room using a communication range of 50 (instead of 200) are presented in Fig. 5.7; results using a communication range of 300 are presented in Fig. 5.8. (Again, a value of 50 corresponds to 1/16 the width of the room; a value of 300 corresponds to 3/8 the width of the room.)

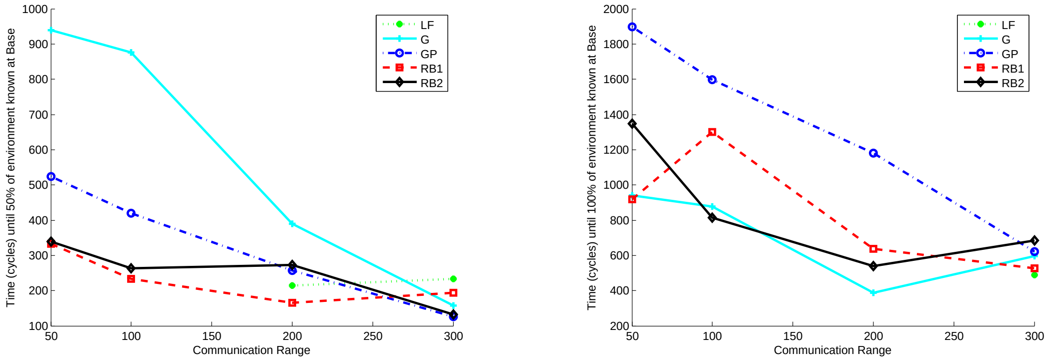
As is to be expected, all algorithms explore faster when they have a greater communication range.

When communication range is increased to 300, all five algorithms start to behave fairly similarly. G and GP exploration demonstrate inferior information sharing and responsiveness as compared with the LF and RB approaches, but in general the algorithms explore at a fairly similar pace and information is returned to BaseStation at similar rates. This is to be expected: when two robots each have a communication range close to half the width of the room, they will explore the room fully fairly quickly in any case.

Of considerably greater interest are the algorithms' relative behaviour when communication ranges are small: here the algorithms' relative strengths and weaknesses become much more pronounced. G exploration is once again the much faster method for total exploration, but with poor communication, relay to the BaseStation is very poor indeed. LF exploration maintains perfect connectivity, but only explores 20% of the environment. The role-based approaches show a good trade-off between these two extremes, and as far as information relay to BaseStation is concerned show significantly better performance. In this regard, RB2 is also slightly better than RB1.

In short, the smaller the communication ranges of the individual robots, the greater the relative advantages of role-based approaches over greedy or leader-follower approaches.

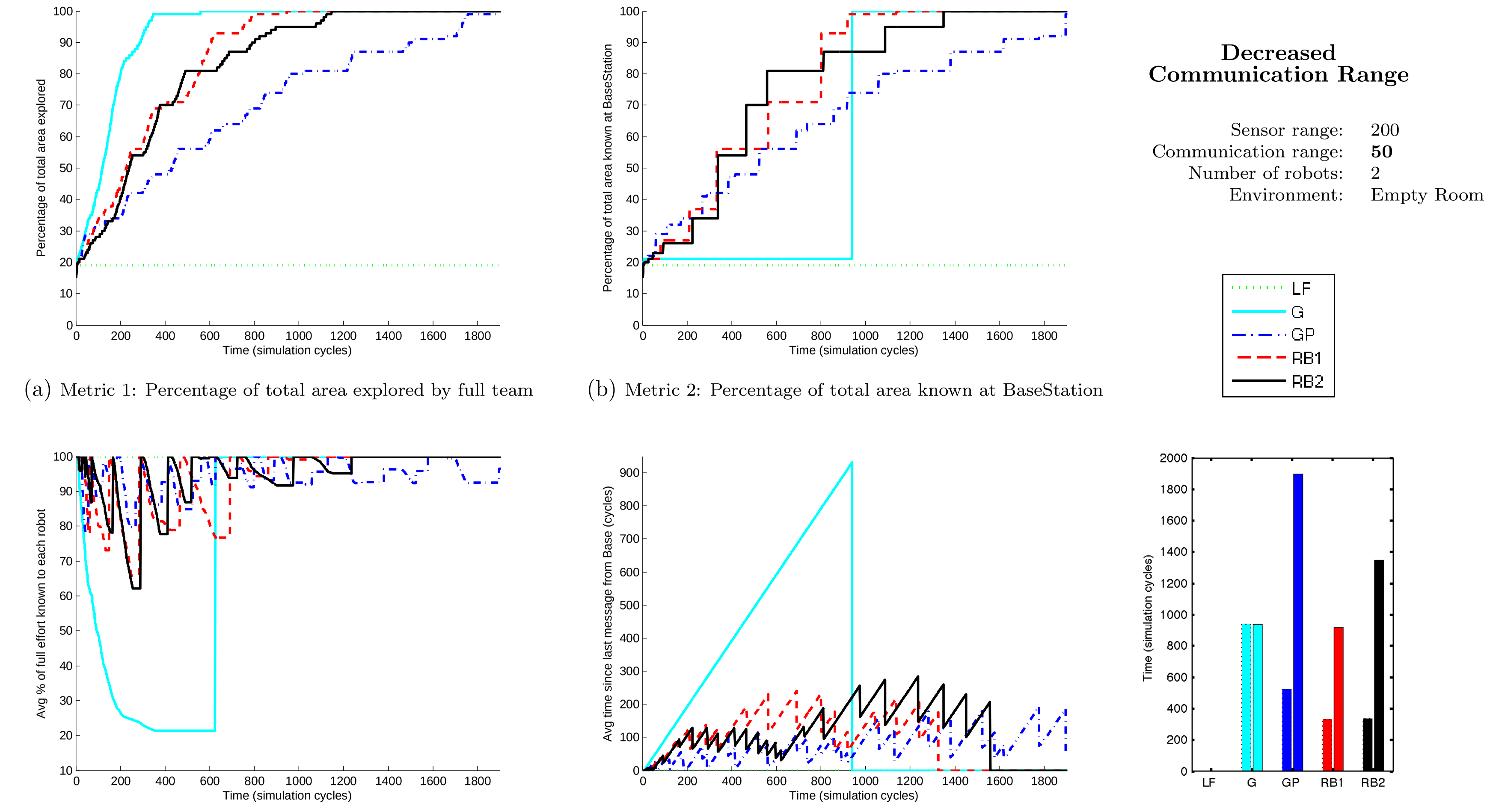
This is confirmed by Fig 5.6: as communication range decreases, the time it takes to return knowledge to the BaseStation become better and better for the role-based approaches relative to the G, GP and LF approaches (although, it must be noted that this is more true for intermediate stages of the exploration effort – such as attaining knowledge of 50% of the environment – than it is for exploration of the full environment).



(a) The effect of communication range on discovering 50% of the environment at BaseStation

(b) The effect of communication range on discovering 100% of the environment at BaseStation

Figure 5.6: The effect of communication range



Decreased Communication Range

Sensor range:	200
Communication range:	**50**
Number of robots:	2
Environment:	Empty Room

(a) Metric 1: Percentage of total area explored by full team

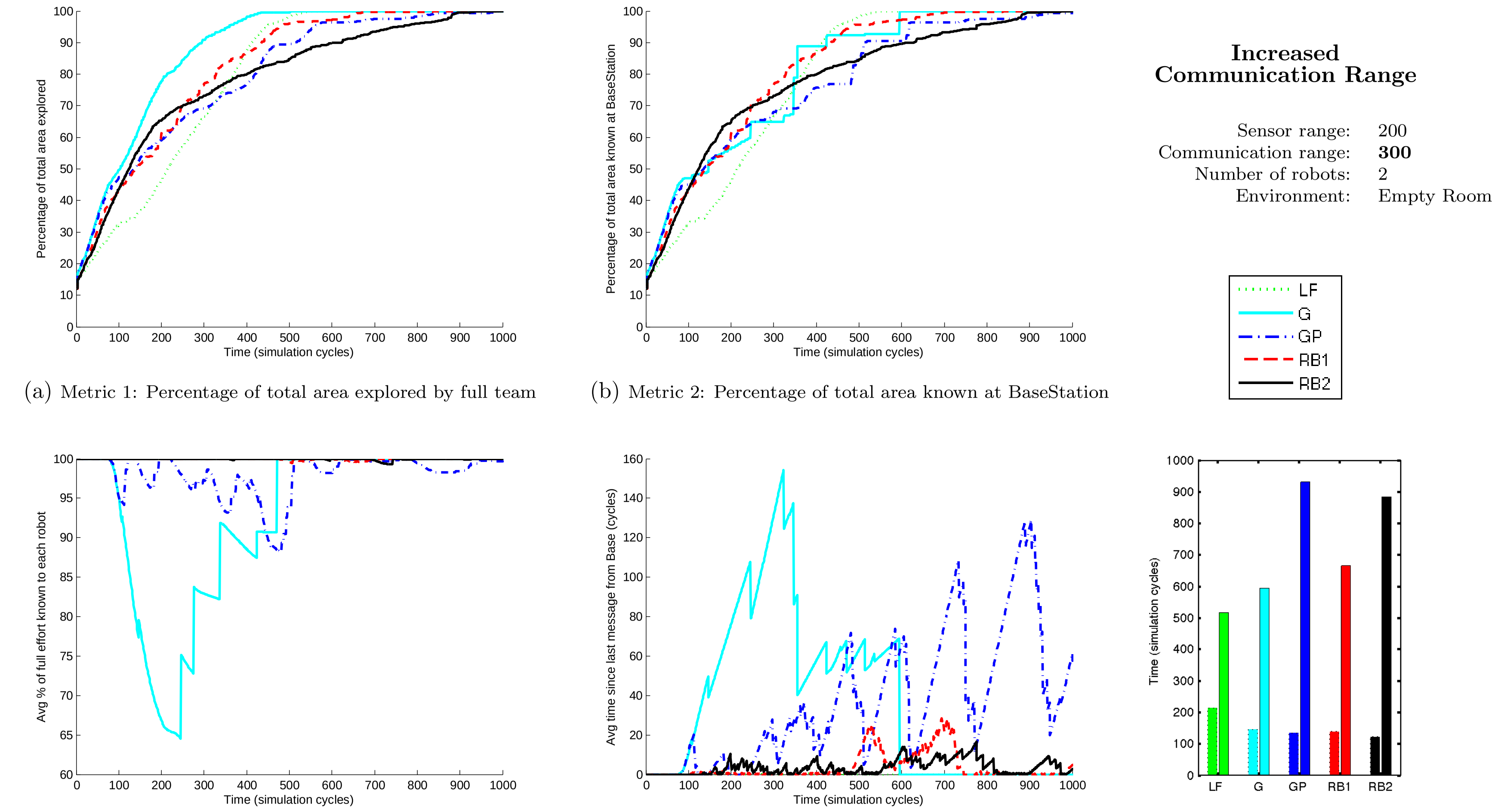
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.7: Results for decreased communication range



Increased Communication Range

Sensor range: 200
Communication range: **300**
Number of robots: 2
Environment: Empty Room

(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.8: Results for increased communication range

5.7 Variable Team Size

This section explores the effect of how team size (all other factors remaining equal) can affect the performance of the exploration algorithms. Results for each of the five performance metrics in the empty room using a team of 4 robots (instead of 2) are presented in Fig. 5.10; results using a team of 6 robots are presented in Fig. 5.11.

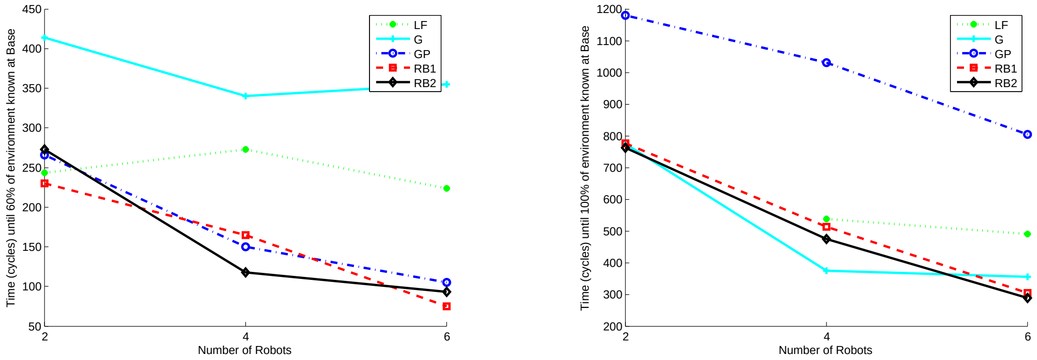
As is to be expected, all algorithms explore faster when they have a larger number of robots in the team.

Concerning total exploration speed, the more robots that are in the team, the more and more RB1 and RB2 approach the exploration speed of G. GP does not show significant improvement with more robots; this is due to the fact that they are all spending much time returning to the BaseStation. LF also does not show a significant improvement, except that finally the environment can be fully explored.

With increasing numbers of robots, knowledge at the BaseStation is not improved for G exploration; even though there may be more robots, this does not make any of them more inclined to return to the BaseStation. With a large number of robots, RB1 and RB2 however show even better information relaying ability. Having more relaying robots in the team means that the likelihood of chance encounters increases, so information is likely to spread through the team and back to the BaseStation faster.

RB1 and RB2 perform similarly well; again the responsiveness of RB2 is slightly better than that of RB1. This is due to the fact that with more robots there is a higher likelihood of chance encounters between robots. Without role swaps this doesn't make much difference, but with role swaps, each chance encounter represents a potential further improvement of the efficiency of exploration.

In short, the greater the number of robots in the team, the better the performance of Role-Based Exploration relative to the competing algorithms. This is confirmed by Fig 5.9: all algorithms improve with increasing numbers of robots, but the improvements of the RB approaches are relatively greater, particularly regarding speed of full exploration[1].

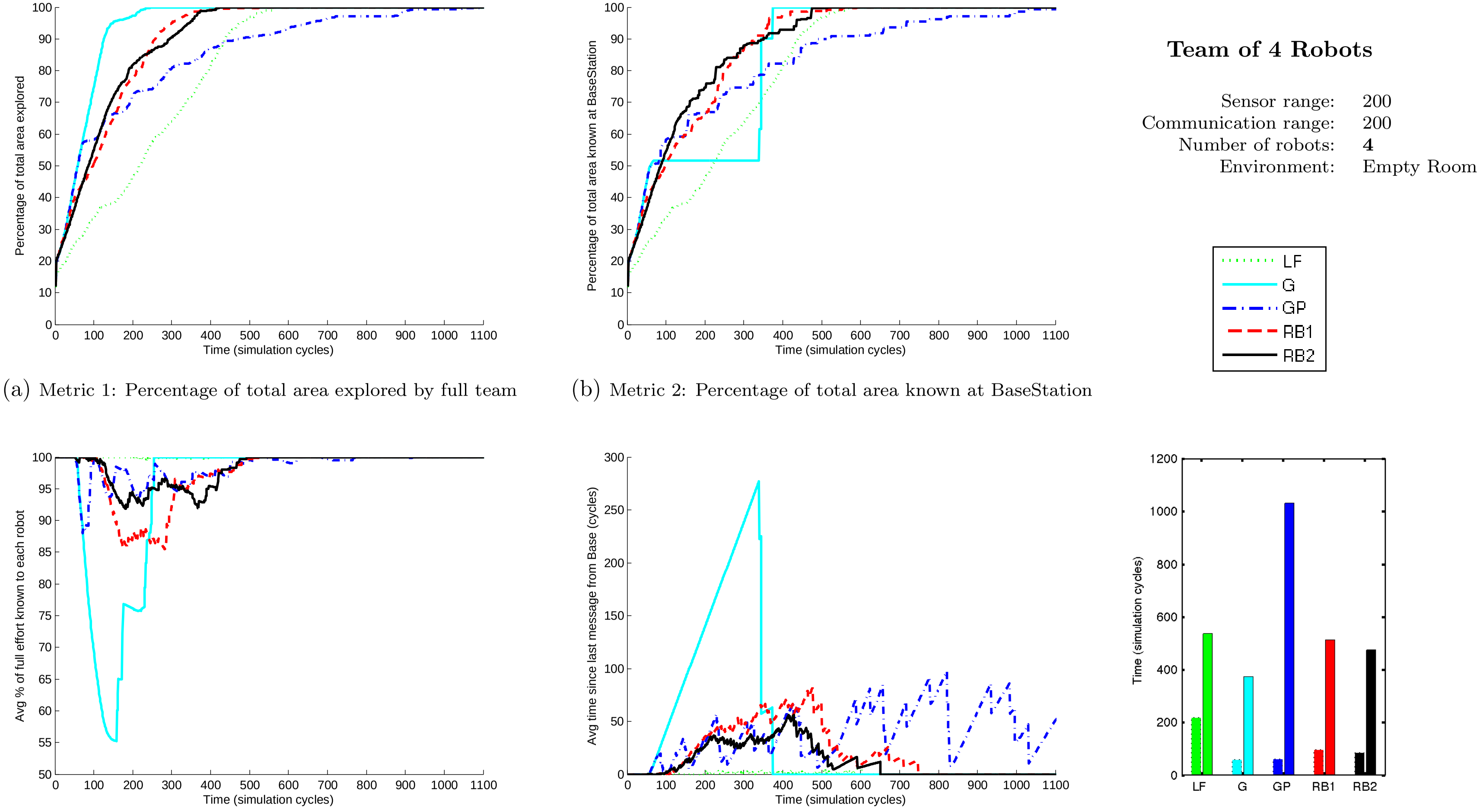


(a) The effect of number of robots on discovering 60% of the environment at BaseStation

(b) The effect of number of robots on discovering 100% of the environment at BaseStation

Figure 5.9: The effect of number of robots

[1] It must be pointed out that unlike in the previous two sections, here a threshold of 60% was used for Fig. 5.9a, and not 50%. As Figs. 5.10 and 5.11 show, 50% is near a major information update for G exploration, and led to slightly misleading plots; using 60% as a threshold better demonstrated trends in performance).



(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

Team of 4 Robots

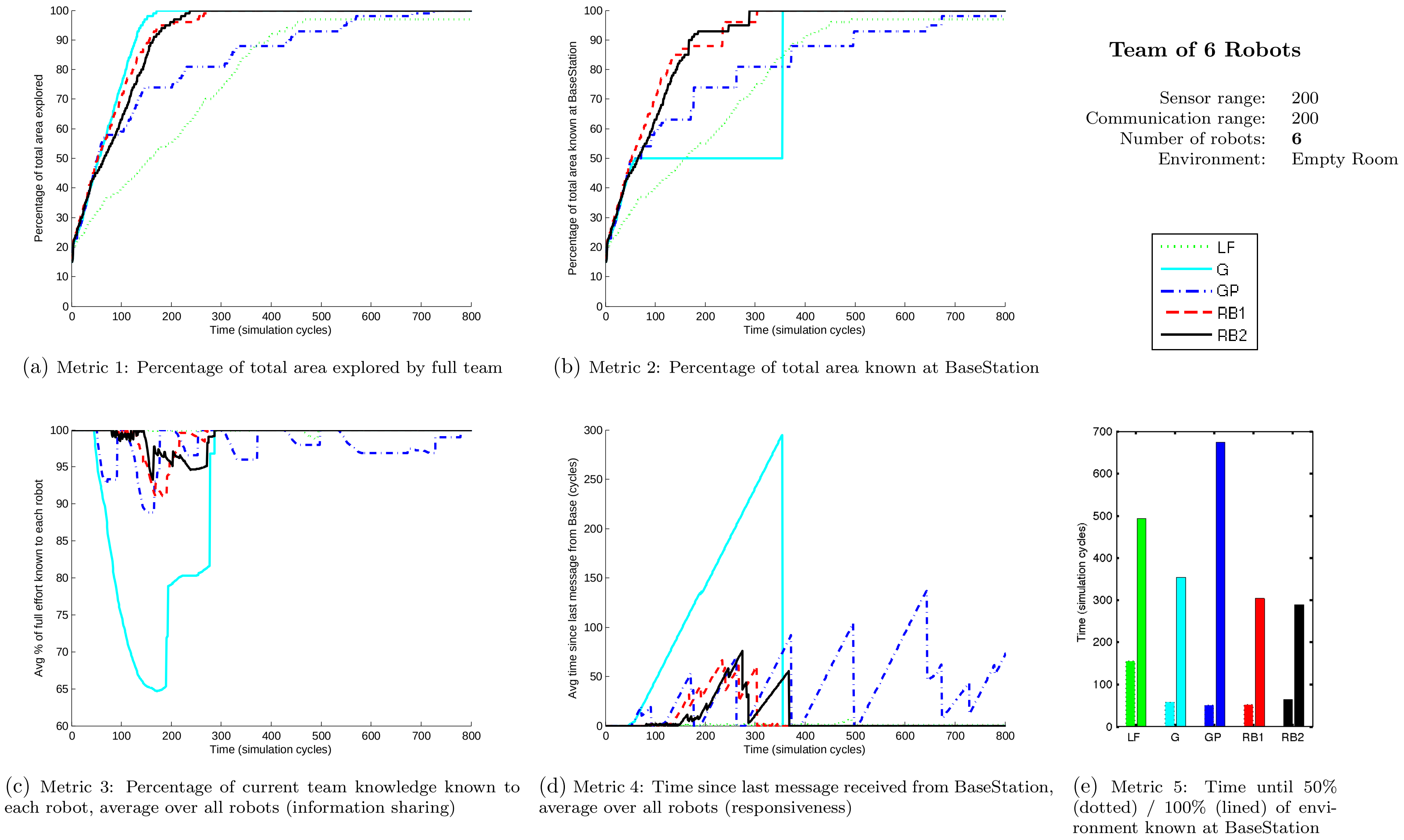
Sensor range:	200
Communication range:	200
Number of robots:	**4**
Environment:	Empty Room

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.10: Results for a team of 4 robots



(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.11: Results for a team of 6 robots

5.8 Variable Environment Types

This section explores the effect of environment types on the relative performance of the exploration algorithms. There is a very wide range of possible environments that robots may be used to explore (consider for example the surface of Mars compared with a pile of rubble after a building collapse), and some exploration algorithms may perform better in certain environments than others.

Four categories of environments were compared:

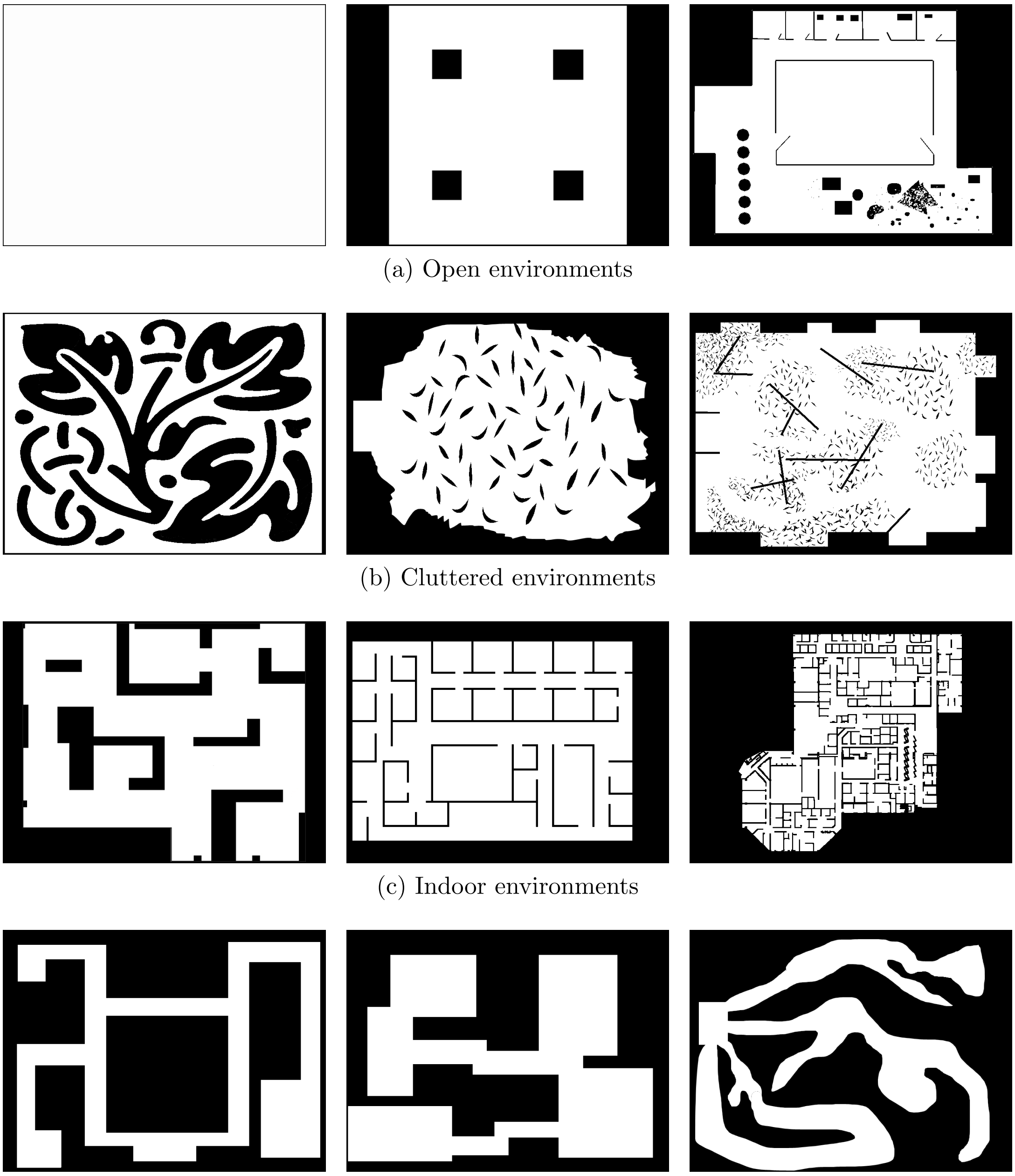
- **Open**: These environments contain some obstacles but are primarily composed of open spaces. A real world equivalent application would be for example a team of unmanned aerial vehicles trying to find a lost hiker in the countryside or a ship at sea.

- **Cluttered**: These environments contain many small and fragmented obstacles. A real world equivalent application would be search and rescue in rubble.

- **Indoor**: These environments contain rooms and hallways, and are likely to contain large loops. A real world equivalent would be reconnaissance and mapping of an unknown building.

- **Elongated**: These environments are characterised by features such as long, narrow spaces, often leading to a dead end. A real world equivalent application would be exploration of mines.

Several environments were created for each environment category; three examples for each category are provided in Fig. 5.12. To show the full results from each environment would be impractical, but to average results over a number of environments could lead to misleading data, since some environments will be explored much faster than others. Consequently, in this thesis a sample set of results is presented for



(a) Open environments

(b) Cluttered environments

(c) Indoor environments

(d) Elongated environments

Figure 5.12: Environment type examples

one environment from each category (always the middle one in Fig. 5.12); these are representative of typical behaviours of each algorithm in that environment category.

Open environments

Behaviour of the algorithms in open environments has been performed in all of the experiments discussed thus far, since both the benchmark and its variations were conducted in an open room. In short, G exploration explores the fastest, but doesn't relay information well to base; LF maintains the best link to BaseStation and the best communication within the team, but doesn't fully explore the environment; and both RB1 and RB2 provide a trade-off between the two with reasonably fast exploration and good connectivity (with RB2 usually slightly outperforming RB1). Tests in a different open environment, with results shown in Fig. 5.13, confirm these conclusions.

Cluttered environments

The results in a cluttered environment are presented in Fig. 5.14. Performance is markedly different from performance in open environments. While G exploration is faster still regarding speed of total exploration, relative to the other approaches its ability to relay information back to the BaseStation is even worse. This is likely due to the fact that presence of clutter decreases the chances of an exploring robot wandering back into communication range of the BaseStation haphazardly, as the BaseStation's range is significantly reduced.

Even more noticeable is a significant difference in the relative performances of RB1 and RB2. Clutter, in the environments tested here, involved many small obstacles, and hence a large number of small loops. Loops mean that Explorers are likely to double back and meet their parent Relay, which is then often in a more advantageous position to explore the next frontier. The use of role swaps takes advantage of these scenarios, leading to better exploration.

Indoor environments

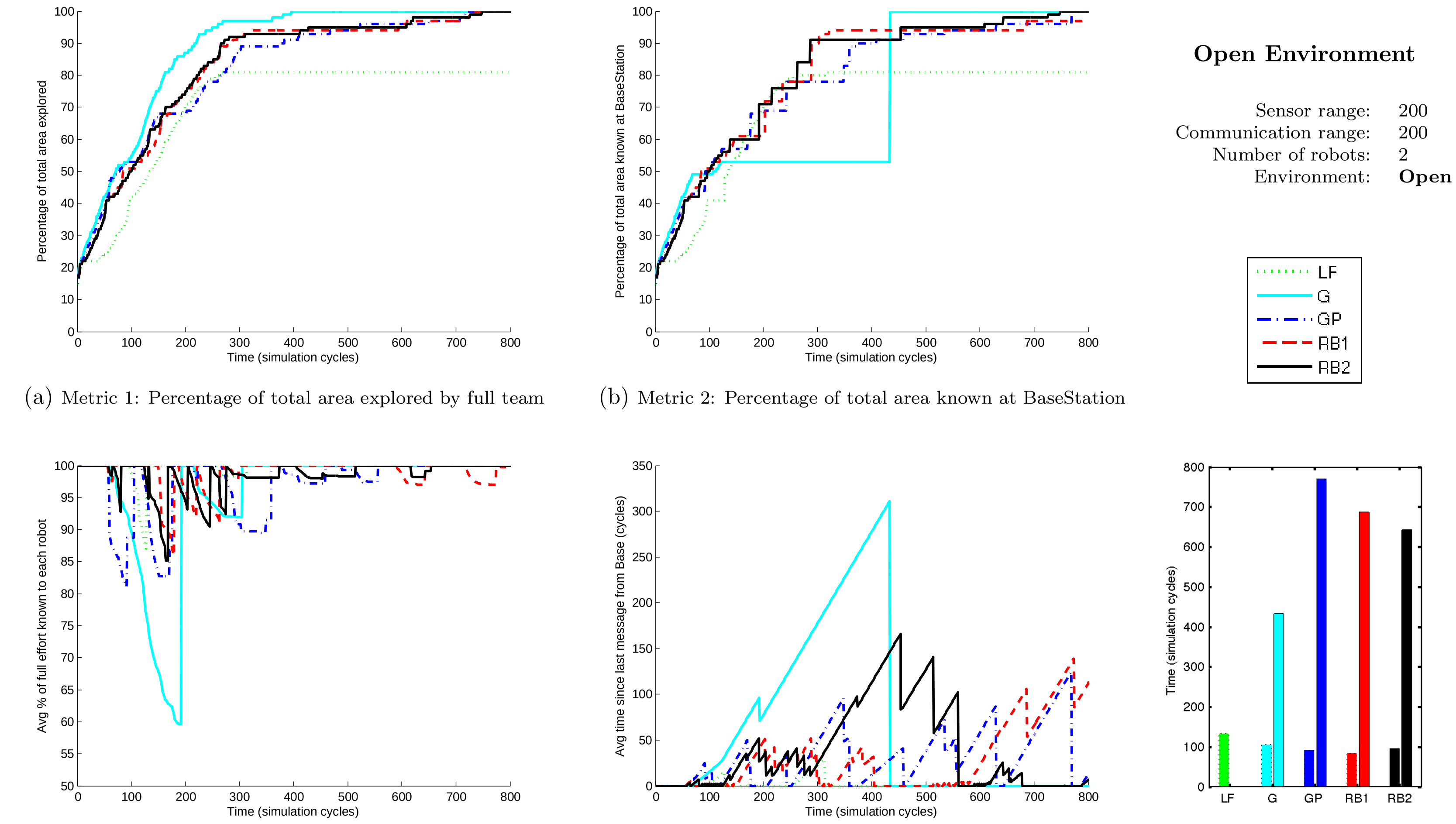
Results in an indoor environment are presented in Fig. 5.15, and behaviour closely mirrors behaviour in open environments, except that the relative advantages and disadvantages of the algorithms are more strongly pronounced. It was originally anticipated that role-swapping would lead to significant improvements in RB2 over RB1, but as it turns out, they behave very similarly. This can be explained by the fact that the indoor environments tested here have long hallways, and at rendezvous the Explorer will therefore almost always be closer to the next frontier than the Relay (meaning minimal role swaps).

Elongated environments

Finally in elongated environments (results presented in Fig. 5.16), the role-based approaches explore almost as fast as G exploration, and are easily the better choices for returning information to the BaseStation. This is particularly true when the number of halls (or passages, or dead-ends) is smaller than the number of robots. If there are n passages and n robots, G exploration will reach full exploration faster since each passage will be explored by one robot. However, if there are n passages and $2n$ robots, then greedy exploring robots will only duplicate one another's work, whereas a team of role-based robots will have one robot exploring each passage and one robot relaying from each passage.

Summing Up

The advantages and disadvantages of the five algorithms as discovered in earlier sections remain the same in all types of environments. However, G exploration is particularly poor at relaying to BaseStation in cluttered and elongated environments, and the case for using a role-based approach in such scenarios is even stronger. Role swaps turn out to provide the greatest improvement to role-based exploration when there are many small loops or dead-ends in the environment. This result is fully in line with the original motivation for introducing role swaps in the first place, as outlined in section 4.3.1.



(a) Metric 1: Percentage of total area explored by full team

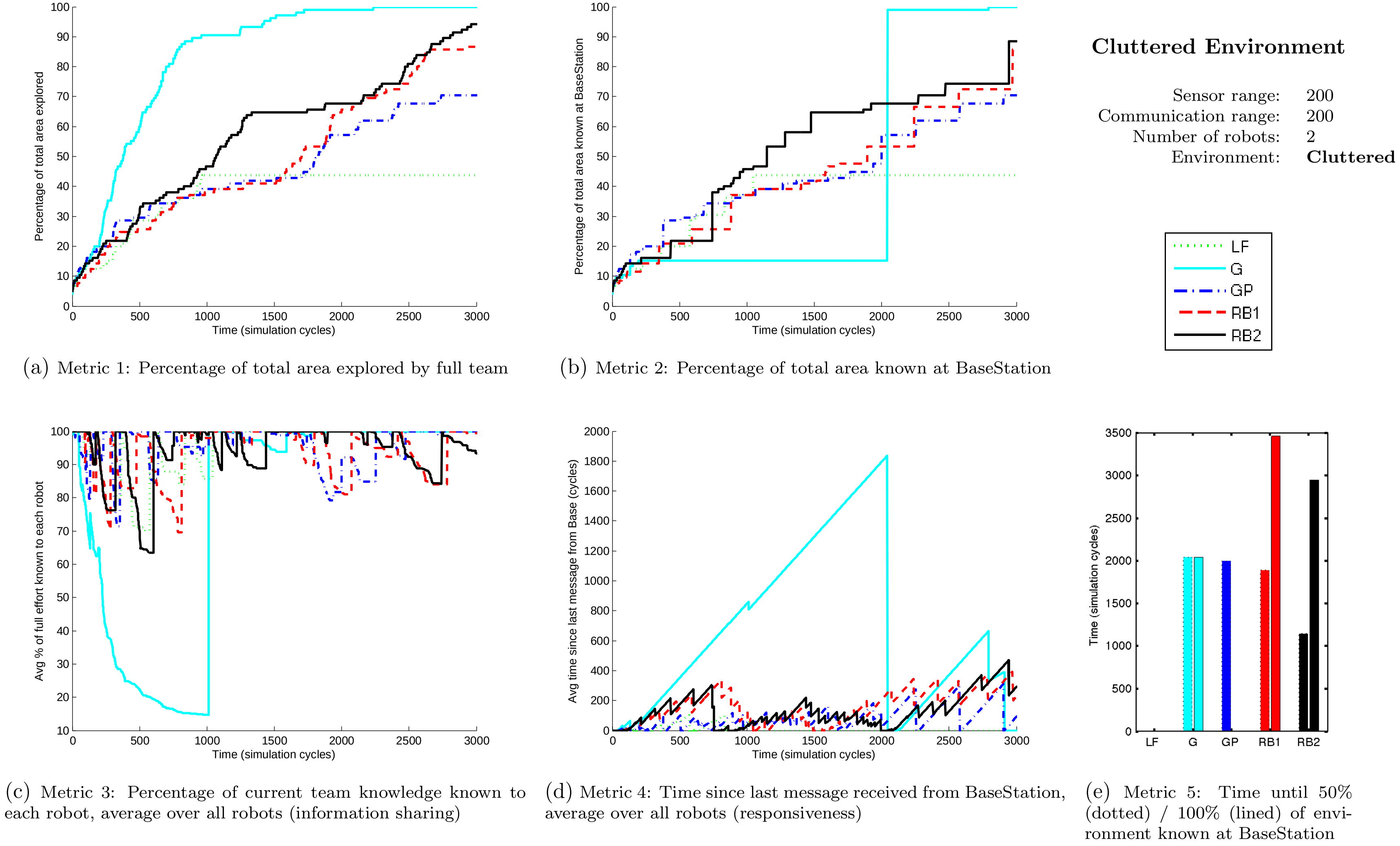
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.13: Results in an open environment



Cluttered Environment

Sensor range:	200
Communication range:	200
Number of robots:	2
Environment:	**Cluttered**

(a) Metric 1: Percentage of total area explored by full team

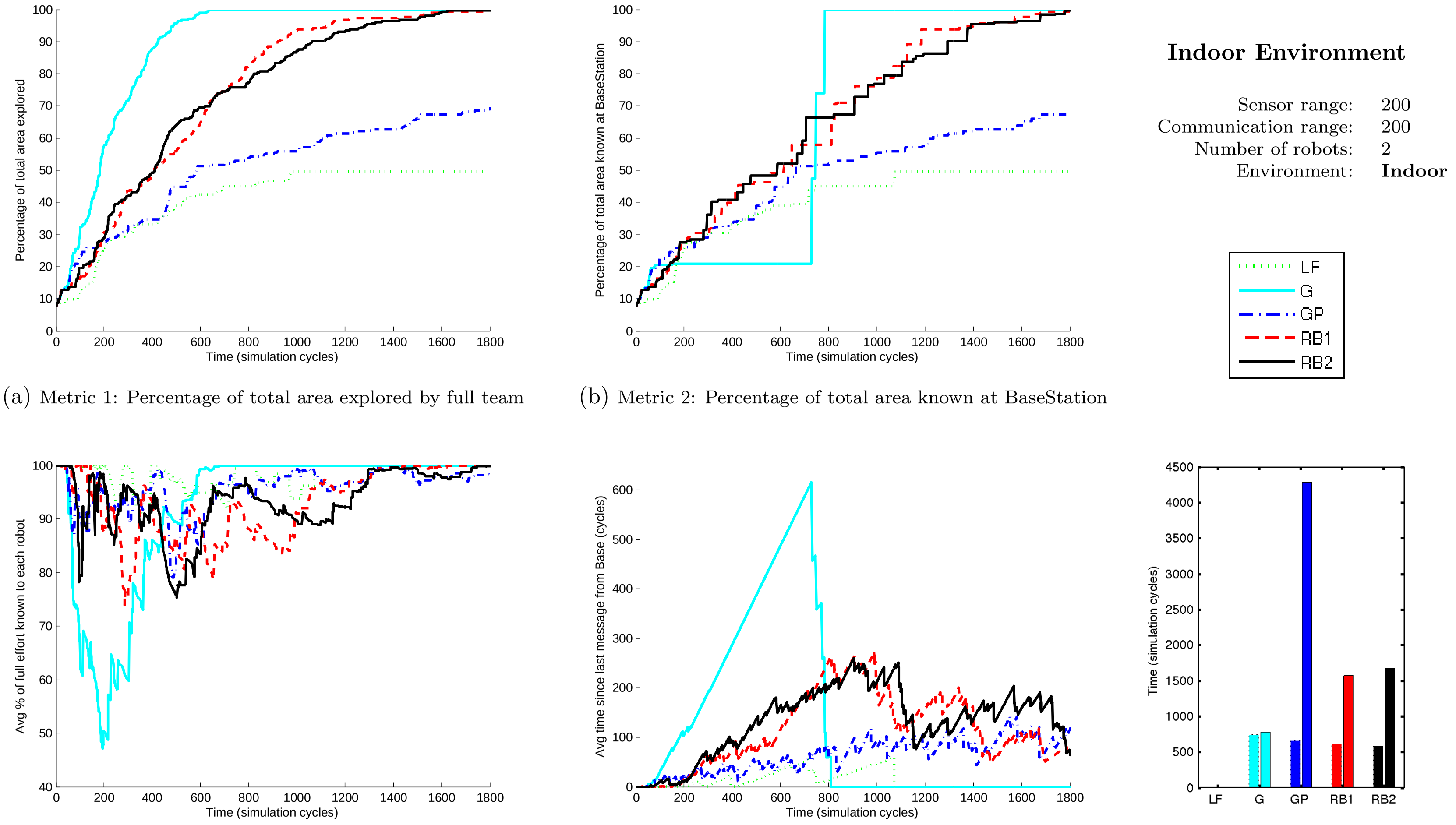
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.14: Results in a cluttered environment



Indoor Environment

Sensor range:	200
Communication range:	200
Number of robots:	2
Environment:	**Indoor**

(a) Metric 1: Percentage of total area explored by full team

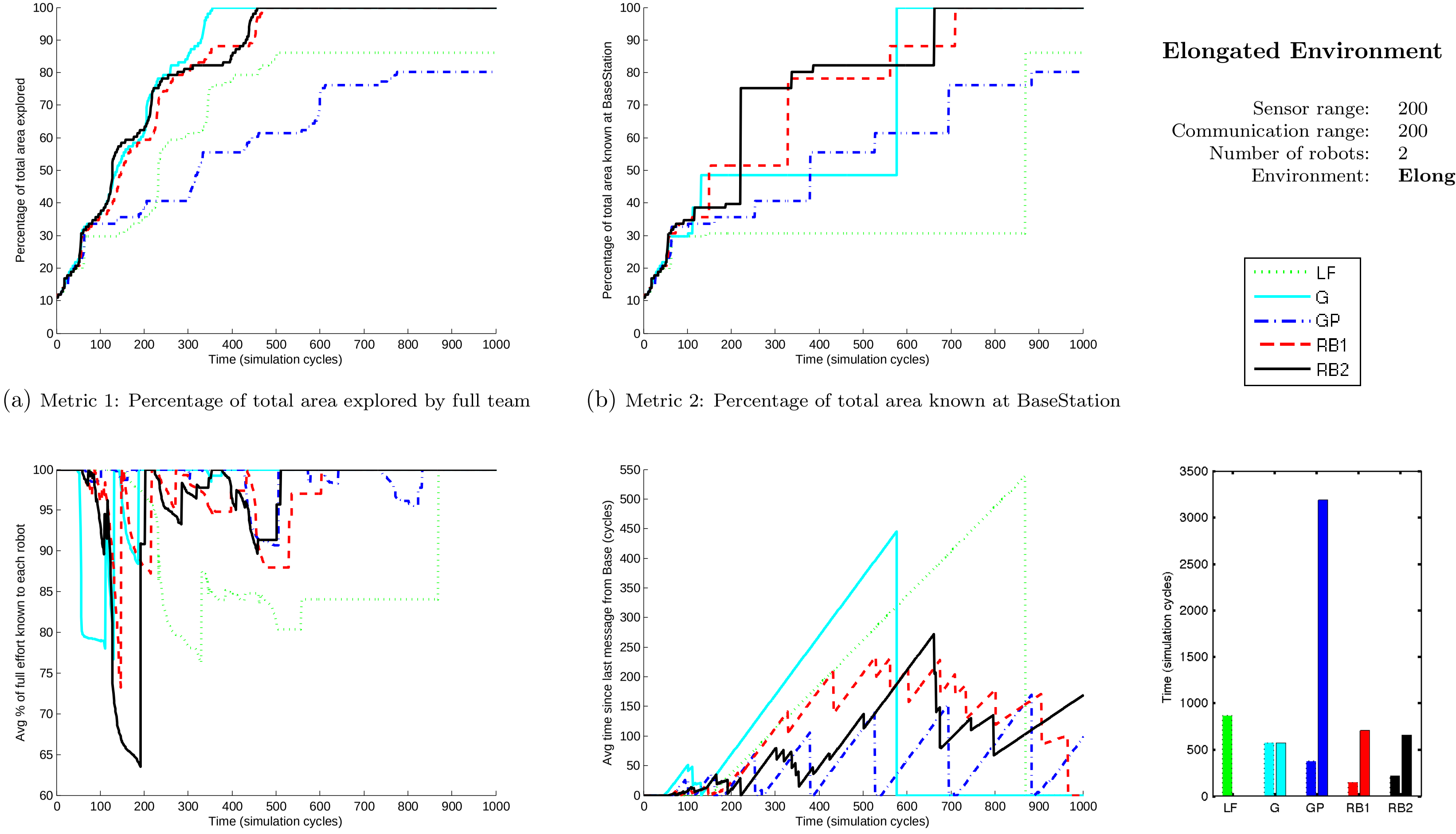
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.15: Results in an indoor environment



Elongated Environment

Sensor range:	200
Communication range:	200
Number of robots:	2
Environment:	**Elongated**

(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.16: Results in an elongated environment

5.9 Some Specific Scenarios

In this section, four scenarios are chosen that are each meant to represent possible scenarios in the real world. Whereas previously the parameters were only varied and examined one at a time, here for each case all parameters are chosen to simulate the given scenario as closely as possible. Naturally the sets of parameters for each case are hugely hypothetical, but nevertheless this remains a useful exercise for examining the relative performances of the different exploration algorithms under very different conditions.

Reconnaissance by UAVs

As discussed in Chapter 2, the use of unmanned aerial vehicles (UAVs) is becoming more and more widespread, with possible applications including tracking, surveillance, and search and rescue. Here, a reconnaissance scenario is examined: consider the case of a hiker having been lost in the mountains, or a ship being lost at sea. A team of UAVs could be used to cover the area, searching for the target (this task has much overlap with the "Coverage" problem discussed in section 2.2.8).

Using again the empty room as an environment, if this 800×600 pixel map represents an 8×6 kilometre area, then each grid cell represents $10\text{m}\times10\text{m}$. A UAV searching for a target will likely use GPS for localisation (no range scanner), but would have a field of view allowing it to cover say a radius of 50m, i.e. a sensor range of 5 cells. Outdoor wireless communication depends significantly on the hardware used, but a range of 500m is not unreasonable to assume, i.e. a communication range of 50 cells.

Results for each of the five performance metrics in an empty space using a team of 6 robots, a sensing range of 5 and a communication range of 50 are presented in Fig. 5.20.

As expected, the results in this scenario are similar to the performance of the various algorithms in an Open scenario (discussed in the previous section): G exploration is by far the fastest, although the relay to BaseStation is poor until some robots wander in range by chance. GP exploration performs much better than in previous tests, outperforming RB2, and performing similarly well to RB1. This is due to the fact that while GP robots will spend much time moving back and forth to the BaseStation, 6 exploring robots will cover new area much faster than 3 exploring robots; *i.e.*, the time spent relaying is made up for by the size of the team.

Role-Based Exploration does provide a good speed of exploration and regular updates to the BaseStation, but the advantages of Role-Based Exploration over other approaches are not nearly as evident in this scenario as in some other ones. Coverage of open space is a well studied problem and it is likely that better methods exist elsewhere.

Inspection of a Building

As discussed in section 2.1.3, the potential for robots to be used for inspection of hazardous areas is immense. This scenario examines the use of a team of robots to inspect a floor of a building. Real-world situations to which this applies could involve searching for people or other information in a building that might contain gas leaks, radiation, or risk of secondary structural collapse (*e.g.* after an earthquake).

As an environment, the floor plan of of the first floor of the CSU Stanislaus library is used[2] (Fig. 5.17). This environment contains many small rooms, long hallways, and loops (mostly large). An exact scale of the map could not be found, but it is estimated that the maximum width is approximately 120m. Therefore, in the 800 × 600 pixel map, each grid cell represents 0.2m×0.2m. Laser range finders today can have a variety of ranges, but an indoor range of 30m, *i.e.* 150 cells, seems realistic. Indoors,

[2] This data set was obtained from the Robotics Data Set Repository (Radish) [67]. Thanks go to Ashley Tews for providing this data.

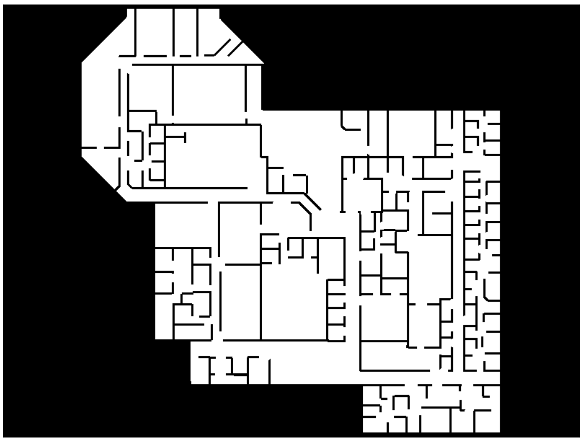


Figure 5.17: Floor plan of the CSU Stanislaus library used for Scenario 2

communication between robots can be assumed to be possible within 100m, *i.e.* 500 cells.

Results for each of the five performance metrics in the CSU Stanislaus library using a team of 6 robots, a sensing range of 150 and a communication range of 500 are presented in Fig. 5.21.

As usual, G exploration leads to the fastest total exploration, followed by the RB approaches. Uncharacteristically, G exploration even performs better regarding the relay of information to the BaseStation, though only after approximately 600 cycles. The reason for this is that the Stanislaus library can be divided into two significant chunks, as one continuous wall separates the environment cleanly in two. The robots enter the lobby area, which is the big rectangular space near the bottom. As some of the greedy robots complete exploration of the right half of the environment (containing many small rooms), their only remaining frontiers are in the other half of the environment. They must pass by the BaseStation on the way to these remaining frontiers, leading to a large jump in knowledge gain at the BaseStation.

The RB approaches do have benefits over G exploration regarding information sharing and responsiveness though. For both of these metrics both RB approaches perform similar to one another, and significantly better than G.

LF displays its usual characteristics of perfect information sharing and responsiveness, but at a large cost: only a small part of the environment is explored, and it is explored slowly.

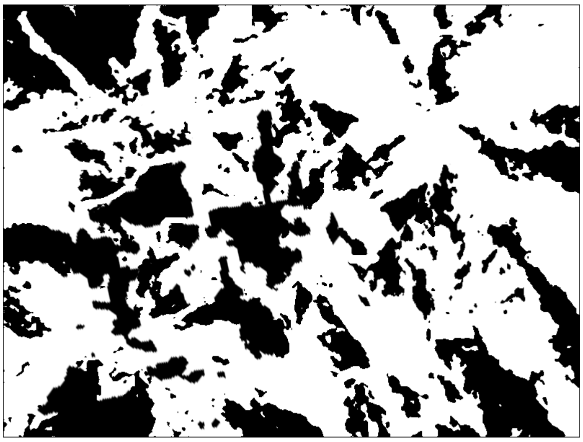


Figure 5.18: A highly cluttered environment used for Scenario 3.

Search and Rescue in a Complex Environment

Section 2.1.4 has already discussed the great potential for robots to be used in search and rescue situations, *e.g.* for entry into environments after landslides or earthquakes to find people. Such environments would likely contain narrow passages, small cavities, and many obstacles.

This scenario aims to simulate such a situation. It uses a highly cluttered environment with many fractured obstacles, many small loops and many dead ends (Fig. 5.18). A search and rescue robot might explore an environment of 40m × 30m, so using an 800×600 map each cell corresponds to 5cm×5cm. The robot's range scanner and communication system might be inhibited by dust, so ranges of 20m (*i.e.* 400 cells) are used for each.

Results for each of the five performance metrics in a cluttered environment using a team of 2 robots, a sensing range of 400 and a communication range of 400 are presented in Fig. 5.22.

Somewhat surprisingly, both RB approaches lead to faster total exploration than G. This is an unexpected result, given that two robots are always exploring in G, whereas only one is exploring in each of RB1 and RB2. It is possible that environmental factors contributed to this result (since some spaces in the environment are more open and quicker to explore than others). However, it is more likely that the G robots performed much redundant exploration, re-exploring areas that had already been discovered by the teammate previously.

The performance of RB2 is significantly better than the performance of RB1, which is in line with the results found for cluttered environments in section 5.8: many small obstacles lead to more role swaps, which lead to increased efficiency in exploration.

LF and GP are slow, but LF demonstrates perfect information sharing and responsiveness as always. The responsiveness of the RB approaches seems worse, relative to their performances in other scenarios, but that is to be expected when they are covering large areas in highly cluttered environments.

Remote Exploration of a Mine

As discussed in section 2.1.5, there is significant potential for using robots to inspect, map or otherwise examine mines. In this scenario, a team of 4 robots must enter and fully explore a mine.

As an environment, the floorplan of the Bruceton Research mine (as mapped in [145]) was chosen (Fig. 5.19). This contains many long, narrow hallways, and is in many respects similar to the elongated environments previously discussed. The real map is 250m by 180m, so when reduced to an 800 × 600 map, each grid cell is therefore equivalent to 0.3m×0.3m. Robots exploring such a mine would likely have long range laser scanners, *i.e.* a sensing range of 30m (or 100 cells). While there may be obstacles interfering with communication (overhead beams, equipment etc), a communication range of 60m seems reasonable to assume (*i.e.* 200 cells).

Results for each of the five performance metrics in the Bruceton Research mine scenario using a team of 4 robots, a sensing range of 100, and a communication range of 200 are presented in Fig. 5.23.

G exploration is again the fastest, but only just. Both role-based approaches are almost as fast, and perform much better relaying to the BaseStation. The LF approach explores fast initially, but as before has a limit to the range it can explore. GP exploration is very slow.

Relay of information to the BaseStation is performed the best by the RB approaches, by far. The G robots only re-enter the BaseStation's range after more than 11,000 cycles.

In spite of their fast exploration, the RB approaches lead to fairly good information sharing and responsiveness. Both LF and GP do even better, but the robots are not travelling nearly as far for either of them.

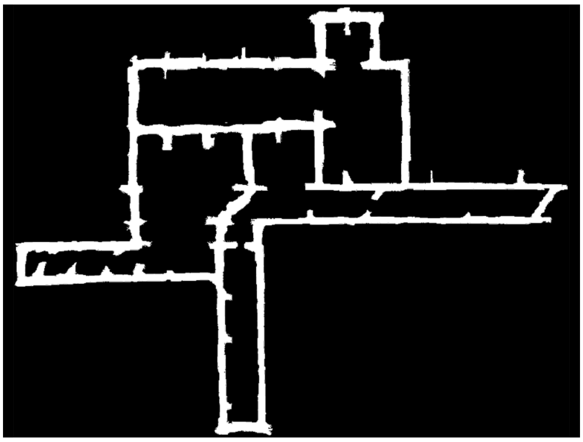
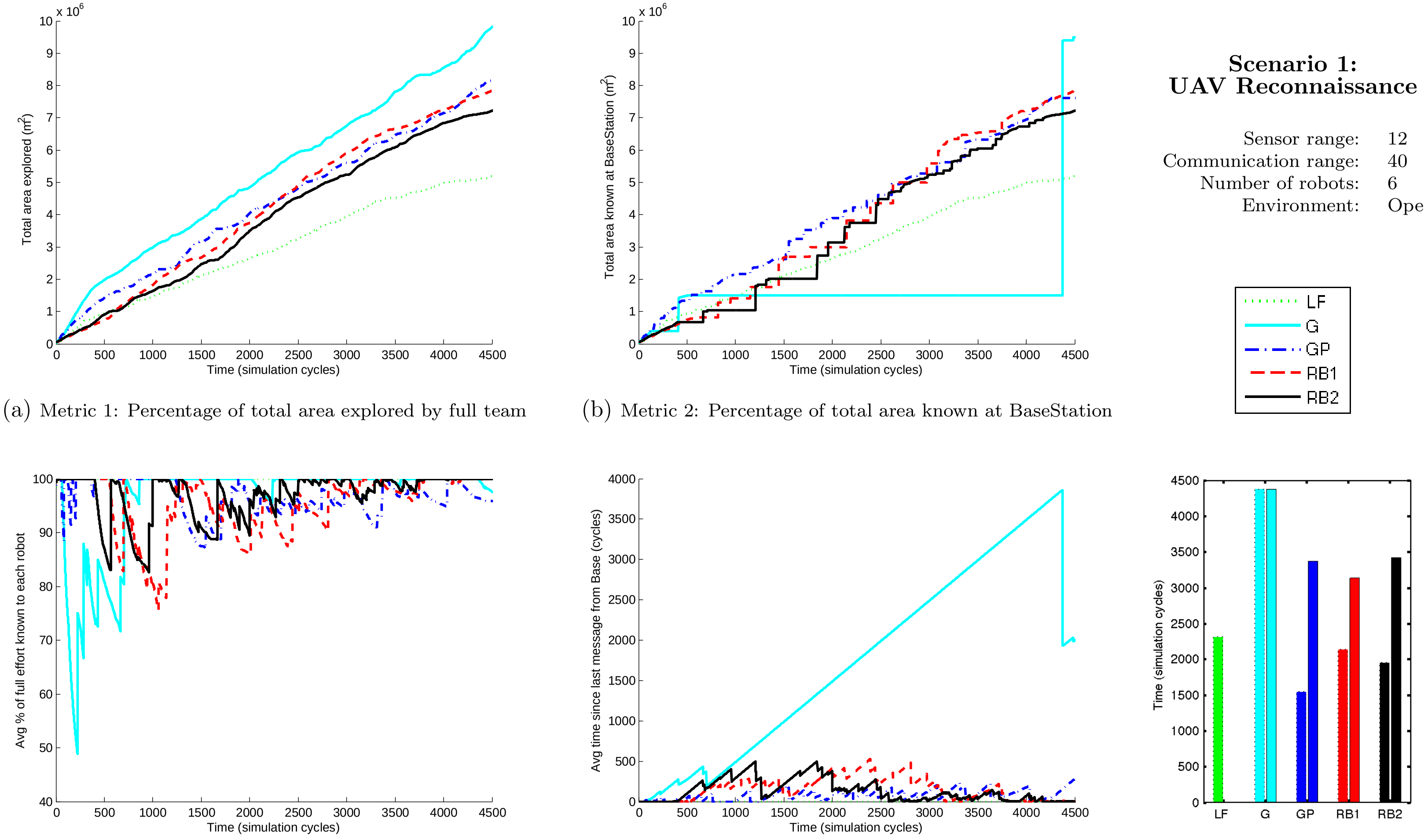


Figure 5.19: Floor plan of the Bruceton Research mine (as mapped in [145]), used for Scenario 4



**Scenario 1:
UAV Reconnaissance**

Sensor range:	12
Communication range:	40
Number of robots:	6
Environment:	Open space

(a) Metric 1: Percentage of total area explored by full team

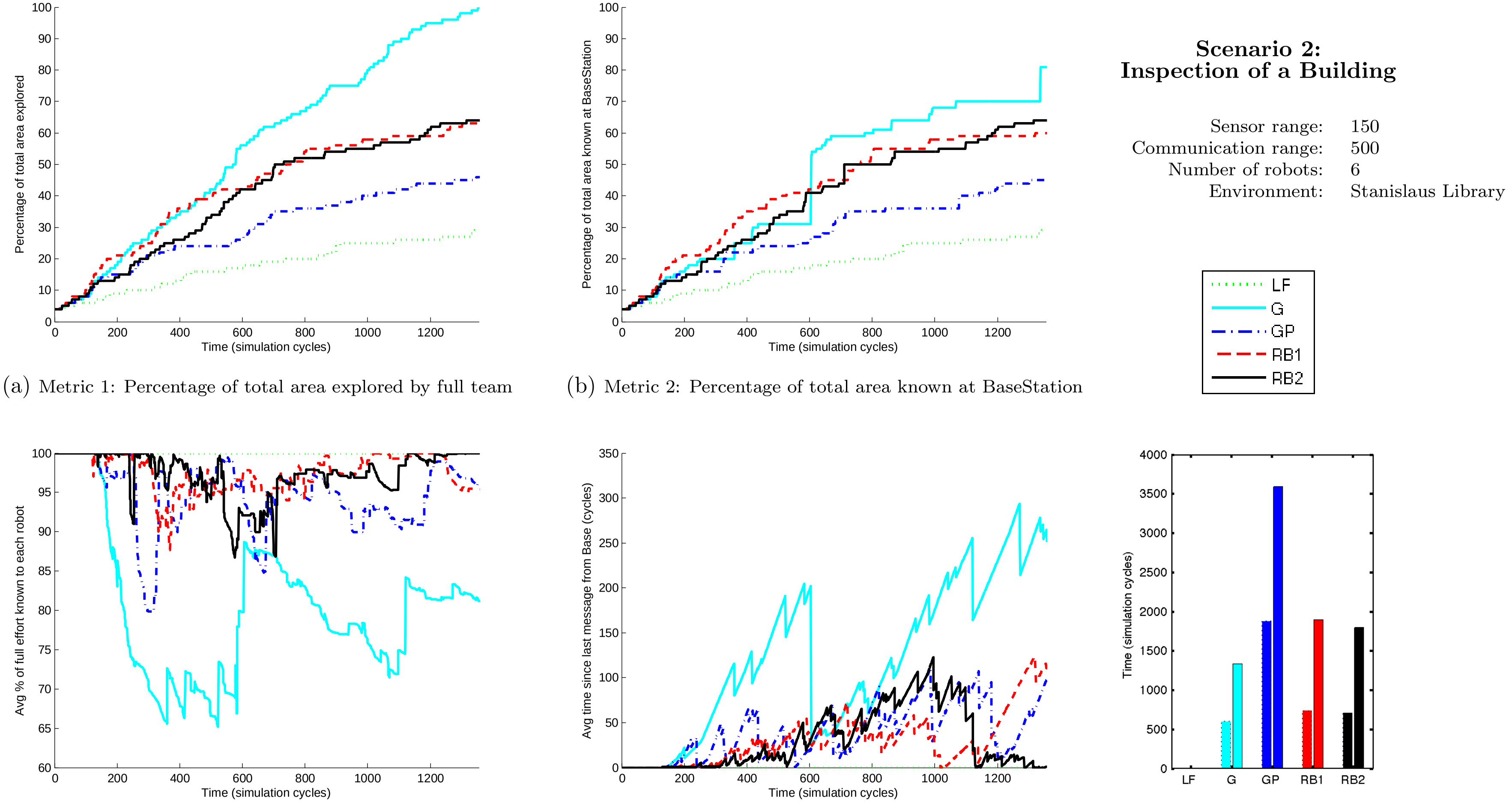
(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 30,000m^2 (dotted) / 60,000m^2 (lined) of environment known at BaseStation

Figure 5.20: Results in Scenario 1: UAV Reconnaissance



(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

Scenario 2:
Inspection of a Building

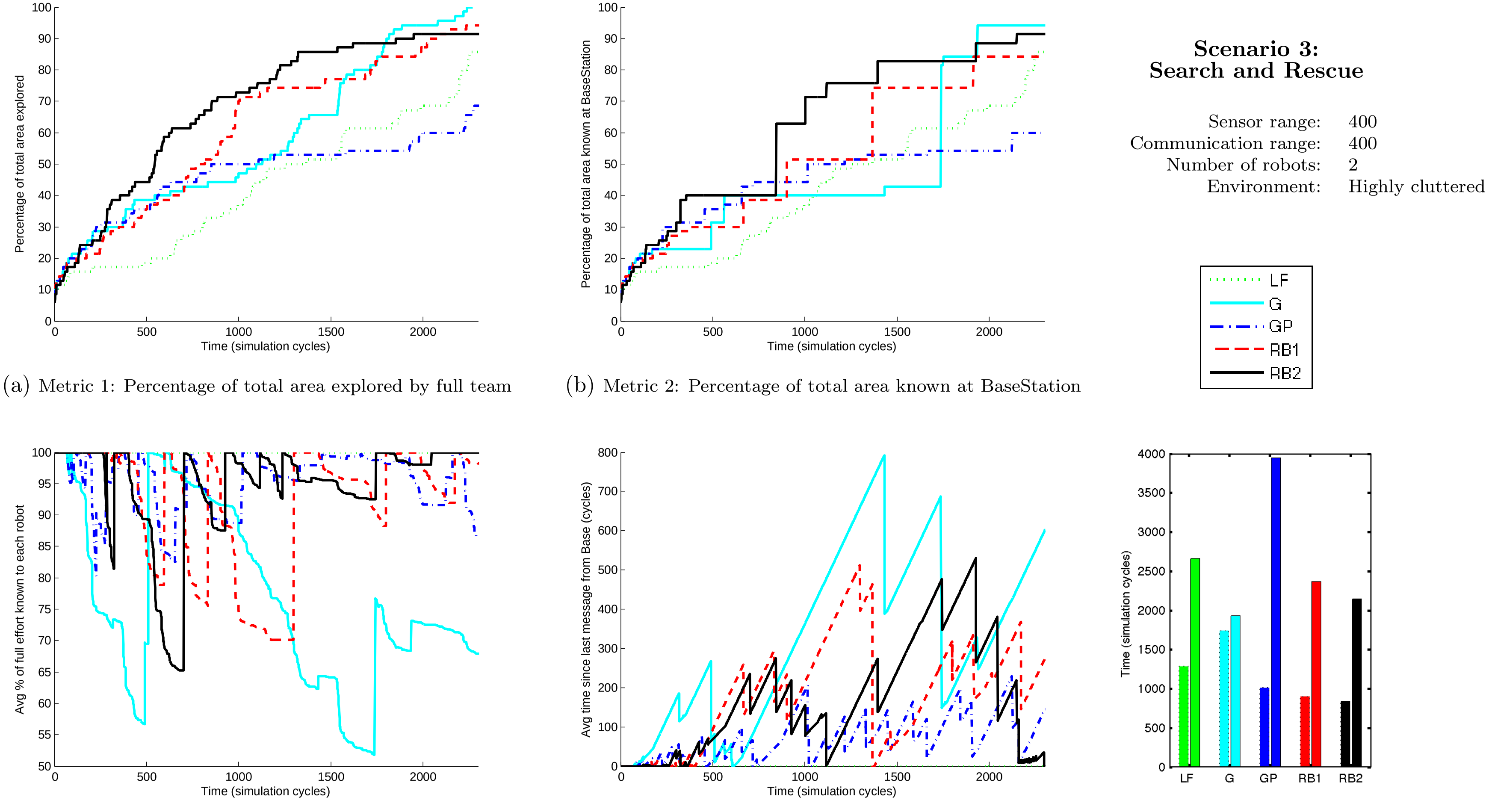
Sensor range:	150
Communication range:	500
Number of robots:	6
Environment:	Stanislaus Library

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 75% (lined) of environment known at BaseStation

Figure 5.21: Results in Scenario 2: Inspection of a Building



(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

Scenario 3:
Search and Rescue

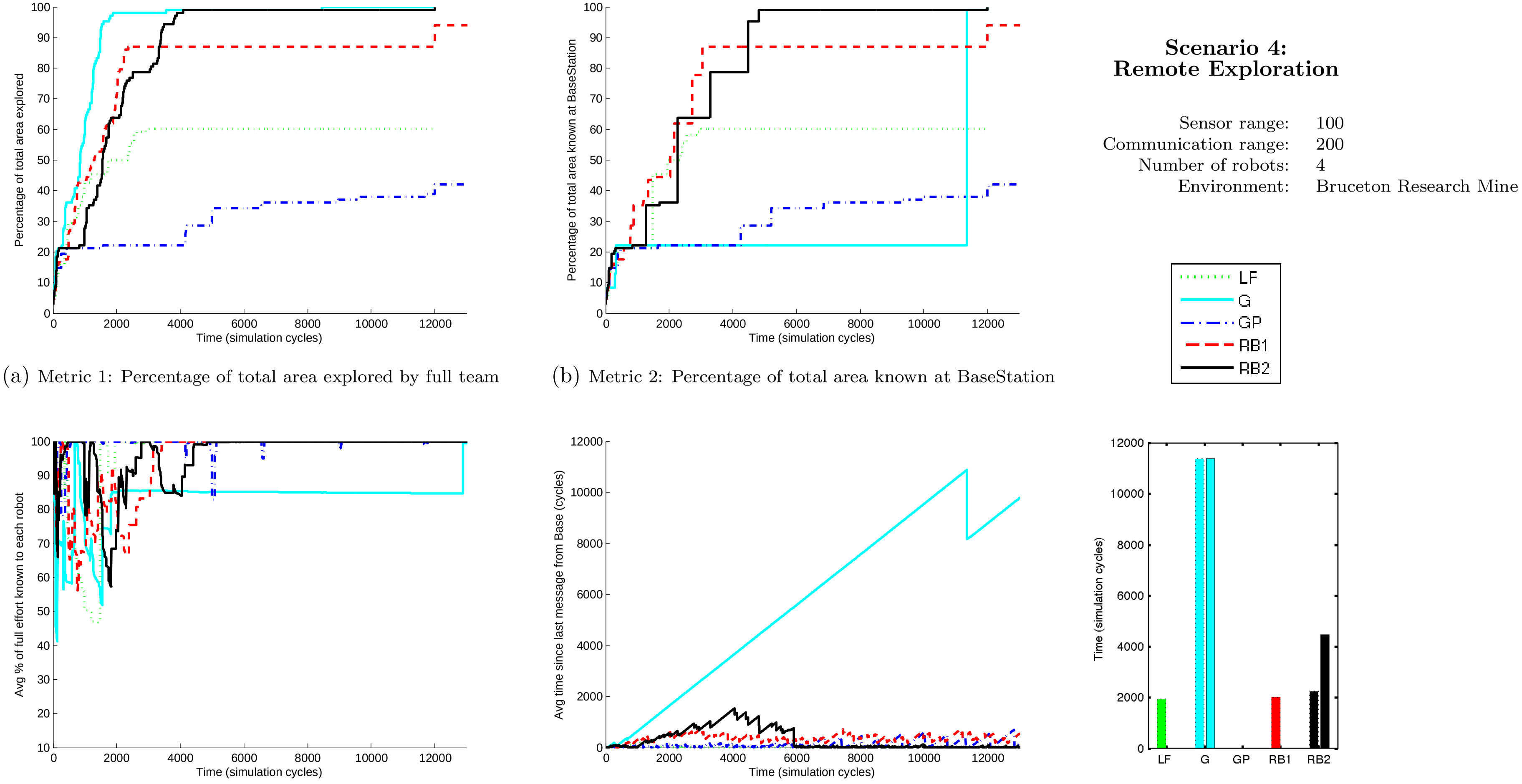
Sensor range:	400
Communication range:	400
Number of robots:	2
Environment:	Highly cluttered

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 90% (lined) of environment known at BaseStation

Figure 5.22: Results in Scenario 3: Search and Rescue in a Complex Environment



Scenario 4:
Remote Exploration

Sensor range:	100
Communication range:	200
Number of robots:	4
Environment:	Bruceton Research Mine

(a) Metric 1: Percentage of total area explored by full team

(b) Metric 2: Percentage of total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

(e) Metric 5: Time until 50% (dotted) / 100% (lined) of environment known at BaseStation

Figure 5.23: Results in Scenario 4: Remote Exploration of the Bruceton Research mine

5.10 Summary

This section summarises the results that were observed in the large number of experiments presented in this chapter.

Speed of total exploration

G exploration leads to the fastest discovery of new information in almost all cases. The RB methods approach the speed of G only when the sensor or communication ranges are very large relative to the size of the environment, when there are many robots exploring, or when environments contain long hallways (or fewer frontiers than robots in the team). However, it must be remembered that this speed of exploration is not necessarily of use in many situations, as it might not reach human responders. GP exploration typically lags behind the RB approaches, and by far the slowest approach is LF exploration.

Relay of information to BaseStation

LF exploration demonstrates perfect relaying of new information to the BaseStation (since all team members are connected), but being quite slow still lags behind the other approaches in terms of the amount of information brought to the BaseStation per unit of time. The RB approaches demonstrate the most regular, reliable relaying of new information to the BaseStation in almost all cases. There are some situations where G exploration performs similarly or better (*e.g.* as in Scenario 2, if the shape of the environment forces robots to pass by the BaseStation) and there are some situations where GP exploration is almost as fast or faster (*e.g.* in open environments). However, there are many more situations where the advantages of Role-Based Exploration in this respect are tremendous, for example when communication range is small relative to the environment, when there are more and more robots in the team, and particularly in obstacle-filled or elongated environments.

Information sharing within the team

The RB approaches also provide major advantages regarding information sharing within the team. In this regard they always perform much better than G exploration. In many experiments, GP exploration demonstrated better information sharing than the RB approaches, but it must be remembered that most of the time the RB approaches had covered more area, and thus had longer paths and longer intervals outside of one another's ranges. LF exploration of course demonstrates perfect information sharing, since all members are always connected.

Responsiveness to BaseStation commands

If an operator had to make sudden changes to the exploration effort, such as designating parts of the environment as being of greater interest or pulling the whole team out, then again the RB approaches would be of much greater use than G exploration. Responsiveness is better for the RB approaches over G in almost all cases, and generally outperforms GP as well when taking size of explored environment into consideration. This is particularly true when larger numbers of robots are involved or environments have many obstacles. Again, LF exploration shows the greatest responsiveness of all, which follows from the fact that the team is fully connected at all times.

Completeness

LF exploration only completely explores the environment in rare cases. G exploration typically leads to the fastest complete exploration. Each of RB1, RB2 and GP are complete, but the RB approaches typically require longer, and GP in particular can take a very long time indeed to finish exploring certain environments.

The advantages of the role swap

When examining the data from the experiments presented here, it may seem that the use of role swaps is not hugely significant. Indeed, in many cases there is no obviously discernible difference between RB1 and RB2, and either can seem to outperform the other depending on chance circumstances such as small random differences early in the exploration effort.

However, there are certain situations where the use of role swaps leads to important and worthwhile gains in efficiency: when there are many small, rather than large loops (resulting *e.g.* from many obstacles); when there are long dead ends; and when there are more and more robots in the team. This behaviour makes sense when one considers the original motivations for introducing the Role Swap Rule.

The Role Swap Rule is easy to implement, its added complexity comes at little additional computational cost, and it can be easily toggled on or off depending on the task at hand, so there is good reason to implement it in any case.

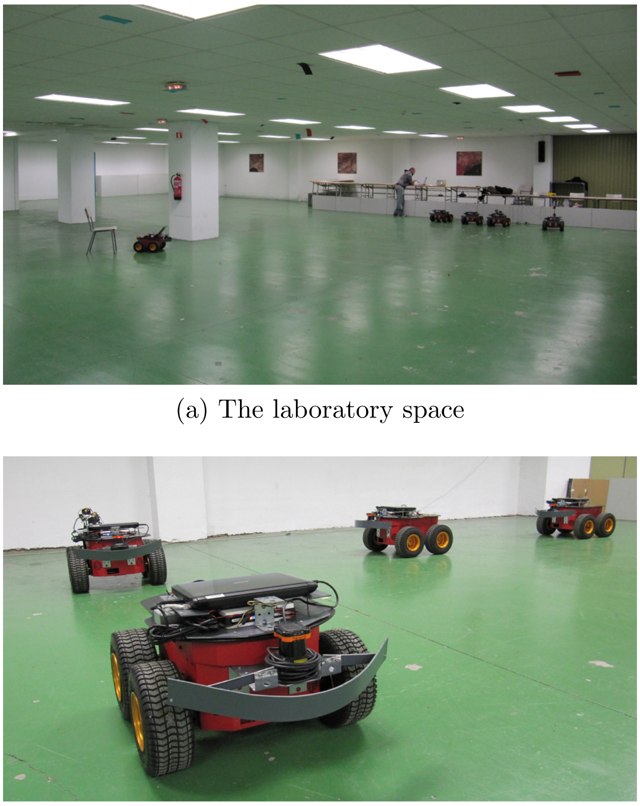
~

This concludes the chapter on simulation results. In the next chapter, Role-Based Exploration is demonstrated on a team of Pioneer robots, and the results are compared to the results found here.

Chapter 6

Implementation on a Real System

It is a common experience that results obtained in simulation do not always closely reflect reality. Unexpected problems often occur when algorithms are ported to real robots, and assumptions that were considered viable can turn out to be unrealistic. It was therefore considered necessary to implement Role-Based Exploration on a real multi-robot system. This chapter details the results of a series of tests and experiments performed using a team of P3AT robots at the University of Seville, Spain. Experiments were conducted in two phases: first, in a small, controlled environment (the laboratory space); and second, in a large, uncontrolled environment (the hallways of the university).



(a) The laboratory space

(b) The team of P3AT robots

Figure 6.1: The CONET Integrated Testbed

6.1 Laboratory and Robots

All experiments involving real robots were conducted in the Integrated Testbed of the University of Seville, Spain [70], under the auspices of the CONET project[1]. The Testbed is run in an open space measuring 23.9m by 20.3m (Fig. 6.1a). Numerous objects are available for creation of custom environments (desks, chairs, boxes, etc.).

The Testbed includes 5 *Pioneer 3-AT* robots, a skid-steered four-wheeled research platform (Fig. 6.1b). Each robot is mounted with a wireless a/b/g/n bridge, a *Hokuyo UTM-30LX* 2D laser range scanner having a range of 30m and resolution of 0.25 degrees at 25ms/scan, and an *Acer AspireOne 532h Netbook* (1.66GHz, 1GB Memory) running Ubuntu 10.04.

[1] CONET: Cooperating Objects Network of Excellence (INFSO-ICT-224053) funded by the European Commission under ICT, Framework 7.

To control the robot actuators and receive data from the laser scanner, each robot runs Player, a widely used open-source network server that includes support for a large variety of devices, robots and sensors [52]. Within Player there are a number of useful drivers that take care of common robotics tasks. These include for example the *vfh* (vector field histogram) driver for local obstacle avoidance, the *wavefront* driver for global path planning and motion commands, and the *amcl* (adaptive Monte Carlo localisation) driver for localisation of the robot.

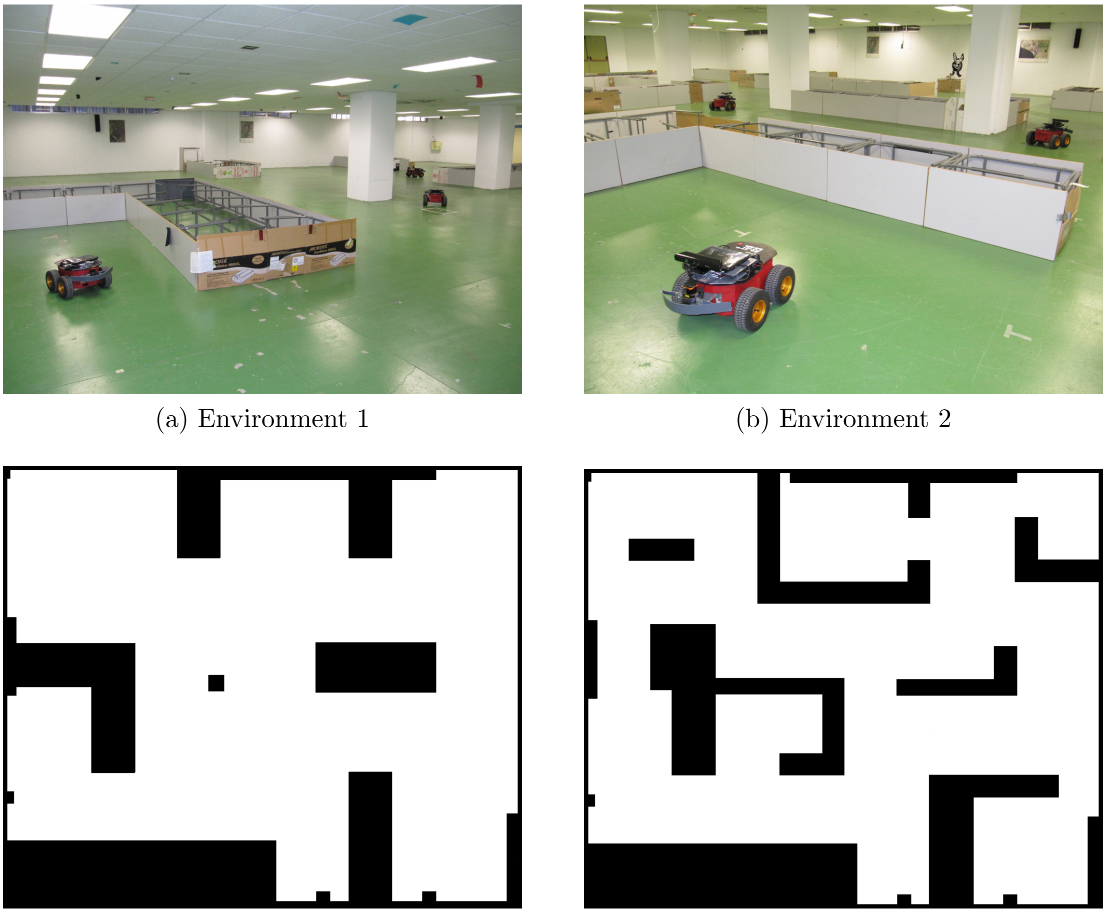
6.2 Experiments in a Small, Controlled Domain

The first phase of experiments was carried out solely in the Testbed, *i.e.* in the 23.9m by 20.3m space detailed above. This section describes the environments that were used, details the adjustments that had to be made to the mapping system in moving from simulation to reality, explains how communication was artificially limited, and presents the results of the first experimental phase.

6.2.1 Environments and Localisation

Two environments were created in the laboratory space using desks and cardboard. The first was fairly simple and open, with ample space between walls for robots to navigate (this environment would fall into the category of "Open" environments explored in section 5.8). The second contained several tight spaces and room-like structures (this environment would fall somewhere between the "Open" and "Indoor" environment types). Pictures and floorplans of these two environments are shown in Fig. 6.2.

In anticipation of the experiments on real robots, a number of open-source simulataneous localisation and mapping (SLAM) algorithms were tested in Stage, the



(a) Environment 1

(b) Environment 2

(c) Floorplan of Environment 1

(d) Floorplan of Environment 2

Figure 6.2: Environments 1 and 2, and their floorplans

simulation framework tied to Player. Player's own *amcl* driver[2] turned out to be the most robust; other localisation systems were equally or more accurate, but took a long time to compute and would occasionally become irrecoverably lost. The *amcl* driver, even if temporarily inaccurate, always recovered and provided highly reliable localisation information.

The *amcl* driver requires an *a priori* map of the environment to probabilistically determine the robot's location using data from odometry and laser. In applications such as robotic search-and-rescue, it is unlikely that a map of the environment would

[2] The *amcl* driver was created by Andrew Howard and is based on an Adaptive Monte-Carlo Localisation algorithm originally proposed by Dieter Fox. A more thorough description can be found on the Player website: `http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group__driver__amcl.html`

be available prior to the robots' entry. However, the purpose of these experiments was to compare exploration algorithms, not to develop a SLAM system. SLAM is a well studied field of research and numerous solutions have been found for the localisation and mapping problem. Therefore, use of the *amcl* driver for these experiments was considered an acceptable solution, since it ensured reliable mapping and localisation, and repeatability of experiments.

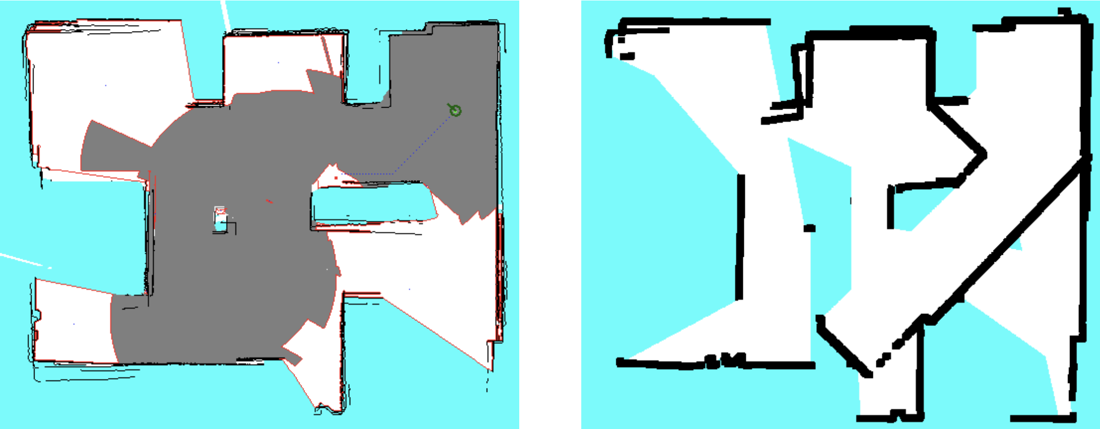
6.2.2 Adjustments to Mapping

In moving from simulation to reality, several adjustments had to be made to the mapping system.

First, while the *amcl* localisation system is quite good, the pose it returns is not matched to laser data, so synchronisation of pose and laser data is crucial. This was conducted using a "catch-up" system: if the most recent pose data contains a newer timestamp than the most recent laser data, the laser data is discarded and new data is popped off the queue until data within a minimum time threshold is obtained. If the laser data is more recent, then vice versa. Synchronisation in this manner led to significantly more accurate mapping.

Second, even after synchronisation the pose returned by the *amcl* driver can occasionally be slightly off, leading to multiple copies of walls appearing close to one another (Fig. 6.3a). This meant that many small, fractured frontier polygons were found, which led to a more time-consuming robot-frontier assignment process. A simple solution was to thicken all walls in the map by several cells in the occupancy grid (a step that can optionally be undone later).

Third, there were situations where a single particularly poor pose estimate (or sensing of a teammate) led to incorrect obstacle cells in the grid (Fig. 6.3b). Having incorrect obstacle cells means that frontiers may not be calculated correctly, or paths may be planned poorly. This problem was solved with the introduction of a first-in-



(a) Fractured frontiers

(b) Shifted map patches

Figure 6.3: Two reasons for poor initial maps

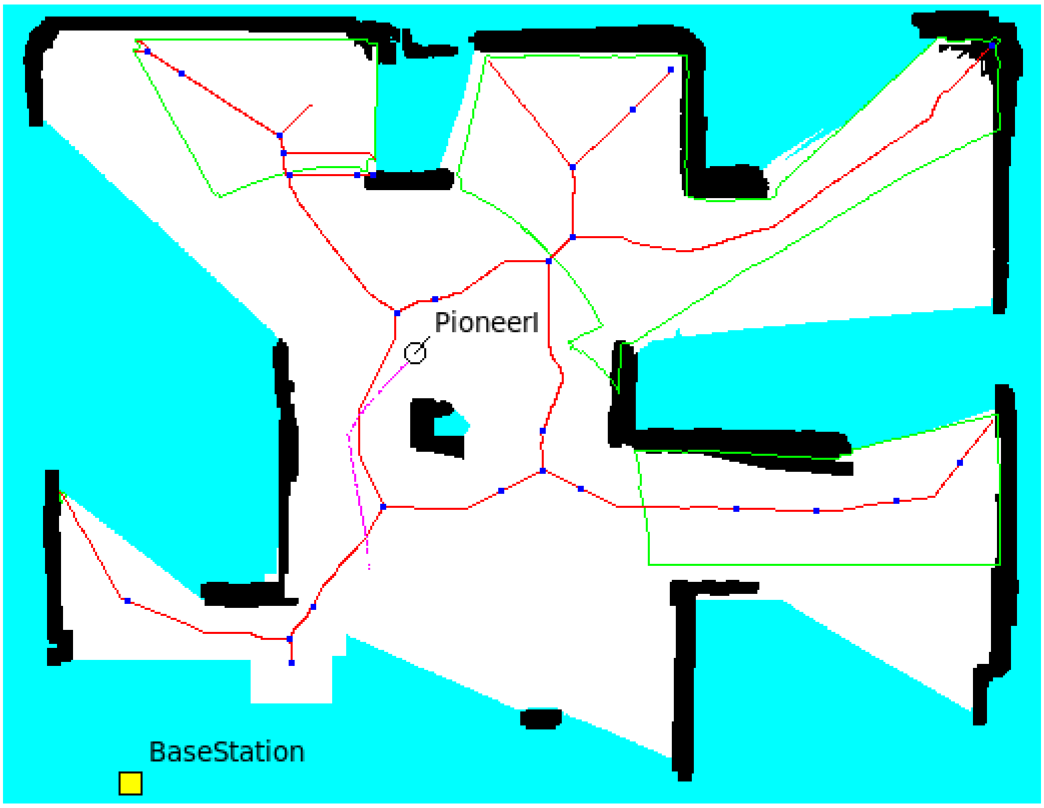


Figure 6.4: A screenshot from an early run that tested mapping in the Testbed. One robot (Pioneer I) is exploring Environment 1. Frontier polygons are outlined in green; the skeleton is red; the robot's path is purple; and potential rendezvous points are blue.

first-out queue for each grid cell. Every newly recorded measurement for a given cell (free space or obstacle) is added to this queue, while the oldest element is removed. If the majority of the most recent measurements indicate free space, the cell is labeled *free*; if they indicate obstacle, it is labeled *obstacle*.

There are certainly more sophisticated solutions to this problem, such as map merging (for incorrectly mapped patches of the map) and information filters (for determining exact locations of teammates). However, the experiments at the testbed had to be completed in a constrained time period, and the queue-per-cell method is very simple to implement, fast (a full update of a 600×600 map takes about 200ms on the Acer Netbooks that are on the Pioneers), and turned out to be highly effective in erasing spurious errors in the map.

These adjustments led to a robust, reliable mapping system that allowed each robot to cleanly map the environment, share its maps with its teammates, and compute frontier polygons, skeletons, rendezvous points and paths (Fig. 6.4).

6.2.3 Communication

Communication between robots and the BaseStation was performed over the wireless network using TCP/IP, via YARP[3]. Access to the wireless network is ubiquitous within the testbed. However, since the goal of the experiments was to examine the performance of exploration algorithms under *limited* communication, it was necessary to artificially limit communication between all robots. This was performed using a central "Dispatch", a program that opened send and receive channels to each robot and the BaseStation, and only forwarded messages amongst robots or between robots and the BaseStation when they were within a designated range of one another.

[3] YARP: Yet Another Robot Platform. Available at `http://eris.liralab.it/yarp/`. YARP is an open-source set of libraries, protocols and tools for communication that greatly simplifies the communication process through TCP, UDP and Shared Memory. YARP uses a nameserver to connect different output and input ports which are capable of interchanging formatted data (integers, floats, strings, arrays, lists, etc.) or binary data.

For most of the experiments, “in range” was defined as a radius of 8m around each robot; this was approximately one third the width of the lab, and meant that extensive areas of the environment remained beyond communication range.

6.2.4 Results and Performance Metrics

Since there was limited time available for the experiments in the Testbed, and since the porting of the mapping and exploration systems from simulation to reality was fairly time-consuming, the experiments conducted in the Testbed were not nearly as extensive and exhaustive as those conducted in simulation. In the laboratory environments discussed in this section, only the Greedy (G), Role-Based without role swaps (RB1) and Role-Based with role swaps (RB2) algorithms were implemented (more detailed descriptions of these algorithms were presented in section 5.2).

The most extensive comparison was conducted between the G and RB1 approaches. Each of these was implemented on a team of either 2 or 4 robots, in each of the two environments. For each scenario {RB1 / G} * {team of 2 / team of 4} * {Environment 1 / Environment 2}, multiple runs were conducted. Due to time constraints, role swaps and the resulting RB2 exploration were only implemented near the end of this phase of experiments. By this point Environment 1 had been disassembled so the results for RB2 exploration are only available in Environment 2.

Here, the results from two sets of experiments are presented; these are representative of a larger number of runs in both environments with different team sizes.

Four robots exploring Environment 1

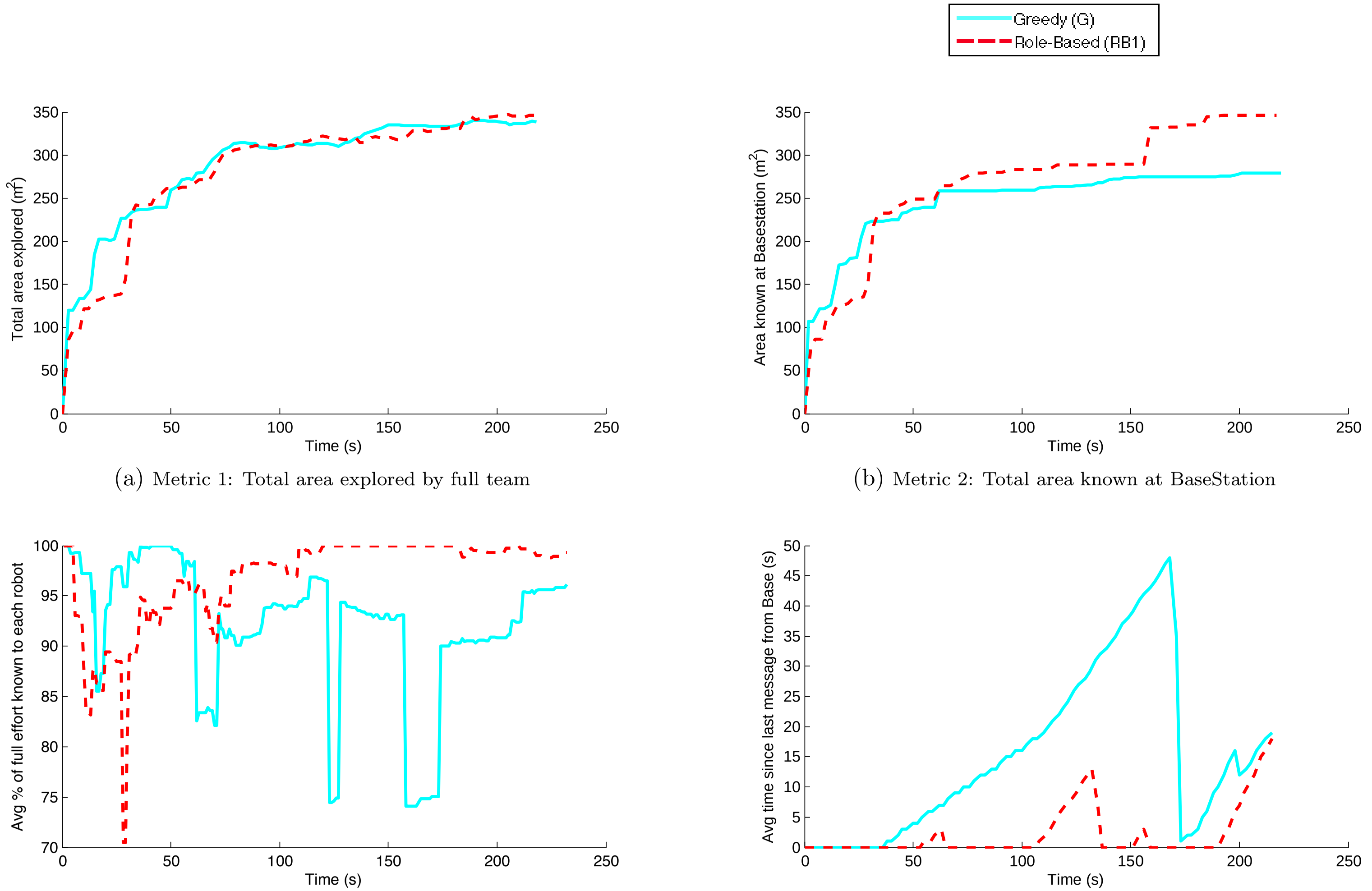
The first set of results involves a team of 4 robots exploring Environment 1; results for the performance metrics (as discussed in section 3.3) of these runs are presented in Fig. 6.5.

Initially, Greedy exploration leads to faster exploration, but Role-Based exploration soon catches up and performs as well or better (metric 1). The reasons for this are simple: in Greedy exploration, twice as many robots are exploring as in Role-Based exploration, so initial progress is faster. The Greedy teammates however do not make an effort to coordinate their exploration (unless wandering within range of one another by chance), and soon duplicate one another's efforts, re-exploring parts of the environment.

While the robots are in range of the BaseStation, knowledge at the BaseStation mirrors total team knowledge (metric 2). However, soon the relaying of RB1 proves to be better than the relaying of the G approach. In Role-Based exploration, regular updates from the relaying robots mean that there are frequent increases in the BaseStation's knowledge. In Greedy exploration, updates are infrequent, happening only when a robot wanders back within range by chance.

Overall, the sharing of information within the team is performed better by RB1 (metric 3). This is not the case initially (as explorers are discovering much new space while relays are returning to the BaseStation for the first time) but becomes more and more true as the effort goes on. Information is only shared by the Greedy robots when they meet one another by chance.

Finally, the responsiveness of the team is better for RB1 throughout the entire effort (metric 4). Messages from the BaseStation would be received by all team members fairly quickly in the Role-Based approach, reaching even the explorers fairly soon via the relays. In the Greedy approach, messages from the BaseStation only reach the robots when they happen to wander in range.



(a) Metric 1: Total area explored by full team

(b) Metric 2: Total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

Figure 6.5: Performance metrics for exploration by a team of 4 robots in Environment 1

Two robots exploring Environment 2

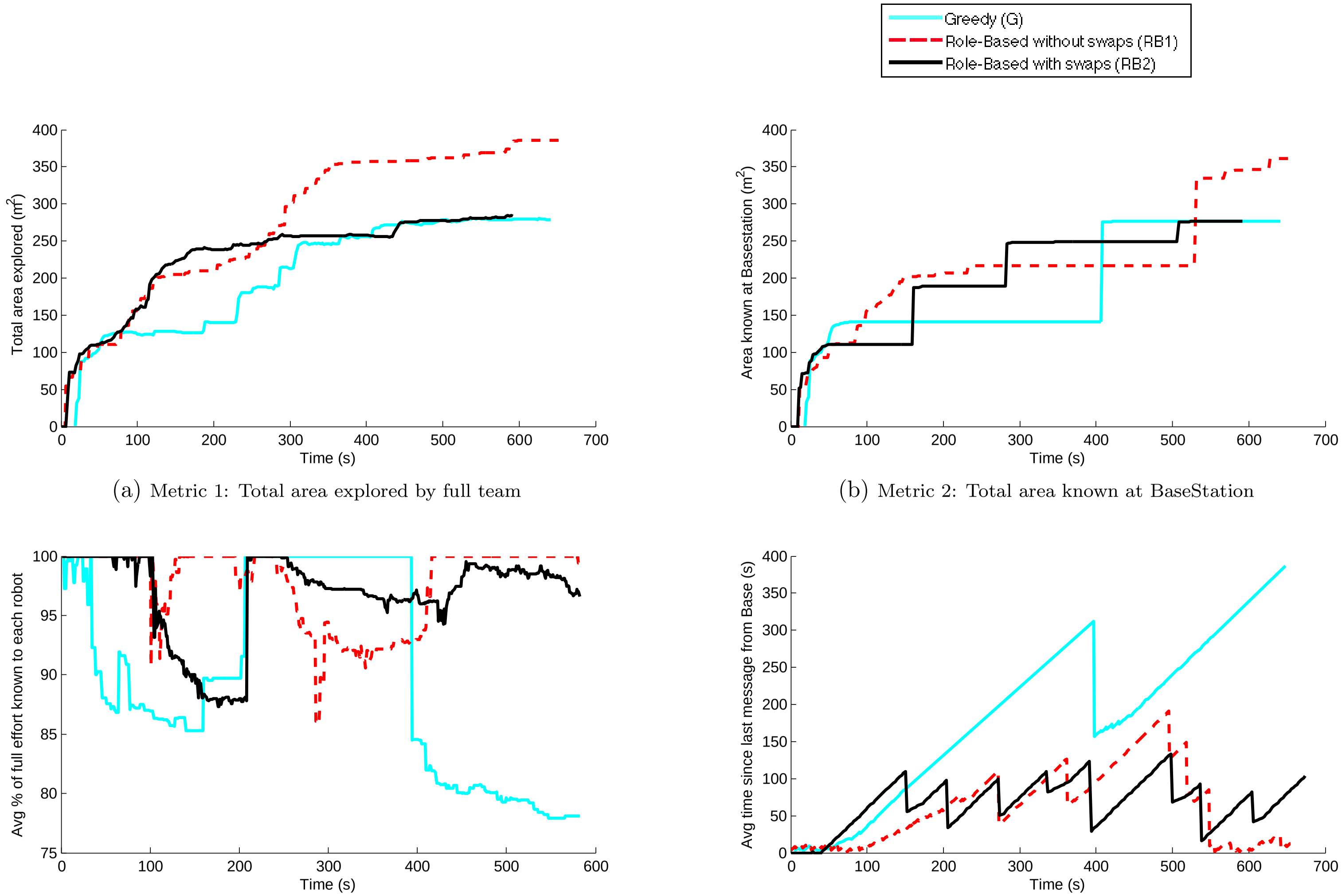
The second set of results involves a team of 2 robots exploring Environment 2; results for the performance metrics of these runs are presented in Fig. 6.6. Unlike the previous set of runs, this one also includes results for Role-Based Exploration *with* role swaps (RB2).

Role-Based Exploration leads to faster total exploration of the environment (metric 1). This came as a surprise and was not consistent with previous results, but re-running the experiment from log data revealed the reason: early on in the Greedy approach, shortly after the two robots lost contact with one another, one of the two robots chose to explore along a route that the first had already visited. Thus, the second robot duplicated some of the first robot's work. This is a typical pitfall of Greedy exploration, and one of the motivations for Role-Based exploration in the first place: for efficient exploration, robots should share information with one another as much as possible.

Relay of new information to the BaseStation (metric 2) is once again better for both role-based approaches; in this regard, swapping of roles within the team leads to slightly more regular updates.

Information is shared fairly well by both robots for all approaches (metric 3), with the RB1 algorithm showing best performance overall.

Finally, responsiveness (metric 4) is once again worst for the Greedy approach, with RB2 showing slightly better responsiveness than RB1. This is in line with RB2's performance for metric 2; more regular contact with the BaseStation should lead to parallel improvements in metrics 2 and 4.



(a) Metric 1: Total area explored by full team

(b) Metric 2: Total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

Figure 6.6: Performance metrics for exploration by a team of 2 robots in Environment 2

A series of screenshots demonstrating the performance of RB2 and the application of the Role Swap Rule are presented in Fig. 6.7.

Initially, Pioneer IV (P_{IV}) is an explorer and sets out to explore; Pioneer III (P_{III}) is a relay and follows Pioneer IV.

After 49 seconds, P_{III}'s connection to the BaseStation breaks. It follows P_{IV} for a short time before turning around to return new knowledge to the BaseStation.

After 153 seconds, P_{III} has reached the BaseStation, and turns to rendezvous with P_{IV}. P_{IV} has been exploring, but realises it is now time to turn around to meet P_{III}.

After 222 seconds, P_{IV} and P_{III} meet for their first rendezvous. P_{IV} wants to explore the small room near the BaseStation next. However, P_{III} is closer, so the two swap roles and P_{III} becomes an explorer, exploring the room.

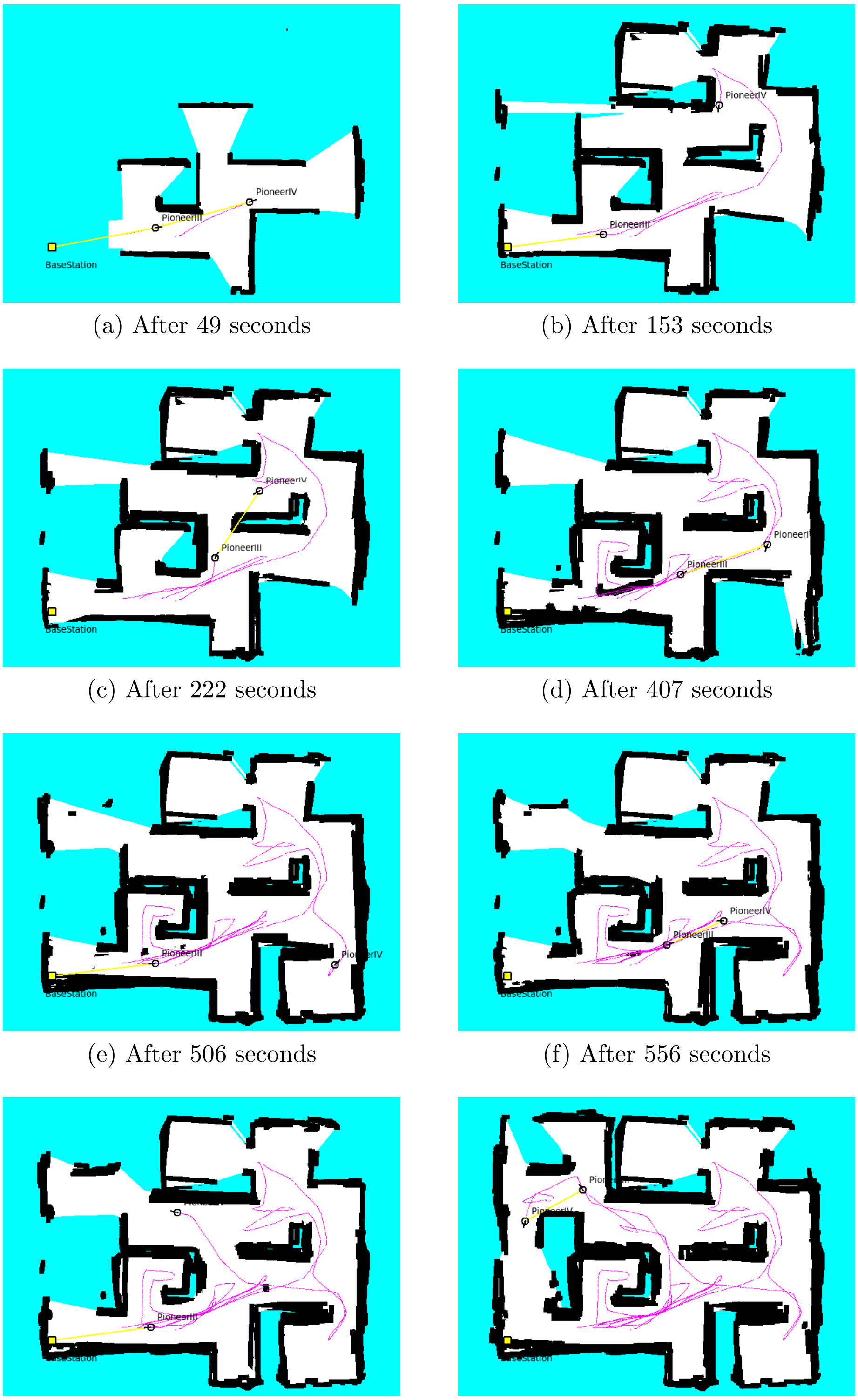
After 407 seconds P_{III} and P_{IV} meet again for a second rendezvous. P_{IV} is now closer to the next frontier of greatest interest, namely the room at the bottom right. P_{IV} becomes an explorer again, while P_{III} relays new information back to BaseStation.

After 506 seconds, P_{III} has relayed new information to the BaseStation, and P_{IV} has finished exploring the room.

After 556 seconds, they meet again for a third rendezvous. The next frontier of interest is at the top left of the environment; there is no benefit in switching roles so P_{III} remains a relay while P_{IV} remains an explorer.

After 617 seconds, P_{III} has relayed, and turns to follow P_{IV} into the unexplored part of the environment.

After 762 seconds, P_{IV} has explored much, but not all of the top left part of the environment. P_{III} reaches it. The shortest path to the BaseStation now known is down the hallway along the left hand side. P_{IV} is now closer to the BaseStation and becomes the relay, while P_{III} becomes an explorer and continues with exploration.



(a) After 49 seconds

(b) After 153 seconds

(c) After 222 seconds

(d) After 407 seconds

(e) After 506 seconds

(f) After 556 seconds

(g) After 617 seconds

(h) After 762 seconds

Figure 6.7: Screenshots demonstrating the performance of RB2 and the swapping of roles by two robots in Environment 2

6.3 Experiments in a Large, Uncontrolled Domain

The second phase of experiments was carried out in the hallways of the University. This is a large environment; some hallways are 100m in length (Fig. 6.8). It was not possible to use *a priori* maps for localisation, and communication did not need to be artificially constrained – there were significant communication challenges inherent in the environment as it was. This section details the changes made to mapping and communication in the move to such an uncontrolled domain, and details the results that were achieved.

6.3.1 Scan Matching

Whereas Player's *amcl* driver had proven highly useful in the first phase of experiments, this was no longer an option given that (i) no *a priori* map of the environment was available; and (ii) the point of the experiments was to use as realistic a domain as possible. Experiments with several existing mapping systems revealed that another Player driver, *mricp*[4], provided excellent maps using a standard scan-matching algorithm. The main drawback of the *mricp* driver is that it takes some time to compute subsequent positions, but this was circumvented by using odometry for localisation when new poses were not yet available from *mricp*. The result was a highly accurate mapping system that provided maps even better than those obtained in the smaller, controlled system (an example is provided in Fig. 6.9).

[4] The *mricp* driver (map reference iterative closest point) was written by Tarek Taha, Centre of Autonomous Systems, University of Technology, Sydney Australia. It uses iterative closest point laser scan matching and odometry correction. A more thorough description can be found on the Player website: `http://playerstage.sourceforge.net/doc/Player-2.1.0/player/group__ComponentNavigator.html`

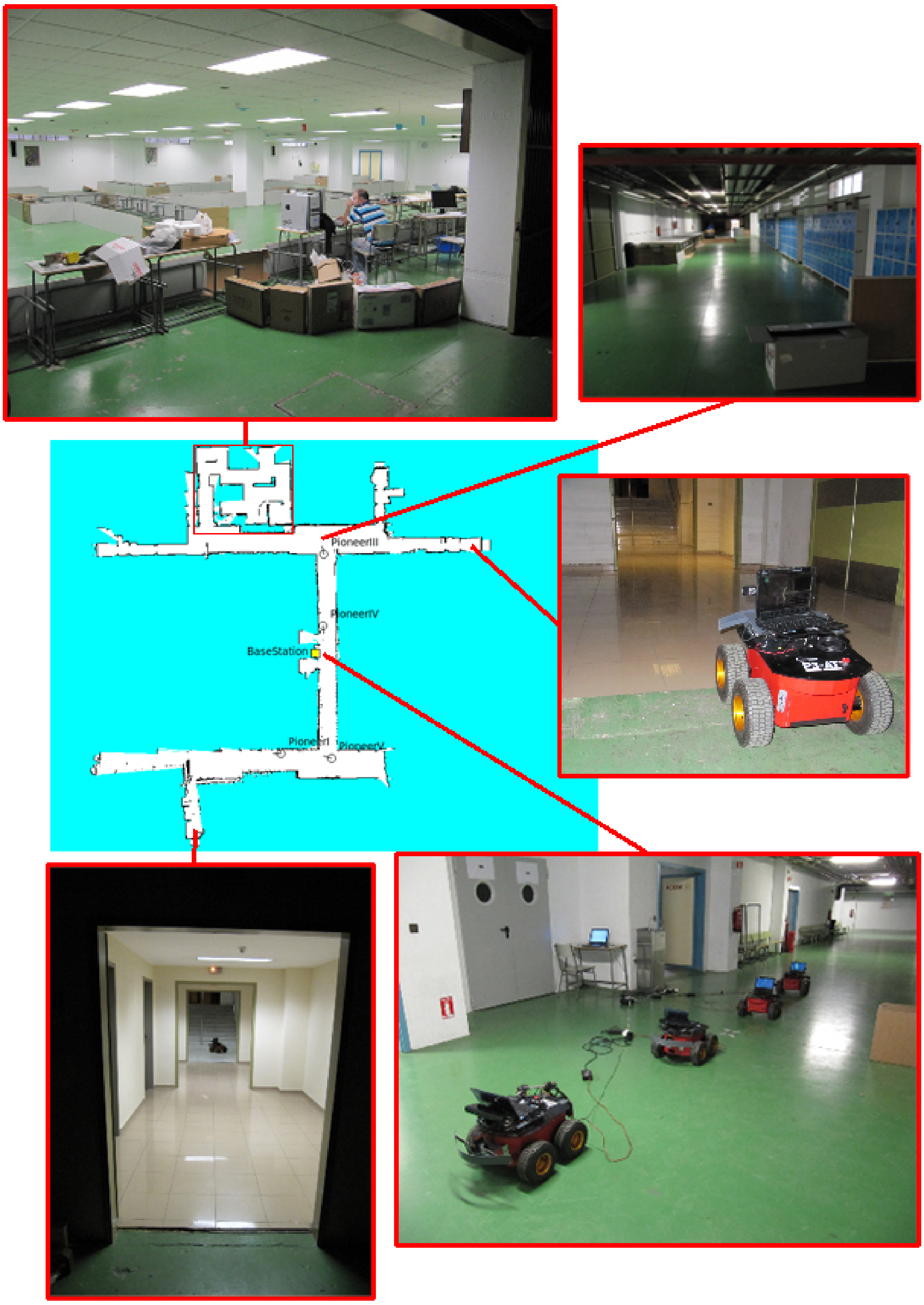


Figure 6.8: The basement of the Escuela Técnica Superior de Ingenieros, University of Seville. The small square sub-environment near the top corresponds to Environment 2 used in section 6.2. The horizontal hallway at the top is 101m long, while the vertical hallway is 65m long. There were several challenging situations such as changes in elevation (small ledges) and narrow passages.

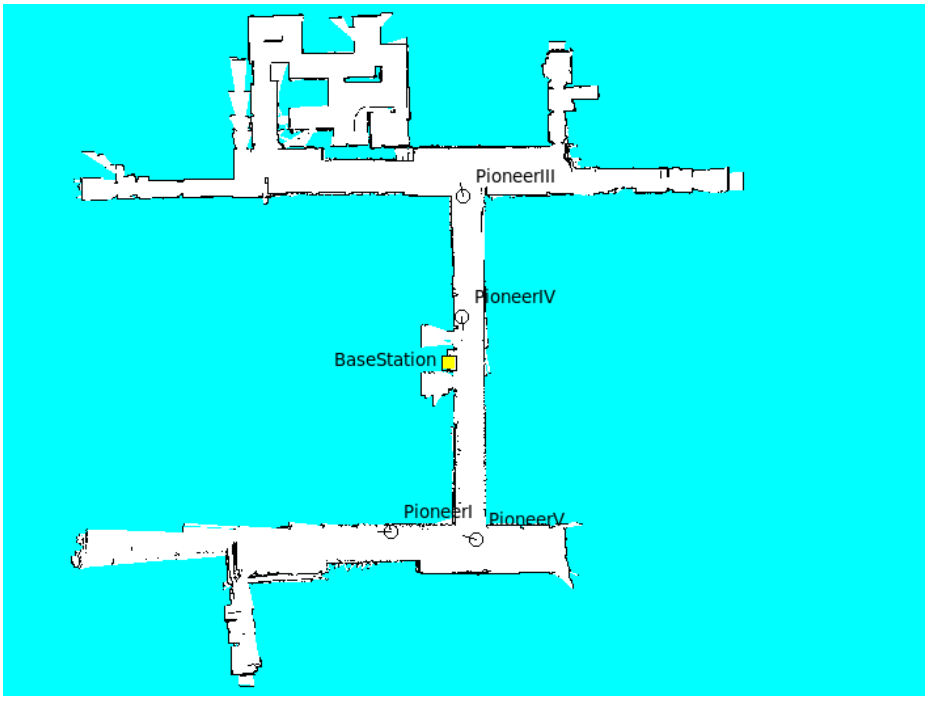


Figure 6.9: A closer look at a map provided by the *mricp* scanmatching system.

6.3.2 Adhoc Networking

In the controlled environment, access to the wireless network had been available throughout the room and communication had to be artificially constrained to test exploration under limited communication. In the hallways of the university, no continuous wireless network was available, hallways were very long, and thick walls separated parts of the environment from one another. Communication was conducted between robots and the BaseStation over an ad-hoc TCP/IP network, using each robot's Player server. Each robot opened one thread to listen for messages, and continuously tried for each of its teammates to connect, send messages, and disconnect. This proved to be a robust system; robots remained in range and sent messages regularly while in the same hall as the BaseStation; as soon as they turned the corner into another hallway, communication was lost and no map updates were received; and once they returned into the same hall, communication was reestablished and the full map was once again communicated.

6.3.3 Map Compression

In the controlled system, each robot had communicated its full occupancy grid. At one byte per cell (see section 4.1.2), a 600 × 600 grid has a size of 360KB. In the controlled system messages of this size were communicated at a rate of approximately once per second. In the uncontrolled system using adhoc networking, especially as robots travelled greater and greater distances from one another or the BaseStation, messages of this size took much longer to reach their destination; in one test, the time for a map to be transmitted took 45 seconds.

To solve this problem, maps had to be reduced to a smaller size. It had been noted that screenshots from various runs, saved in the standard PNG image format, had only taken 2KB of memory for a 600 × 600 grid – 180 times less than the grid maintained in memory during a run. The PNG format is lossless and raster-based, but uses a number of compression filters to significantly reduce image size[5]. Thus, for efficient communication, each robot converted its map to a PNG image and sent this as a chunk of binary data. The receiving robot then converted the data into a PNG image, and read occupancy grid information out of the PNG image. This turned out to be a highly effective method; data transfer times were reduced from 45 seconds to multiple messages per second.

6.3.4 Results and Performance Metrics

In the uncontrolled system, the time available for these experiments permitted implementation of three exploration algorithms: Greedy (G); Greedy with periodic return (GP); and Role-Based without role swaps (RB1) (for descriptions, see section 5.2). Several runs were conducted for each algorithm using teams of either 2 or 4 robots. As before, the results were fairly consistent across these tests and the data presented here are representative of several runs using different team sizes.

[5] More information on the PNG format is available at `http://www.libpng.org/pub/png/`.

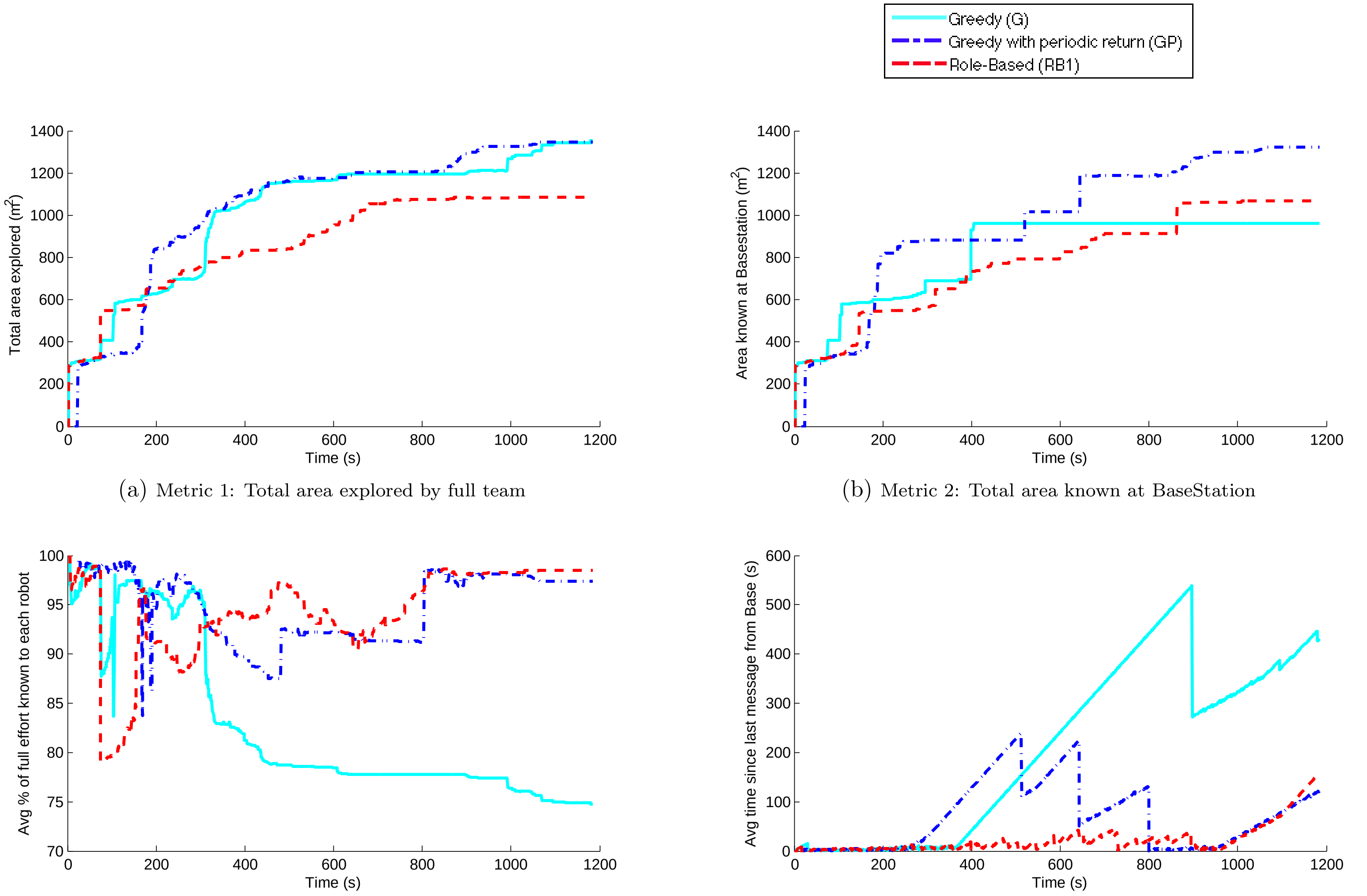
Exploration by 2 robots

The first set of results involves a team of 2 robots; performance metrics for these runs are presented in Fig. 6.10; screenshots are presented in Fig. 6.11.

At first glance, it appears that Role-Based Exploration is the weakest of the three algorithms implemented. Other than strong performance in a short initial period, RB1 lags behind G and GP both in terms of total exploration speed, and in terms of information returned to the BaseStation. RB1 demonstrates good information sharing and the best responsiveness, but the slow exploration speed is not in line with previous results.

This unusual behaviour can be explained however by examining the nature of the environment. The robots begin their exploration of the hallways from the centre of the basement, marked by the BaseStation in Fig. 6.9. There are two significant directions that robots can head: up the hallway, or down the hallway. When there are only 2 robots in the team, greedy approaches will send one robot in each direction. In Role-Based exploration, only a single robot explores while a relay follows. This explains the difference in performance.

This run demonstrates how the shape of the environment and any prior knowledge thereof can affect the decision on what algorithm to use for exploration; this is discussed in greater detail in Chapter 8.



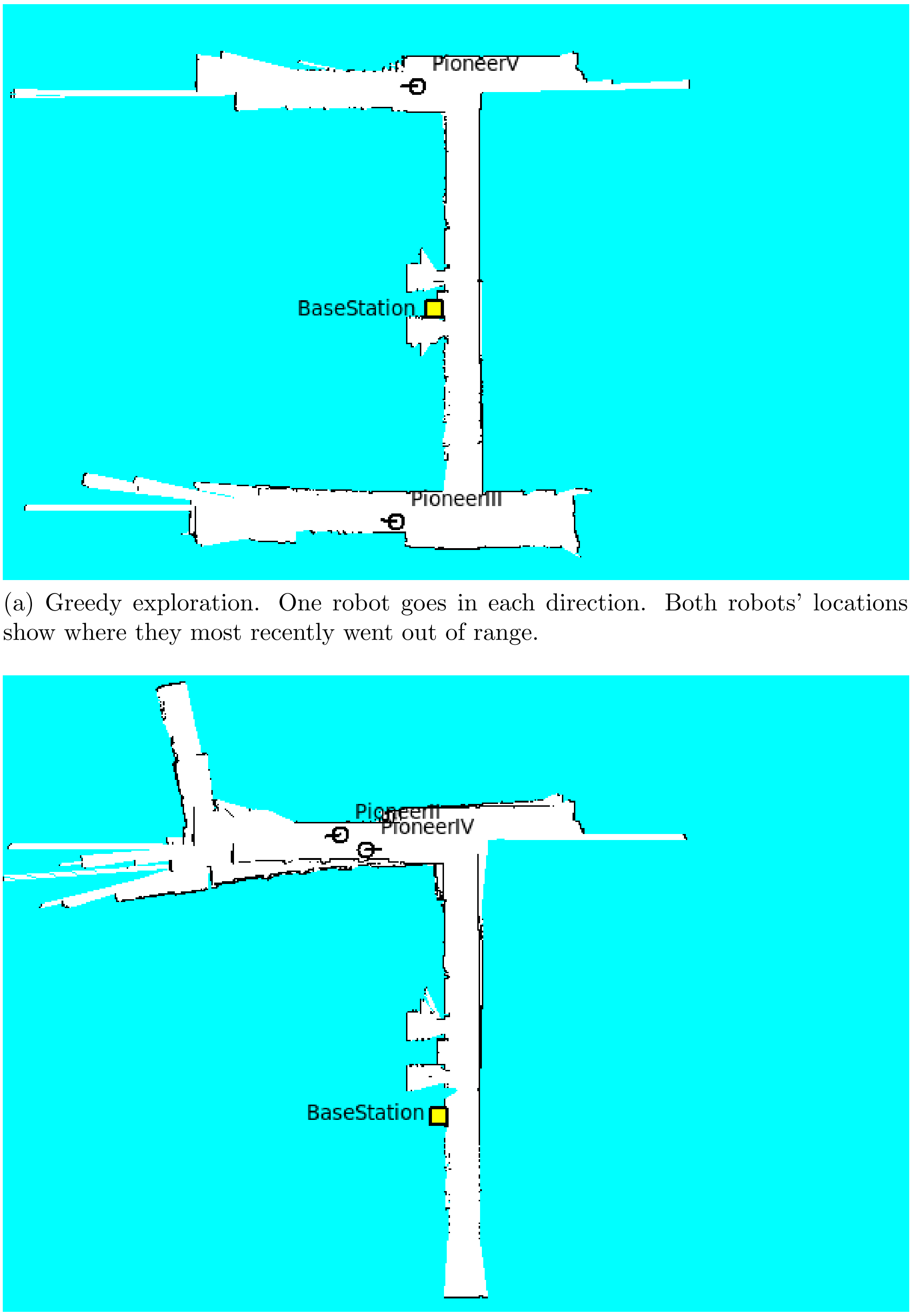
(a) Metric 1: Total area explored by full team

(b) Metric 2: Total area known at BaseStation

(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

Figure 6.10: Performance metrics for exploration of the hallways by a team of 2 robots



(a) Greedy exploration. One robot goes in each direction. Both robots' locations show where they most recently went out of range.

(b) Role-Based exploration. The explorer (Pioneer II) chooses to explore the top of the environment. The relay (Pioneer IV) follows. Pioneer II is actually deeper in the environment than shown – its location is where it went out of range. New data from deeper within the environment (top left) is being relayed to the BaseStation via PioneerIV.

Figure 6.11: Screenshots for the runs involving 2 robots.

Exploration by 4 robots

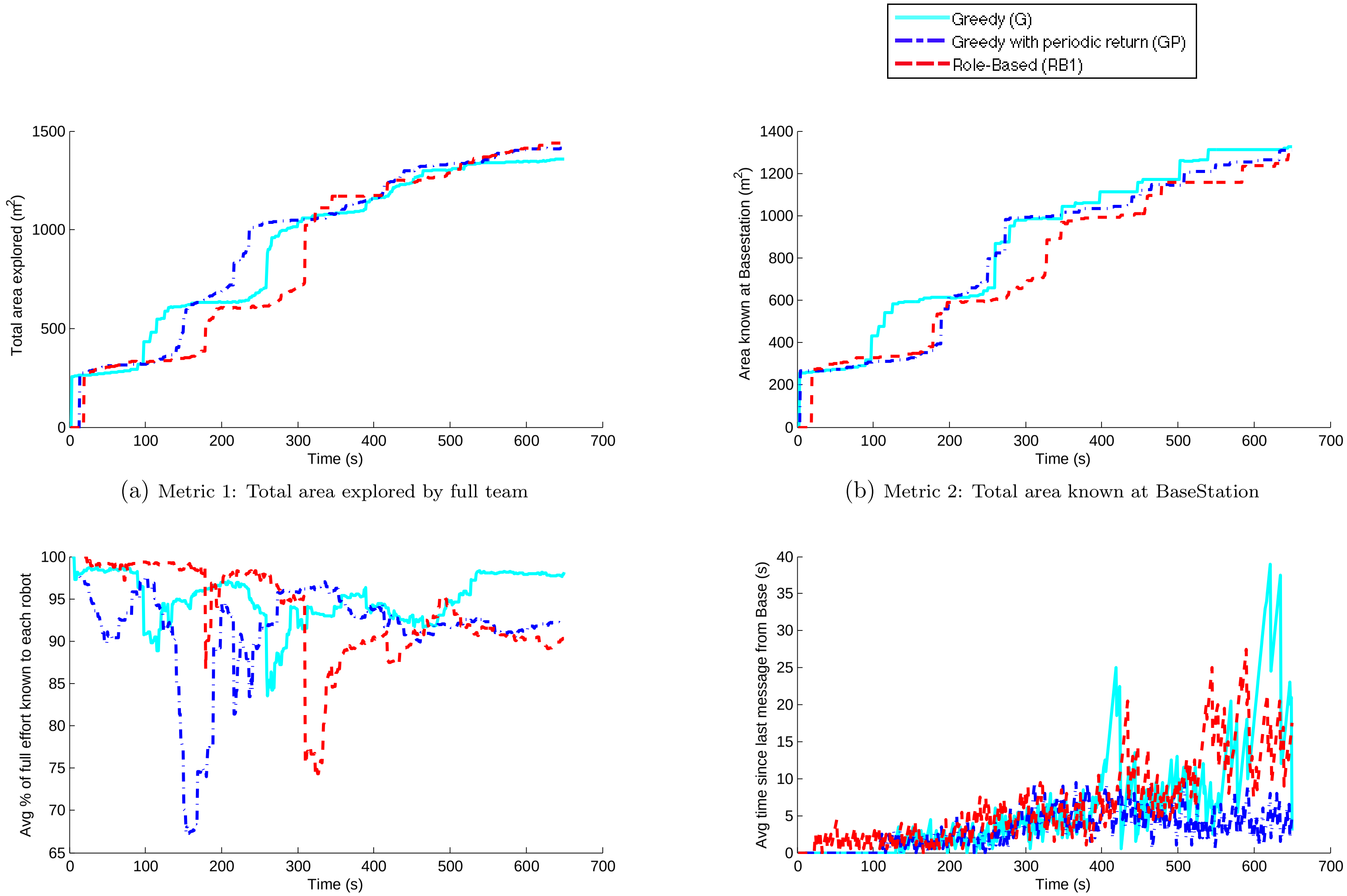
The second set of results involves a team of 4 robots; performance metrics for these runs are presented in Fig. 6.12; screenshots are presented in Fig. 6.13.

Again, RB1 lags slightly behind G and GP in term of total exploration and knowledge gained at the BaseStation. Information sharing is fairly similar for all three approaches, and GP demonstrates the best responsiveness, slightly better than RB1.

In this case, given that 4 robots are involved, the shape of the environment is not the determining factor. At this point it is worth drawing comparisons to the experiments performed in simulation in Chapter 5. The environment presented here has much in common with the "elongated" environments explored in simulation, or the Bruceton research mine explored in section 5.9. A key difference here however is that the communication range is quite large relative to the size of the environment. Robots remained within range of the BaseStation even for about 10m or 20m beyond the corners at the ends of the hallways. The main advantages of Role-Based Exploration, as discovered in Chapter 5, become apparent when communication range is small relative to the size of the environment. Consequently, the results found in these experiments do not disagree with the results found in simulation.

A better test in reality may have been a more complex environment, with greater communication challenges (or alternatively, an even larger environment). Such experiments are left as future work. Nevertheless, the runs shown here demonstrate that Role-Based Exploration can be implemented on a team of real robots, and that its performance even in unfavourable conditions is similar to existing greedy approaches.

A full guide to when Role-Based Exploration is most suitable for use (and when not) is presented in Chapter 8; the next section ties these results to those obtained in simulation.



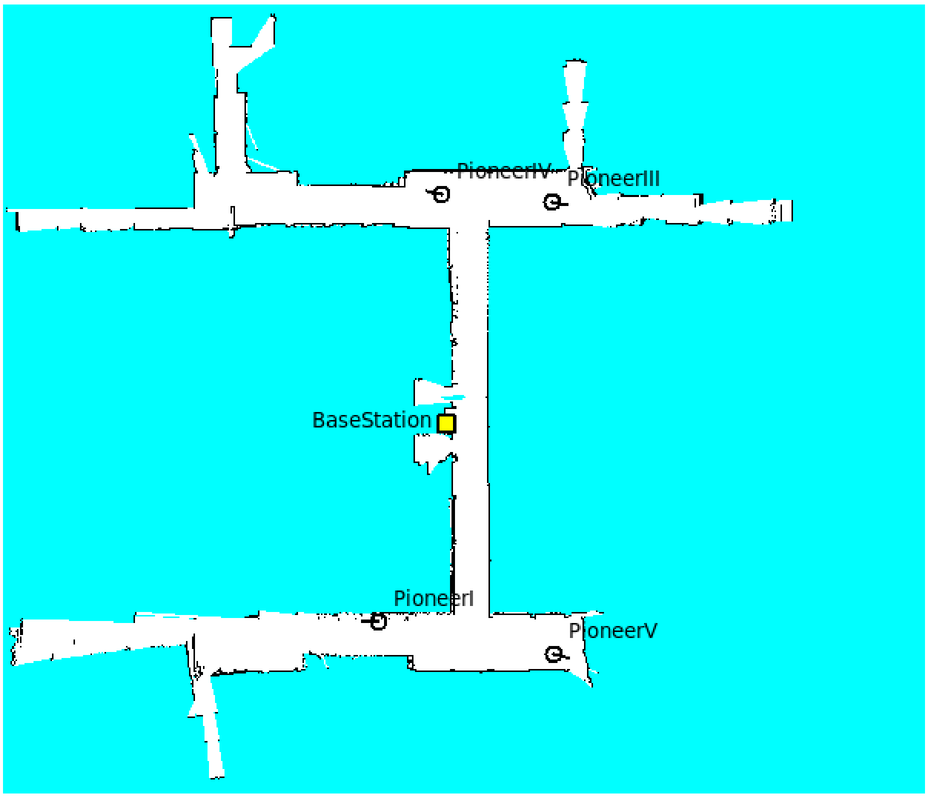
(a) Metric 1: Total area explored by full team

(b) Metric 2: Total area known at BaseStation

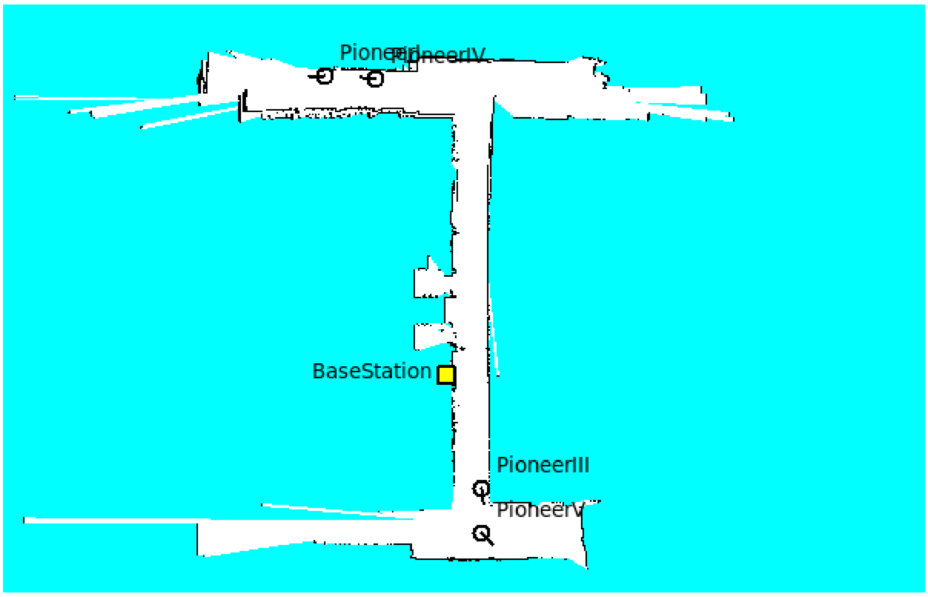
(c) Metric 3: Percentage of current team knowledge known to each robot, average over all robots (information sharing)

(d) Metric 4: Time since last message received from BaseStation, average over all robots (responsiveness)

Figure 6.12: Performance metrics for exploration of the hallways by a team of 4 robots



(a) Greedy exploration. Two robots go in each direction. When they reach the end of the hall, they split up, each choosing opposite directions in both cases. Again robots' locations indicate where they last were in range; they are likely even deeper in the environment than shown.



(b) Role-Based exploration. Pioneer IV is a relay for Pioneer I; Pioneer III is a relay for Pioneer V.

Figure 6.13: Screenshots for the runs involving 4 robots.

6.4 Summary and Comparison to Simulation

Role-Based exploration was successfully implemented and tested on a team of real robots. Robots were able, in real time, to explore and map unknown environments; perform roles assigned to them; calculate skeletons of known, free space; agree on rendezvous locations; successfully meet at those rendezvous locations; and successfully relay new information from Explorers to the BaseStation in an efficient manner. Furthermore, robots were able to successfully swap roles and maintain a consistent exploration effort even with a dynamically changing team hierarchy.

The sample set of experiments conducted with the team of real robots is considerably smaller than the sample set of experiments conducted in simulation, and consequently the conclusions drawn from the experiments in this chapter are not as definitive as the conclusions drawn in Chapter 5. However, the results obtained in reality have much overlap with the results obtained in simulation, suggesting that the conclusions drawn in Chapter 5 are reasonably valid.

~

This concludes the presentation of experimental results; the next chapter discusses various characteristics of Role-Based Exploration in greater detail.

Chapter 7

Discussion

Chapter 4 described Role-Based Exploration, and Chapters 5 and 6 demonstrated its behaviour in simulation and reality, respectively. This chapter provides a deeper analysis of Role-Based Exploration by examining the Role Swap Rule in greater depth. It further discusses the possibility of having higher branching factors in the team hierarchy, lists some additional advantages of the approach, and explores some points of failure, offering potential solutions where possible.

7.1 The Role Swap Rule in Greater Depth

The Role Swap Rule leads to more complex behaviour than one might at first presume. This section explains why the rule was chosen over competing solutions, and details some specific scenarios that must be taken into consideration when applying the rule.

Average Path Cost as an Alternative Criterion

The Role Swap Rule described in section 4.3.2 was not the only method attempted to solve the problem of creating a dynamic hierarchy; there are several alternatives. An early attempt tried to minimise *average* path cost, instead of maximum path cost, see Table 7.1. :

Consider two robots A and B, each having destinations D_A and D_B, respectively. Let $\gamma(u, v)$ represent the path cost from location u to location v in a given map. Suppose A and B have encountered one another and established a communication link. If

$$\frac{\gamma(A, D_A) + \gamma(B, D_B)}{2} > \frac{\gamma(A, D_B) + \gamma(B, D_A)}{2}$$

then let A assume B's role, state, and location in the tree, and let B assume A's role, state, and location in the tree.

Table 7.1: An early attempt to create a dynamic hierarchy that compared average path cost

However, this rule turned out to be too loose: in certain situations where role swaps were desired, they would not occur. Consider the example in figure 7.1. Explorer A and Relay B set out to explore. A chooses to explore the frontier at F_1 and reaches the end of the hallway, while B takes up a position at the beginning of the hallway where it can communicate with both the BaseStation and A. The only remaining frontier is now at F_2 – the path cost for A to reach it is displayed in light green, whereas the path costs if A and B were to switch roles are displayed in dark green.

Using an average path cost criterion, no role swap would occur; the cheapest

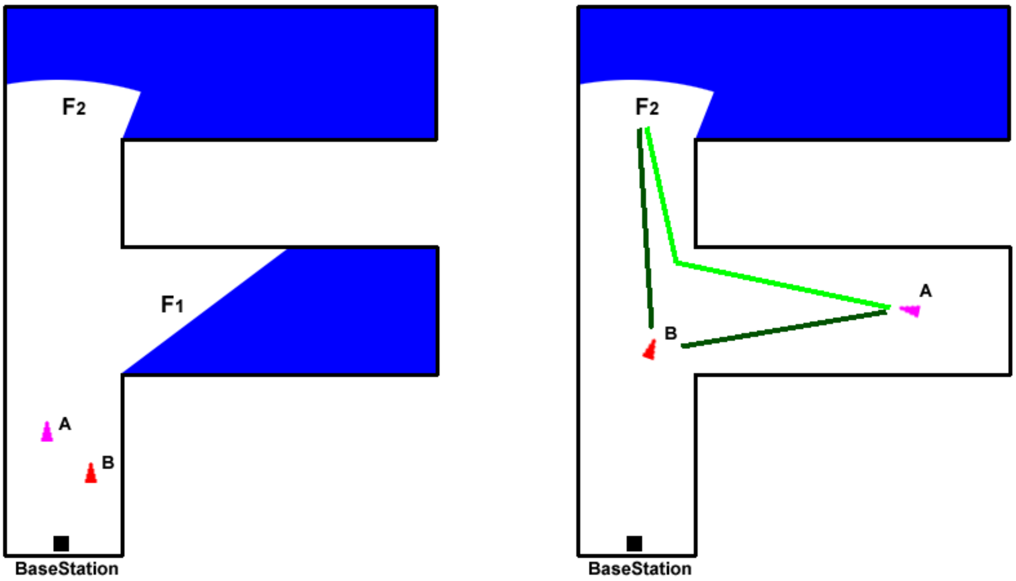


Figure 7.1: A situation where a role swap criterion based on average path lengths fails

average path cost is for A to remain an explorer and travel to F_2 while B stands and watches. But this is inefficient; it would be better for B to travel to F_2 to explore while A becomes a relay. There is a risk that for a short period the connection to the BaseStation may be lost (as B wanders out of range, and A has not yet wandered into range), but overall the environment would be explored faster.

This is only a single simple example, but there are many similar situations where a role swap based on average path cost is ineffective. The Role Swap Rule that was eventually chosen (using maximum path costs) does successfully solve this situation, and indeed guarantees that the longest path in the local scenario (of the two robots considering a role swap) is eliminated.

Load Balancing and a Stricter Role Swap

It would have been useful to formally prove that the eventually chosen Role Swap Rule (standard version, using maximum path cost as a criterion) is optimal in some

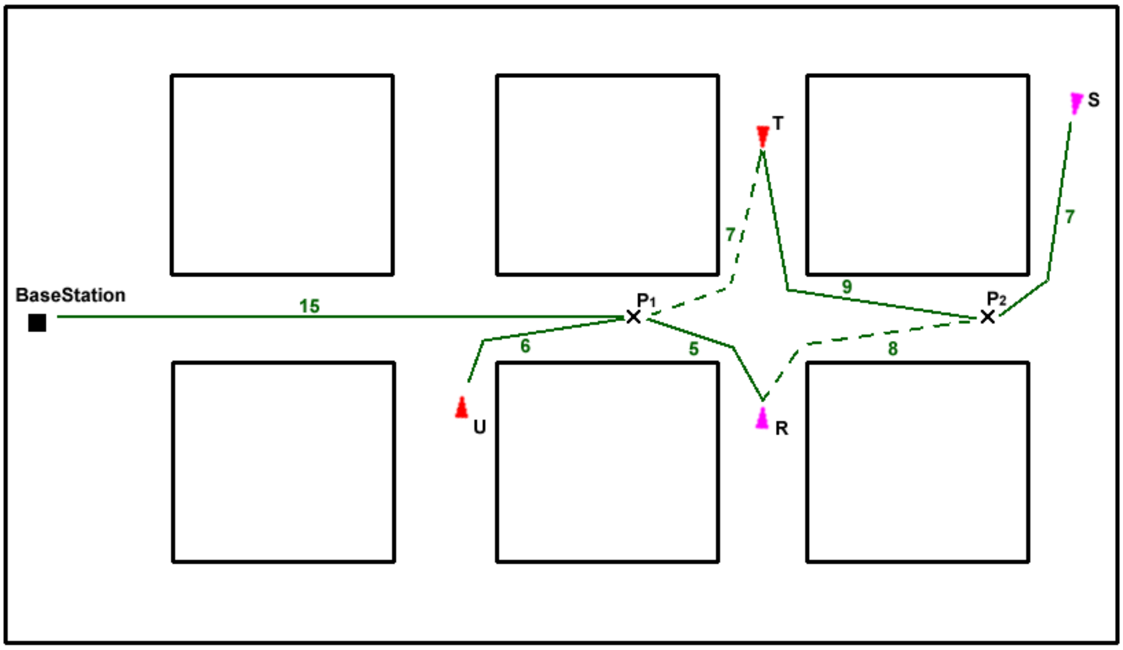


Figure 7.2: A situation warranting a stricter role swap criterion. Numbers indicate path costs.

manner, or at least to prove that it never worsens the flow of new information to the BaseStation. However, unfortunately this was not possible, since in some situations this is simply not the case.

Consider the example in Fig. 7.2. R and S are explorers; T and U are relays. R and U are about to rendezvous at P_1; S and T are about to rendezvous at P_2. Suddenly R and T enter one another's communication range and have to decide whether to swap roles or not.

In this example, the path costs might be as follows: $\gamma(R, D_R) = 5$; $\gamma(T, D_T) = 9$; $\gamma(R, D_T) = 8$; $\gamma(T, D_R) = 7$. Since $max\{5, 9\} > max\{7, 8\}$, R and T swap roles and R becomes a relay for S while T becomes an explorer with parent relay U.

If we consider the time it would have taken for new information known by R to reach the BaseStation, it is likely that this will increase. T and R share all relevant information when communicating, so T knows the same information as R. However, it will take T longer to rendezvous with U and subsequently U will relay new information to the BaseStation at a later point in time.

On the other hand, R's path to P_2 is now shorter than T's would have been, so new information discovered by S will likely be relayed to the BaseStation faster. This is what is meant by “load balancing” when describing the Role Swap Rule: new information travelling up one branch may reach the BaseStation later as a result of a role swap, but this loss is offset by an increase in the speed of information relay in another branch.

One way to guarantee that new information never reaches the BaseStation slower is to use a stricter Role Swap criterion that takes such situations into account. In other words, whenever a role swap is about to occur, a second check could be made to examine whether new knowledge known by the robots engaged in the swap would reach the BaseStation slower, and if yes to discard the swap. This would make it possible to formally prove that relay of new information to the BaseStation occurs, at worst, at a rate equal to Role-Based Exploration without role swaps.

However, given the dynamism of the system, the difficulty in accurately modelling communication ranges and estimated time of entering a teammate's communication range, and complete lack of knowledge regarding what may happen further down the line (perhaps there would be more role swaps along the way), the overhead of introducing such a stricter criterion was not considered worthwhile. Two of the Role Swap Rule's best characteristics are its simplicity, and its ease of implementation.

7.2 Hierarchies with Higher Branching Factors

The implementation described and analysed in this thesis only considers chains of robots in a hierarchy having a branching factor of 1 (except for the root). In other words, a robot may have at most a single child and a single parent. What if this limitation were to be relaxed – for example, what if one relay were to serve two (or three, or four) explorers? Multi-robot rendezvous is a well studied problem, and has been approached by multiple authors [2, 82, 30].

There is no inherent reason not to attempt such a hierarchy, and certainly the results would be interesting to compare. However, such an implementation would significantly increase the complexity of team coordination. A relay serving two explorers, for example, could perform rendezvous in two ways: (i) rendezvousing with both explorers at the same time; (ii) rendezvousing first with one and then with the other in a single trip before returning to the BaseStation; or (iii) rendezvousing with the first explorer, returning to BaseStation, rendezvousing with the second explorer, returning to BaseStation, etc.

Each of these situations introduces additional challenges. In case (i), rendezvous by three (or more) robots requires very careful timing in order for them all to meet at the same time, and risks one robot delaying two others if it is late. In case (ii), it is possible that the relay would have to travel long distances before it could return to the BaseStation. In case (iii), explorers would be on their own for long periods, and the intervals in which new information would reach the BaseStation would be much greater.

As a result, it seems that changing the branching factor of the hierarchy may not be worth the additional complications introduced. In some rare cases environment structure may suggest a higher branching factor (for example, a long hallway with two large rooms at the end would suggest using a relay with two explorers). But even then, the emergent behaviour that arises with the use of the Role Swap Rule (see section 4.3.4) typically deals well with any shape of environment.

7.3 Further Advantages

The strengths of Role-Based Exploration as compared with some competing approaches have already been discussed in Chapters 5 and 6: Role-Based Exploration is generally better than greedy approaches at relaying new information to the Base-Station at regular intervals, demonstrates better inter-teammate connectivity and information sharing, and allows for tighter, more responsive control of the team, all while still providing faster exploration than leader-follower approaches. However, there are some further possible advantages worth mentioning here.

Cooperative Localisation

Given that rendezvous is an essential element of the approach, Role-Based Exploration leads to frequent meetings of teammates. Meetings between robots have been used for a long time to improve mutual localisation, and cooperative localisation is a well-studied problem [121, 48, 161, 50]. Thus, an approach such as Role-Based Exploration that plans on repeated mutual observation can lead to cleaner maps and better localisation.

Acyclic Communication

In the implementations of Role-Based Exploration presented in this thesis, information was gained in a monotonic manner, adding to or overwriting previously gained information. However, many multi-robot systems elsewhere use probabilistic information gathering methods, where each information update received through the network contributes to the degree of confidence in a given estimate of the map [135, 26, 86]. In a multi-robot team where messages are passed over multiple hops, there is a risk that a single information update will be received more than once via separate paths, incorrectly increasing the confidence in that estimate. This can be solved by data tagging messages and maintaining a record of their history, but in large networks this could lead to unnecessarily large messages. As a result, *acyclic* communication networks are of great interest in some multi-robot systems, and are assumed or explicitly created in several cases (*e.g.* [87, 26]). Therefore, when such additive probabilistic information gathering methods are employed, Role-Based Exploration could potentially provide a solution: an acyclic team hierarchy already exists, and can be used to prevent double counting of information updates.

7.4 Points of Failure and Possible Solutions

The weaknesses of Role-Based Exploration as compared with some competing approaches have already been discussed in Chapters 5 and 6: Role-Based Exploration does not always explore as fast as greedy approaches, and does not maintain inter-teammate connectivity and information sharing as perfectly as leader-follower approaches. This section examines some additional potential points of failure and problems that may arise, and proposes some possible solutions.

7.4.1 Failing Robots

Robots may fail for a number of reasons (and often do). Motors or sensors may fail, batteries may run out, the localisation system may become irrecoverably lost, software may crash, or environmental factors may destroy the robot. In greedy approaches this does not affect the failed robot's teammates, which continue exploring as if nothing happened. In Role-Based Exploration, however, this is a scenario that must be dealt with carefully.

1. **Failure of a Relay**. When a relay fails, there are two consequences: the relay's information is not returned to the BaseStation, and the relay's child explorer is left waiting for it at rendezvous.

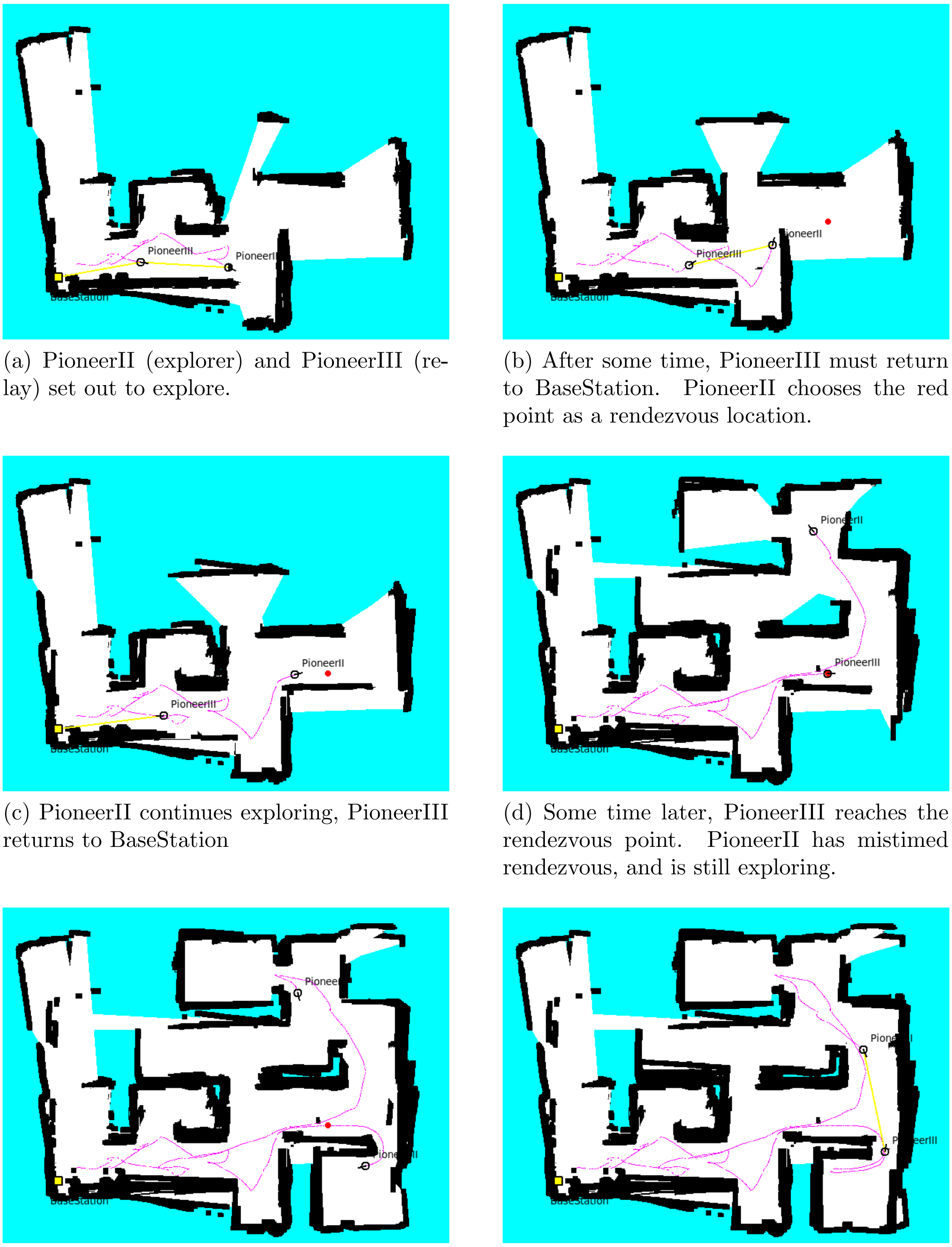
A possible solution to this problem is for the explorer to revert to GP exploration, *i.e.* for the explorer to return to the BaseStation, communicate its knowledge, and proceed with normal exploration, periodically returning to the BaseStation. Since, in its first return to the BaseStation, it is likely to choose the same path that the missing relay would be taking, there is a strong chance of it encountering the relay along the way if it happens to have been delayed. If the relay is not encountered, it is likely to have failed, and the explorer can continue with GP exploration.

Clearly the explorer should not assume failure immediately, but wait for a time at rendezvous before making the decision to return to BaseStation. This delay should be substantial: it is possible that the original relay swapped roles with a teammate, and that the new relay has a longer path to rendezvous than the original relay did.

2. **Failure of an Explorer**. When an explorer fails, there is only a single consequence as concerns its teammates: its parent relay will be left waiting for it at the agreed rendezvous.

   The simplest solution to this problem is the same – the relay can wait for a given amount of time, and ultimately decide to become an explorer itself, explore for a time, and periodically return to the BaseStation.

   A better solution may be for the relay to use the "waiting time" to explore the nearby vicinity, and return to the same rendezvous location once (or twice) to give the explorer a second (or third) chance. If after several attempts the explorer still hasn't turned up, then the relay can convert to becoming a periodically returning explorer. There are two reasons why this solution may be better: the explorer may have been delayed (*e.g.* due to a role swap), or, as happened several times with the team of real robots, may be disoriented in a



(a) PioneerII (explorer) and PioneerIII (relay) set out to explore.

(b) After some time, PioneerIII must return to BaseStation. PioneerII chooses the red point as a rendezvous location.

(c) PioneerII continues exploring, PioneerIII returns to BaseStation

(d) Some time later, PioneerIII reaches the rendezvous point. PioneerII has mistimed rendezvous, and is still exploring.

(e) After waiting for a bit, PioneerIII decides to explore a nearby frontier. PioneerII finally realises that it's time to rendezvous.

(f) PioneerIII returns to rendezvous to give PioneerII a second chance. Pioneers II and III meet, and exploration can proceed as normal.

Figure 7.3: Demonstration of how a relay can deal with a failed rendezvous

nearby location. By exploring its vicinity, the relay may happen to encounter its child explorer lost nearby, and via sharing of maps help it to disorient itself.

These solutions to dealing with failed teammates are not difficult to implement, and were indeed effectively demonstrated both in simulation and on the team of real robots. A series of screenshots demonstrating this behaviour on the Pioneer robots (from chapter 6) are provided in Fig. 7.3.

7.4.2 Dynamic Environments

In many robotics applications, the environment may change as the exploration effort progresses. Robots exploring a building may find that a previously open door has been closed. Robots performing search and rescue may find that a ceiling has collapsed or that rubble has shifted unexpectedly.

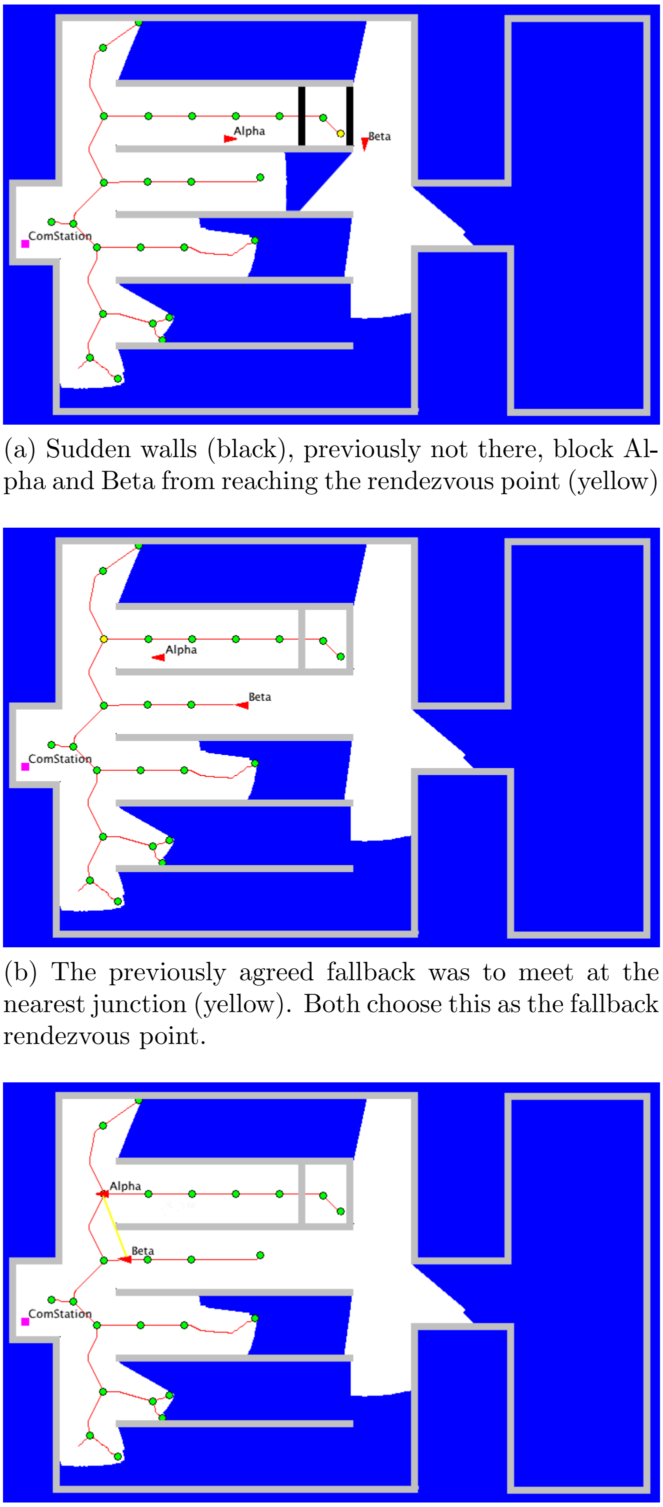
In Role-Based Exploration, robots do regularly replan their paths, and even if a previously open passage becomes blocked robots will find alternative paths to the same destination. However, what happens if there are no alternative paths, *i.e.* what happens if the robot's goal becomes impossible to reach?

There are three possible goals for a robot performing Role-Based Exploration:

1. **Frontiers**. If the goal is a frontier (*i.e.* if the robot is exploring), then it is straightforward to choose another frontier instead, and exploration proceeds as normal.

2. **BaseStation**. If the goal is the BaseStation (*i.e.* if the robot is a relay bringing new information back, or any robot that has completed its exploration task), then it is possible that this is an irrecoverable situation. Such a scenario would affect any exploration algorithm equally.

3. **Rendezvous**. If the goal is a rendezvous location, then possible solutions exist. Let's say explorer A and relay B are due to meet at P_1, but P_1 becomes



(a) Sudden walls (black), previously not there, block Alpha and Beta from reaching the rendezvous point (yellow)

(b) The previously agreed fallback was to meet at the nearest junction (yellow). Both choose this as the fallback rendezvous point.

(c) Alpha and Beta meet at the new rendezvous point and exploration can proceed as normal.

Figure 7.4: Using fallbacks when rendezvous points cannot be reached in dynamic environments

impossible to reach (*e.g.* it is in a room whose doors have locked shut, and A and B were going to enter the room from opposite ends). Recall that a rendezvous location is chosen by the explorer from among a set of candidate points. One possible solution could therefore be to maintain a list of *fallbacks*. In other words, other rendezvous candidates near the chosen candidate could be maintained in a list of priority. If A and B cannot reach P_1, they both try to meet at a previously agreed P_2; if they cannot reach P_2, they try P_3; and so on. A demonstration of how this might work is provided in Fig. 7.4.

It is possible that a rendezvous point may become impossible for one teammate to reach, but not the other. This situation is parallel to the "Failed robot" scenario discussed in section 7.4.1 and can be solved in the same manner.

7.4.3 Unsuitability of Certain Platforms

Role-Based Exploration requires robots to be able to turn in place. Relays in particular must repeatedly retrace paths to ferry information back and forth between BaseStation and Explorers. This is not a problem for tracked ground vehicles or for most skid-steered platforms. However, clearly there are certain robot platforms that are not capable of instant turns. A fixed-wing UAV, for example, requires a large turn radius to return in the direction whence it came.

In addition, there may be environmental factors that do not allow robots to return to where they came from. Dynamic environments have already been discussed; but even non-dynamic environments may contain situations (such as a significant drop for a ground robot) that are irreversible.

Such scenarios must be taken into account when considering the use of Role-Based Exploration – in some scenarios it simply is not a suitable solution.

Chapter 8

Conclusions

This chapter concludes the thesis. A summary of contributions is followed by an explanation of when exactly Role-Based Exploration is most suitable for use. Finally, future directions are discussed.

8.1 Summary of Contributions

The original motivating questions for this thesis were: How can a team of robots be coordinated to explore a previously unknown and communication-limited environment as efficiently as possible; and how can new information obtained by this team be gathered at a single location as quickly and as reliably as possible?

In response to these questions, this thesis proposed Role-Based Exploration, a novel exploration algorithm for teams of mobile robots exploring unknown, communication-limited environments. Role-Based Exploration demonstrates several useful characteristics: it leads to efficient and regular relay of new information to a single, central "BaseStation"; it results in efficient coordination of the team and prevents redundant task completion; and it allows for straightforward monitoring of the team effort and reasonably tight control of the full team even when some robots are beyond communication range. Two significant improvements to the approach were introduced. In the first, improved choices of rendezvous allow robots to find one another and meet faster. In the second, robots can swap roles when it is useful to do so, leading to a dynamic team hierarchy. The team adjusts reactively to the availability of communication, and to different types of environments.

Extensive experiments in simulation compared Role-Based Exploration to some competing algorithms, notably greedy and leader-follower approaches. Greedy approaches can lead to quicker exploration of the full environment, and leader-follower approaches can maintain better connectivity and information sharing within the team. However, Role-Based Exploration provides a useful trade-off between the two, particularly as communication becomes less and less reliable, as more robots are used in the team, and as environments become more complex. In such situations, Role-Based Exploration provides regular updates to the BaseStation (which greedy approaches do not) while still exploring the farthest reaches of the environment (which leader-follower approaches do not).

Role-Based Exploration was also successfully demonstrated on a team of Pioneer robots. In a first series of experiments, the robots jointly explored a small, controlled environment, while in a second series of experiments, the team explored fully autonomously with no external control of the environment or communication availability. Using Role-Based Exploration, the robots were able to successfully explore the hallways of a large building, meet at pre-decided rendezvous locations, share their maps and coordinate their actions, and regularly communicate new knowledge back to a central BaseStation.

8.2 When to use Role-Based Exploration

Extensive experiments using Role-Based Exploration have demonstrated the approach's strengths and weaknesses. For multi-robot systems having the same characteristics as those used throughout this thesis (a homogeneous team of robots cooperatively exploring an unknown environment), the following guide could be used to determine when Role-Based Exploration is most applicable.

Role-Based Exploration *should* be used when:

1. the highest priority is to receive quick and regular information updates at a central BaseStation.

2. the communication range of the robots is small compared with the size or complexity of the environment. The more difficult it is for the robots to communicate, the greater the benefit is in having mobile relays improving connectivity and information sharing within the team.

3. there are a large numbers of robots in the team (four or more). This is particularly true in environments with fewer frontiers. The more robots there are in the team, the more important it is for them to coordinate their effort well.

4. environments contain many obstacles or long hallways and passages.
5. the SLAM system is of a type that benefits from mutual localisation between the robots, or an acyclic communication protocol is desirable.

Role-Based Exploration should *not* be used when:

1. quick exploration of the full environment is the highest priority. Greedy methods perform this faster in most cases.
2. full connectivity of the team is required at all times.
3. availability of communication is ubiquitous throughout the environment.
4. the team is composed of robots that have difficulty turning on the spot, or cannot retrace their paths due to environmental factors.
5. the desired application is not primarily exploration-oriented. Role-Based Exploration was designed for exploration of unknown environments, and better algorithms are likely to exist for other applications, such as coverage of open space.

8.3 Future Work

There are many ways in which Role-Based Exploration could be extended or improved. This section explores several such possibilities.

Dynamic Hierarchy Structures

In the current implementation of Role-Based Exploration with role swaps, robots may jump around within the team hierarchy and assume different roles. However, the structure of the hierarchy is fixed from the start, and the structure itself never changes. If the exploration effort begins with 6 relays and 3 explorers, these numbers of each role will stay the same throughout the entire effort, even if robots are swapping roles. Why not allow for a dynamic structure, where branches of the hierarchy can lengthen or shorten as required?

A dynamic hierarchy structure would certainly introduce additional challenges: any robot in a branch affected by a change in hierarchy structure would need to be informed of the change. Given that robots are likely to wander in and out of range, this would need to be managed carefully. Furthermore, the emergent behaviour resulting from the Role Swap Rule already provides a reasonably good solution. As demonstrated in section 4.3.4 and Fig. 4.13, the nature of Role-Based Exploration with role swaps is such that the team adjusts dynamically to the shape of the environment.

There is however at least one situation where a change in hierarchy would be quite beneficial, namely the very early stages of the exploration effort. Greedy methods tend to outperform role-based methods significantly at the start, since there are twice as many robots exploring. A useful adjustment to Role-Based exploration could be to begin the exploration effort with all robots acting as explorers, and certain robots only taking on the roles of relays once the limits of communication ranges are reached. This as well would need to be managed carefully: robots would likely not be able to predict the moment that they lose contact, and agreements on which robots become explorers and which robots become relays would need to be formed *before* communication is lost.

An ability to dynamically form a hierarchy could also be useful in situations where robots do not start with a common frame of reference. For example, if two groups of robots enter a mine from different entrances and meet in the middle, it would be useful for them to be able to agree on some sort of hierarchy “on-the-fly”. This would be a very interesting and useful behaviour to develop in future work.

Heterogeneous teams

The current implementation does not take into account potential heterogeneity in the team. It is possible that different types of robots with different capabilities and different sensor loads may be involved in the same effort.

For example, it is possible that the robot team could be composed of fast, simple robots (ideal for relaying) and more advanced robots with more sophisticated sensors (ideal for exploring). In such a scenario, the Role Swap Rule would need to be adjusted to take robot types and their ideal roles into account.

As another example, the potential use of aerial robots for creation of a communication infrastructure has received more and more attention recently. Using UAVs as mobile relays that can either land in strategic positions or ferry information between robots without taking ground-based obstacles into account could be a very interesting extension to Role-Based Exploration.

Incorporating Prior Knowledge

Teammate Prediction

Gains in exploration efficiency could likely be achieved with better prediction of where teammates may be when they are out of range. This is particularly true for explorers returning to rendezvous. A scenario observed often in simulation was that an explorer took a long path back to a rendezvous point, when it would have been faster to predict the relay's location and intercept it at an earlier point. This is particularly true in environments with loops and multiple passageways. The route of a relay should be highly predictable to its child explorer, and this knowledge could be used to greater effect.

Environmental Prediction

It is entirely possible that in some robotics applications prior knowledge of the environment exists. For example, if a building needs to be inspected, floor plans of the building may be available. In such cases, plans performed in advance could help to steer explorers into areas of greatest interest. One possible way to do this would be to incorporate desirability into the utility calculations of frontier polygons. Polygons close to desired areas would thus be chosen more likely than those farther away.

Another problem that was observed in simulation is that an explorer doesn't prioritise "finishing the job". The point at which an explorer turns to return to rendezvous is timed exactly, and currently there is no system to slightly adjust the timing of the state change even when it is of advantage. This means that in some scenarios an explorer will explore 90% of a room before it turns to rendezvous, and must then subsequently come all the way back into the room to finish the job. This is expensive, and it would make more sense to leave the relay waiting for a short time, in order to complete exploration of the room and not have to visit it again.

This is certainly not an easy problem, as it involves knowledge of concepts such as what shape a room may have. Furthermore it cannot be solved perfectly when there is no prior knowledge of the environment; in the final unexplored 10% of a room, there may be a further passage leading to another room. Nevertheless, some degree of environmental prediction could lead to significant gains in exploration efficiency.

Planning Under Uncertainty

In recent years, there has been a significant increase in the use of probabilistic methods for motion planning, navigation, manipulation and target tracking. In particular, partially observable Markov Decision Processes (PO-MDPs) have been applied to a variety of tasks.

A PO-MDP is much like a standard Markov Decision Process (MDP) in that it can be used to make decisions in sequential processes where outcomes are partly random or uncertain. In an MDP, the world is modelled as a finite set of states, where a finite set of actions are available to move from one state to another. The probability of a given state leading to another is represented by a state transition function, and rewards are provided for each transition according to a reward function. Where PO-MDPs differ from standard MDPs is in the *observability* of the current state: the decision making agent (*i.e.*, the robot) does not know what exact state it is in, and must use a probability distribution over the state space when making its decisions, taking all possible states consistent with its observations into account.

PO-MDPs suffer from high computational complexity since the belief space (the states that the robot believes it could be in) can be very large, and since there may be many steps required to reach a given goal. Thus the main challenge in the use of PO-MDPs involves finding efficient algorithms to solve them approximately, sacrificing optimality for time while still finding a good solution [24, 38]. Nevertheless, PO-MDPs show great promise as a tool for planning in uncertain environments and situations, and have been applied with success to robot navigation [133, 142], visual tracking [31], and motion planning [68], among others.

In Role-Based Exploration, there is a great deal of uncertainty. At the lowest level, sensor readings always carry a degree of uncertainty with them. At a higher level, teammates' locations and status are not always known. In this thesis, the mapping and exploration tasks have been kept fairly separate, and prior solutions to the problem of mapping under uncertainty have been used without much alteration (such as Player's *amcl* and *mricp* drivers, see Chapter 6). However, there could be great benefits in solving the exploration and mapping problems with a single solution. This could be approached as follows:

- Each state in the PO-MDP could represent a robot's location, current state (see Fig. 4.4), and current goal (which could be any of the following: frontier polygon, parent rendezvous location, or child rendezvous location)

- Each action could involve a direction in which to travel
- Observations would be received by the sensors, as a combination of odometry and range measurements
- The reward function could provide small rewards for increasing knowledge of the world (sensing new areas) and large rewards for reaching desired goals (such as rendezvous locations)

Reward functions would need to take robot states into account. Explorers while in state Explore should be highly rewarded by the uncovering of new areas, while Relays in state ReturnToParent should not be highly rewarded by anything other than reaching their parent. Nevertheless, using a solution such as this might lead to some interesting results. For example, a Relay, on its way to rendezvous with its child Explorer, might choose a route that allows it to explore new areas and contribute to the exploration effort while making its way in the same amount of time (rather than blindly walking straight to the rendezvous location as in the current implementation).

The state space would be huge, and there could be a large number of steps involved before reaching a given goal, so likely some sort of approximation like random sampling would be necessary. Finding a way to capture these uncertainties and model Role-Based Exploration as a PO-MDP would certainly be challenging, but could be a highly rewarding direction in which to direct future research.

Extending to Three Dimensions

Finally, the work on Role-Based Exploration to date has been limited to flat environments. Even the experiments conducted with real robots involved a mostly planar floor and range sensing in a plane. Clearly most real applications would involve information on multiple planes or in three dimensions. The move to three dimensions is a big challenge in robotics today, and much work still needs to be done on every aspect, from mapping, to navigation, to planning. Nevertheless it is useful to examine how an approach such as Role-Based Exploration might hold up if it were to be employed in a 3D environment at some point in the future.

Consider the case of a team of robots travelling in three dimensions (*e.g.* a team of UAVs or underwater robots, or a team of ground robots in a pile of rubble). Roles could be assigned in the same manner, and robots could either explore, or relay. Localisation and mapping would become quite a challenge, but there has already been much work in this direction by the robotics community. Frontier polygons would become frontier polyhedra, and certainly the calculation of such polyhedra would be more complex and possibly involve some manner of contour surfaces instead of contour lines. Path planning would require significantly more resources, and consequently frontier-to-robot assignment would require significantly more calculation. Considerably greater memory would be required for storage of maps, and thus sharing of maps would require greater bandwidth. Skeletonisation would become a much more expensive operation.

In short, computation and memory requirements would soar in the move to three dimensions. It is likely that any early approach applying Role-Based Exploration (or any multi-robot exploration algorithm) would need to apply some clever compression and optimisation techniques while hardware development catches up with requirements. All of that said, there is no inherent reason for some variant of Role-Based Exploration not to be applied to a team of robots jointly exploring in three dimensions. But there is certainly much future work to be done before that happens.

~

The exploration by multiple robots of environments with significant communication challenges remains a young field of study. To date, not much work has been done in this area, and it is hoped that the ideas in this thesis provide an early stepping stone for future work.

As technologies become ever smaller, processors ever faster, and methods to sense, act and cooperate ever more clever, robots will be used for many tasks in the near future, some of which are likely to be beyond our imagination at the present. May we steer them in the right direction!

Appendix A

MRESim: A Multi-robot Exploration Simulator

To examine and compare various exploration algorithms, a multi-robot exploration simulator (MRESim) was developed in JAVA, using the NetBeans development environment. This appendix briefly explains some important aspects of the simulation process, and finishes with a short discussion on the degree of realism that can be expected from results obtained in the simulator.

The simulation cycle

The simulation cycle is presented in Alg. 3. For purposes of simplicity, the simulator runs in a single thread. In each cycle, each robot makes a decision on where it wants to move next. Each robot is then provided with range data, and messages are passed between any two robots within range of one another. All relevant data is logged, and the user interface is updated.

User Interface and Environments

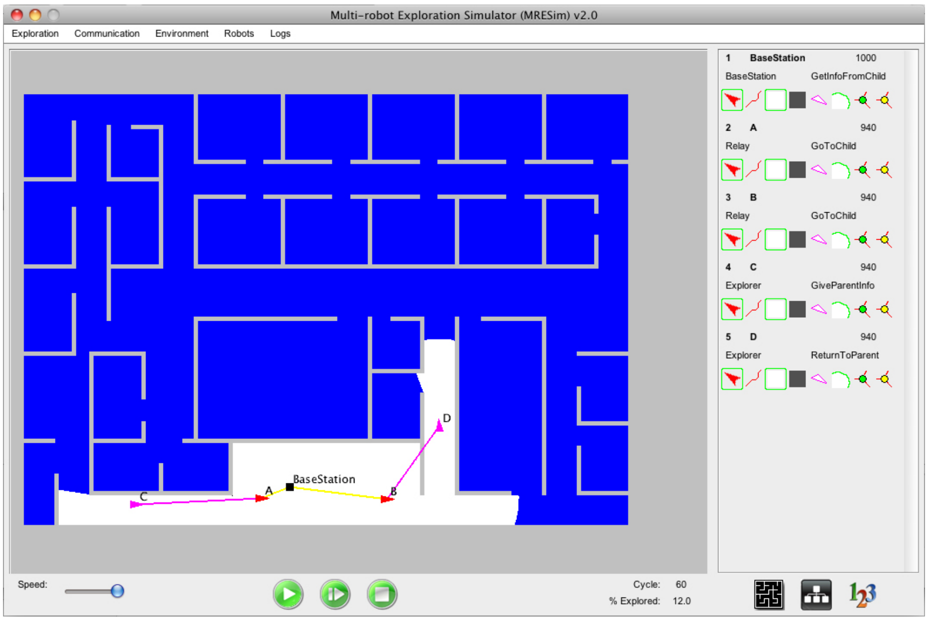
The user interface (Fig. A.1) contains a panel for each robot with toggle buttons for each of the following: location; path; free space; safe space; frontiers; communication range; skeleton and potential rendezvous points; and exact rendezvous points. There are further toggle buttons for the environment walls and the team hierarchy, and a run may at any time be paused, continued, or stopped. This means that it is very easy to examine specific aspects of any supported exploration approach.

Environments in MRESim can be uploaded from text-based or PNG files, and are 2-dimensional and binary, with each cell being either *free* or an *obstacle*.

Sensing and Mapping

Once a robot has taken a step, the simulator provides it with sensor data at its new location (see Alg. 3). This sensor data is generated using raytracing from the robot's location at 1-degree intervals in the 180-degree field of view of the robot (the same field of view as many real laser scanners). An array of 181 measurements is returned to the robot.

The robot subsequently turns this sensor data into a polygon of free space, detecting obstacles where two points are sufficiently close to one another and below the sensor's range limit (see Fig. A.2a). This free space and obstacle detection is maintained in an occupancy grid. When robots are within communication range of one another, they exchange occupancy grids.



1 BaseStation 1000
BaseStation GetInfoFromChild

2 A 940
Relay GoToChild

3 B 940
Relay GoToChild

4 C 940
Explorer GiveParentInfo

5 D 940
Explorer ReturnToParent

Cycle: 60
% Explored: 12.0

Figure A.1: MRESim Graphical User Interface

```
Input: Set $R$ of robots
 // Simulate movement, range data
foreach $r_i \in R$ do
    $nextStep = r_i.takeStep()$
    if $isValid(nextStep)$ then
        $rangeData = findRangeData(r_i, nextStep)$
        $r_i.receiveRangeData(rangeData)$
    else
        $r_i.setError(true)$
    end
end
 // Simulate communication
foreach $r_i \in R$ do
    foreach $r_j \in R,\ i\ != j$ do
        if $isInRange(r_i, r_j)$ then
            $r_i.receiveMsg(r_j.createMessage())$
            $r_j.receiveMsg(r_i.createMessage())$
        end
    end
end
 // Complete cycle
$logData()$
$updateGUI()$
```

Algorithm 3: MRESim: A single simulation cycle

Path planning

Path planning is a key component of the algorithms proposed in this thesis, as it factors highly in utility calculations of various goals, along with decisions that affect the team hierarchy. All robots in MRESim use a simple A* path planner [123], which performs sufficiently well (see Fig. A.2b).

Communication

MRESim supports a variety of communication models. Simple models include a straight-line model (any robot within radius x is considered in range, regardless of obstacles) and a line-of-sight model (any two robots that can be connected by a line that doesn't hit an obstacle are in range). However, for all of the experiments presented in this chapter, a path loss model, originally proposed by Bahl and Padmanabhan, was used [5]. This model is also used for simulating communication in the USARSim simulator used at RoboCup [21], and is considerably more realistic as it takes wall attenuation into account. In this model, communication strength is calculated as follows:

$$S = P_{d_0} - 10 \times N \times log_{10}\left(\frac{d_m}{d_0}\right) \begin{cases} nW \times WAF & nW < C \\ C \times WAF & nW \geq C \end{cases}$$

where P_{d_0} is the reference signal strength, N is the path loss rate, d_m is the distance, d_0 is the reference distance, nW is the number of obstructing walls, WAF is the wall attenuation factor and C is the maximum number of walls to consider. For all of the experiments described in chapter 5, values of $N = 1$, $WAF = 5$ and $C = 4$ were used, since empirically they demonstrated a realistic decay in communication strength through obstacles.

Typical communication ranges using this model are displayed in figure A.2c.

Realism of Simulator Results

Clearly MRESim does not take into account a number of factors that any robot system in the real world would have to consider. The most significant ones are:

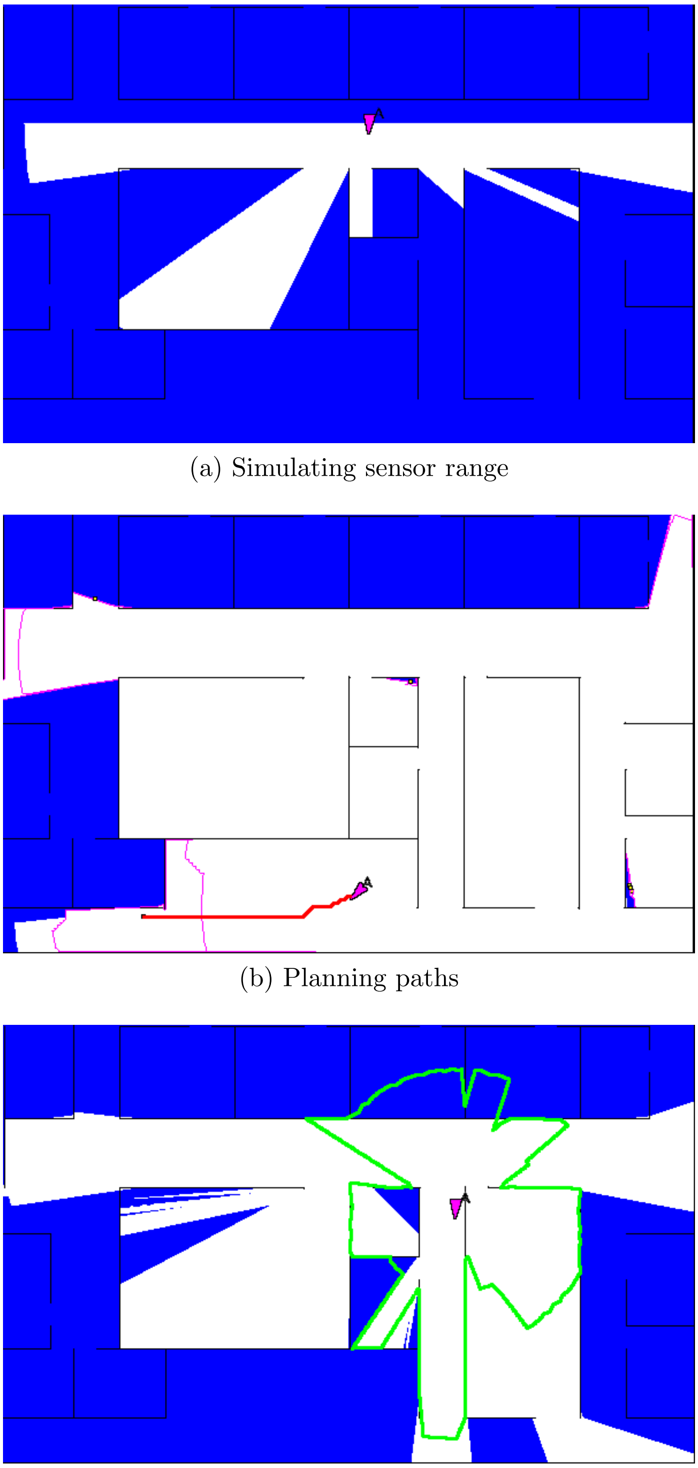
1. There is no sensor noise. In reality, wheel encoders provide inexact data, and laser range finders often have spurious measurements at either 0 or max range. Localisation remains a significant challenge in robotics, even if a number of techniques such as particle filters and scan matching (*e.g.* [85, 65, 48, 113]) show great promise.

2. Environments are two-dimensional and flat. In the real world this is almost never the case.

3. The simulator runs in a single thread in a single process. Real robots would each run their own (multi-thread) processes, and would not always move the same distance in each time segment.

4. Communication in reality is highly variable and very difficult to predict.

Nevertheless, for purposes of quick, simple, controlled, and repeatable comparison of exploration algorithms, MRESim remains a useful tool. This was confirmed by the experiments performed on real robots after simulation results were completed (Chapter 6): results obtained in reality had much in common with the results obtained in simulation.



(a) Simulating sensor range

(b) Planning paths

(c) Simulating communication range

Figure A.2: Demonstrating MRESim

Bibliography

[1] N. Agmon, N. Hazon, and G.A. Kaminka. Constructing spanning trees for efficient multi-robot coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1698–1703, 2006.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, **15**(5):818–828, Oct 1999.

[3] R.C. Arkin and J. Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *7th International Workshop on Advanced Motion Control*, pages 455–461, 2002.

[4] Ronald C. Arkin and Tucker Balch. Cooperative multiagent robotic systems. In *Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, 1998.

[5] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking systems. In *19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, **2**, pages 775–784, March 2000.

[6] Stephen Balakirsky, Stefano Carpin, and Arnoud Visser. Evaluating the RoboCup 2009 Virtual Robot Rescue Competition. In *Proceedings of the Performance Metrics for Intelligent Systems Workshop (PerMIS)*, 2009.

[7] Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, **1**(1):27–52, 1994.

[8] Mitch Barnes, A H. R. Everett, and Pavlo Rudakevych. Throwbot: Design considerations for a man-portable throwable robot. In *SPIE Vol 5804 Unmanned Ground Vehicle Technology VII*, pages 511–520, 2005.

[9] Prithwish Basu and Jason Redi. Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, **18**(4):36–44, 2004.

[10] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. Multi-robot search and rescue: A potential field based approach. *Autonomous Robots and Agents Series: Studies in Computational Intelligence*, **76**:9–16, 2007.

[11] P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended Voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[12] M. Bengel, K. Pfeiffer, B. Graf, A. Bubeck, and A. Verl. Mobile robots for offshore inspection and manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3317–3322, 2009.

[13] Maren Bennewitz and Wolfram Burgard. An experimental comparison of path planning techniques for teams of mobile robots. In *Autonome Mobile Systeme*. Springer-Verlag, 2000.

[14] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, **94**(7):1384–1397, July 2006.

[15] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **2**(1):14–23, 1986.

[16] B. Bruggemann, A. Tiderko, and M. Stilkerieg. Adaptive signal strength prediction based on radio propagation models for improving multi-robot navigation strategies. In *2nd International Conference on Robot Communication and Coordination (ROBOCOMM)*, pages 1 – 6, April 2009.

[17] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **1**, pages 476–481, 2000.

[18] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, **21**(3):376 – 378, 2005.

[19] James Cameron. Terminator 2: Judgement Day. TriStar Pictures, 1991.

[20] Y. Uny Cao, Alex S. Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, **4**(1):7–27, 1997.

[21] S. Carpin, M. Lewis, Jijun Wang, S. Balakirsky, and C. Scrapper. USARSim: a robot simulator for research and education. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405, April 2007.

[22] Stefano Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, **25**:305–316, October 2008.

[23] Jennifer Casper and Dr. Robin Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics*, **33**:367–385, 2003.

[24] A. R. Cassandra. A survey of POMDP applications. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24, 1998.

[25] Kenneth Chang. Obama vows renewed space program, April 2010. `http://www.nytimes.com/2010/04/16/science/space/16nasa.html`.

[26] Airlie Chapman and Salah Sukkarieh. A protocol for decentralized multi-vehicle mapping with limited communication connectivity. In *IEEE In-*

ternational Conference on Robotics and Automation (ICRA), pages 357 –362, may 2009.

[27] HOWIE CHOSET. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, **31**:113 – 126, 2001.

[28] HOWIE CHOSET AND JOEL BURDICK. Sensor-based exploration: The hierarchical generalized Voronoi graph. *The International Journal of Robotics Research*, **19**(2):96–125, 2000.

[29] HOWIE CHOSET, SEAN WALKER, KUNNAYUT EIAMSA-ARD, AND JOEL BURDICK. Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph. *The International Journal of Robotics Research*, **19**(2):126–148, 2000.

[30] J. CORTES, S. MARTINEZ, AND F. BULLO. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, **51**(8):1289 –1298, 2006.

[31] T. DARRELL AND A. PENTLAND. Active gesture recognition using partially observable markov decision processes. In *Proceedings of the 13th International Conference on Pattern Recognition*, **3**, pages 984 –988 vol.3, aug 1996.

[32] A. DAVIDS. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, **17**(2):81–83, March-April 2002.

[33] R. DAVIS AND R. G. SMITH. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, **20**:63–109, 1983.

[34] JULIAN DE HOOG, STEPHEN CAMERON, AND ARNOUD VISSER. Role-based autonomous multi-robot exploration. In *Proceedings of the International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE)*, November 2009.

[35] Julian de Hoog, Stephen Cameron, and Arnoud Visser. Dynamic team hierarchies in communication-limited multi-robot exploration. In *Proceedings of the International Workshop on Safety, Security and Rescue Robotics (SSRR)*, July 2010.

[36] Julian de Hoog, Stephen Cameron, and Arnoud Visser. Selection of rendezvous points for multi-robot exploration in dynamic environments. In *Workshop on Agents in Realtime and Dynamic Environments, International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, May 2010.

[37] M Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems*, pages 115–122, July 2000.

[38] Y.Z. Du, D. Hsu, H. Kurniawati, W.S. Lee, S.C.W. Ong, and S.W. Png. A pomdp approach to robot motion planning under uncertainty. In *International Conference on Automated Planning & Scheduling, Workshop on Solving Real-World POMDP Problems*, 2010.

[39] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, **7**(6):859–865, 1991.

[40] Matthew Dunbabin, Iuliu Vasilescu, Peter Corke, and Daniela Rus. Experiments with cooperative control of underwater robots. *The International Journal of Robotics Research*, **28**:815–833, 2009.

[41] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, **22**:46–57, June 1989.

[42] Joel M. Esposito and Thomas W. Dunbar. Maintaining wireless connectivity constraints for swarms in the presence of obstacles. In *Proceedings of the*

International Conference on Robotics and Automation (ICRA), pages 946–951, 2006.

[43] E. Ferranti and N. Trigoni. Robot-assisted discovery of evacuation routes in emergency scenarios. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2824–2830, May 2008.

[44] E. Ferranti, N. Trigoni, and M. Levene. Brick & mortar: an on-line multi-agent exploration algorithm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 761–767, April 2007.

[45] Amsterdam Oxford Joint Rescue Forces. Online. `http://www.jointrescueforces.eu`.

[46] Jodi Forlizzi and Carl DiSalvo. Service robots in the domestic environment: a study of the Roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265. ACM, 2006.

[47] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, **8**:325–344, 2000.

[48] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, **94**(7):1325–1339, July 2006.

[49] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics*, **14**(2):163 –175, April 2009.

[50] A. Franchi, G. Oriolo, and P. Stegagno. On the solvability of the mutual localization problem with anonymous position measures. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3193 –3199, 2010.

[51] K. Fujimura and K. Singh. Planning cooperative motion for distributed mobile agents. *Journal of Robotics and Mechatronics*, **8**(1):75 – 80, February 1996.

[52] B.P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, and M.J. Mataric. Most valuable player: a robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, **3**, pages 1226 –1231, 2001.

[53] Brian P. Gerkey and Maja J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, **18**(5):758–768, October 2002.

[54] Suzanne Goldenberg. Deepwater Horizon oil spill: Underwater robots trying to seal well. `http://www.guardian.co.uk/environment/2010/apr/26/deepwater-horizon-spill-underwater-robots`, April 2010.

[55] Robert Grabowski, Luis E. Navarro-Serment, Christiaan J. J. Paredis, and Pradeep K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, **8**(3):293–308, 2000.

[56] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. Towards a navigation system for autonomous indoor flying. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[57] Erico Guizzo. Videos of packbot robots inside Fukushima reactors released. `http://spectrum.ieee.org/automaton/robotics/industrial-robots/videos-of-packbots-inside-fukushima-reactors`, April 2011.

[58] Dave Gussow. Robot rescue: These guys go where human searchers can't. `http://www.sptimes.com/2005/09/19/Technology/Robot_rescue__These_g.shtml`, September 2005.

[59] Sabine Hauert, Severin Leven, Jean-Christophe Zufferey, and Dario Floreano. Communication-based Swarming for Flying Robots. In *Proceedings of the Workshop on Network Science and Systems Issues in Multi-Robot Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[60] N. Hazon and G.A. Kaminka. Redundancy, efficiency and robustness in multi-robot coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 735–741, April 2005.

[61] New Zealand Herald. Map: Likely location of trapped miners. `http://www.nzherald.co.nz/nz/news/article.cfm?c_id=1&objectid=10689816`, November 2010.

[62] Jon Herskovitz. Japan a robot power everywhere except at nuclear plant. Reuters, March 2011. `http://www.reuters.com/article/2011/03/17/us-quake-japan-robots-idUSTRE72G2LD20110317`.

[63] C.J. Hilditch. Linear skeletons from square cupboards. *Machine Intelligence*, **4**:403–420, 1969.

[64] D.F. Hougen, S. Benjaafar, J.C. Bonney, J.R. Budenske, M. Dvorak, M. Gini, H. French, D.G. Krantz, P.Y. Li, F. Malver, B. Nelson, N. Papanikolopoulos, P.E. Rybski, S.A. Stoeter, R. Voyles, and K.B. Yesin. A miniature robotic system for reconnaissance and surveillance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **1**, pages 501–507, 2000.

[65] A. Howard. Multi-robot mapping using manifold representations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **4**, pages 4198–4203, 2004.

[66] Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. Experiments with large heterogeneous mobile robot team: Exploration, mapping, de-

ployment and detection. *International Journal of Robotics Research*, **25**(5):431–447, May 2006.

[67] Andrew Howard and Nicholas Roy. The Robotics Data Set Repository (Radish), 2003.

[68] Kaijen Hsiao, L.P. Kaelbling, and T. Lozano-Perez. Grasping pomdps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4685 –4692, april 2007.

[69] Liz Hull. Drone makes first UK 'arrest' as police catch car thief hiding under bushes, February 2010. `http://www.dailymail.co.uk/news/article-1250177/Police-make-arrest-using-unmanned-drone.html`.

[70] Adrian Jiménez-González, J.R. Martínez de Dios, and Anibal Ollero. An integrated testbed for heterogeneous mobile robots and other cooperating objects. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[71] R. Colin Johnson. Robots prove fitness as first responders during Katrina emergency, 3 October 2006. Available at http://www.informationweek.com/news/management/showArticle.jhtml?articleID=171202729.

[72] Miguel Juliá, íscar Reinoso, Arturo Gil, Mónica Ballesta, and Luis Payá. A hybrid solution to the multi-robot integrated exploration problem. *Engineering Applications of Artificial Intelligence*, **23**:473–486, June 2010.

[73] Alexander Kleiner, Johann Prediger, and Bernhard Nebel. RFID technology-based exploration and SLAM for search and rescue. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4054–4059, 2006.

[74] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In

Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), **4**, pages 3232–3238, Oct. 2003.

[75] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[76] Sven Koenig, Boleslaw Szymanski, and Yaxin Liu. Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, **31**:2001, 2001.

[77] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, **2**:83–97, 1955.

[78] Benjamin Kuipers and Yung tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, **8**:47–63, 1991.

[79] Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton Kleywegt, Sven Koenig, Craig Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *In Robotics: Science and Systems*, pages 343–350, 2005.

[80] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(9):869–885, September 1992.

[81] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[82] J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. In *Proceedings of the IEEE Conference on Decision and Control*, **2**, pages 1508 – 1513 Vol.2, 2003.

[83] Thorsten Linder, Viatcheslav Tretyakov, Sebastian Blumenthal, Peter Molitor, Hartmut Surmann, Dirk Holz, Robin Murphy, and Satoshi Tadokoro. Rescue robots at the collapse of the municipal archive of Cologne city: a field report. In *Proceedings of the 8th IEEE International Workshop on Safety, Security, and Rescue Robotics*, July 2010.

[84] Jinguo Liu, Yuechao Wang, Bin Li, and Shugen Ma. Current research, key performances and future development of search and rescue robots. *Frontiers of Mechanical Engineering in China*, **2**:404–416, April 2007.

[85] Feng Lu and Evangelos Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *Journal of Intelligent and Robotic Systems*, **18**:249–275, 1997.

[86] George M. Mathews, Hugh Durrant-Whyte, and Mikhail Prokopenko. Decentralised decision making in heterogeneous teams using anonymous optimisation. *Robotics and Autonomous Systems*, **57**(3):310 – 320, 2009.

[87] G.M. Mathews and H.F. Durrant-Whyte. Decentralised optimal control for reconnaissance. In *Proceedings of the IEEE Conference on Information, Decision and Control (IDC)*, pages 314 –319, 2007.

[88] Fumitoshi Matsuno and Satoshi Tadokoro. Rescue robots and systems in Japan. In *Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics*, 2004.

[89] L. Matthies, Y. Xiong, R. Hogg, D. Zhu, A. Rankin, B. Kennedy, M. Hebert, R. Maclachlan, C. Won, T. Frost, G. Sukhatme, M. McHenry, and S. Goldberg. A portable, autonomous, urban reconnaissance robot. *Robotics and Autonomous Systems*, **40**(2-3):163 – 172, 2002.

[90] D. Meier, C. Stachniss, and W. Burgard. Coordinating multiple robots during exploration under communication with limited bandwidth. In *European Conference on Mobile Robots*, pages 26–31, Ancona, Italy, 2005.

[91] Hada Messia. Rescuers use robot in search for missing balloonists. http://edition.cnn.com/2010/WORLD/europe/10/02/missing.balloonists/index.html?hpt=T2, October 2010.

[92] Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In Jean-Claude Heudin, editor, *Virtual Worlds*, **1434** of *Lecture Notes in Computer Science*, pages 254–263. Springer Berlin / Heidelberg, 1998.

[93] Mark Micire. *Analysis of the Robotic-Assisted Search and Rescue Response to the World Trade Center Disaster*. Master's thesis, CRASAR, University of South Florida, July 2002.

[94] Benoit Morisset, Radu Bogdan Rusu, Aravind Sundaresan, Kris K. Hauser, Motilal Agrawal, Jean-Claude Latombe, and Michael Beetz. Leaving Flatland: Toward real-time 3D navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3786–3793, 2009.

[95] A. Morris, D. Silver, D. Ferguson, and S. Thayer. Towards topological exploration of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2117 – 2123, 2005.

[96] A.R. Mosteo, L. Montano, and M.G. Lagoudakis. Multi-robot routing under limited communication range. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1531–1536, May 2008.

[97] Piyoosh Mukhija, Rahul Sawhney, and K. Madhava Krishna. Multi robotic exploration with communication requirement to a fixed base station. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1515–1516, 2010.

[98] Robin Murphy. Reuters slams Japan for lack of nuclear disaster robots. CRASAR Blog online, March 2011. `http://crasar.org/2011/03/18/reuters-slams-japan-for-lack-of-nuclear-disaster-robots/`.

[99] Robin Murphy. Rikuzen-takada: robots go where divers cannot. `http://crasar.org/2011/04/21/rikuzen-takada-robots-go-where divers-cannot/`, April 2011.

[100] Robin R. Murphy, Jeffery Kravitz, Ken Peligren, James Milward, and Jeff Stanway. Preliminary report: Rescue robot at Crandall Canyon, Utah, mine disaster. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2205–2206, 2008.

[101] Robin R. Murphy and Sam Stover. Rescue robots for mudslides: A descriptive study of the 2005 La Conchita mudslide response: Field reports. *Journal of Field Robotics*, **25**:3–16, January 2008.

[102] Sarfraz Nawaz, Muzammil Hussain, Simon Watson, Niki Trigoni, and Peter N. Green. An underwater robotic network for monitoring nuclear waste storage pools. In *Proceedings of the 1st International ICST Conference on Sensor Systems and Software (SCUBE)*, 2009.

[103] P. Newman, M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **1**, pages 1234 – 1240, September 2003.

[104] P. M. Newman, D. M. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Orlando Florida USA, May 2006.

[105] Hoa G. Nguyen, Narek Pezeshkian, Anoop Gupta, and Nathan Farrington. Maintaining communication link for a robot operating in a hazardous environment. In *Proceedings of the International Conference on Robotics and Remote Systems for Hazardous Environments*, pages 28–31, 2004.

[106] Budget of the United States Government. National Aeronautics and Space Administration. `http://www.gpoaccess.gov/usbudget/fy11/pdf/budget/space.pdf`.

[107] Mars Exploration Rover Mission Online. Looking for signs of past water on Mars. `http://marsrovers.nasa.gov/science/`, May 2011.

[108] Mars Exploration Rover Mission Online. Update: Spirit and Opportunity. `http://marsrovers.jpl.nasa.gov/mission/status.html`, May 2011.

[109] RoboCup Online. `http://www.robocup.org/`.

[110] L.E. Parker. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, **14**(2):220–240, Apr 1998.

[111] Kaustubh Pathak, Andreas Birk, Narūnas Vaškevičius, and Jann Poppinga. Fast registration based on noisy planes with unknown correspondences for 3D mapping. *IEEE Transactions on Robotics*, **26**:424–441, 2010.

[112] M. Pfingsthorn and A. Birk. Efficiently communicating map updates with the pose graph. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[113] Max Pfingsthorn, Bayu Slamet, and Arnoud Visser. A Scalable Hybrid Multi-Robot SLAM method for Highly Detailed Maps. In *Proceedings of the 11th RoboCup International Symposium*, July 2007.

[114] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[115] Matthew Powers and Tucker Balch. Value-based communication preservation for mobile robots. In *7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[116] Steven Rainwater. Robots aid rescue teams in Haiti. `http://robots.net/article/2974.html`, January 2010.

[117] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1340–1345, Nagoya, Japan, August 1997. Morgan Kaufmann.

[118] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, **31**:7–40, 2001.

[119] Martijn N. Rooker and Andreas Birk. Communicative exploration with robot packs. In *RoboCup 2005: Robot Soccer World Cup IX*, **4020** of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 267 – 278. Springer, 2006.

[120] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, **15**(4):435–445, 2007.

[121] Stergios I. Roumeliotis and George A. Bekey. Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*, **18**:781–795, 2002.

[122] Izhak Rubin and Runhe Zhang. Placement of UAVs as communication relays aiding mobile ad hoc wireless networks. In *Proceedings of the Military Communications Conference (MILCOM)*, pages 1 –7, October 2007.

[123] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[124] P.E. Rybski, N.P. Papanikolopoulos, S.A. Stoeter, D.G. Krantz, K.B. Yesin, M. Gini, R. Voyles, D.F. Hougen, B. Nelson, and M.D. Erickson. Enlisting rangers and scouts for reconnaissance and surveillance. *Robotics Automation Magazine, IEEE*, **7**(4):14 –24, dec 2000.

[125] Bruno Santos Pimentel and Mario Fernando Montenegro Campos. Multi-robot exploration with limited-range communication. In *Anais do XIV Congresso Brasileiro de Automatica (CBA'02)*, 2002.

[126] Dave Scheiber. Robots to the rescue. `http://www.sptimes.com/2003/03/02/Floridian/Robots_to_the_rescue.shtml`, 2 March 2003.

[127] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus. From theory to practice: Distributed coverage control experiments with groups of robots. In *Experimental Robotics: The Eleventh International Symposium*, **54**, pages 127–136. Springer-Verlag, 2008.

[128] Mac Schwager, Jean-Jacques Slotine, and Daniela Rus. Decentralized, adaptive control for coverage with networked robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2007.

[129] Antonio Sgorbissa and Ronald C. Arkin. Local navigation strategies for a team of robots. *Robotica*, **21**(5):461–473, 2003.

[130] Weihua Sheng, Qingyan Yang, Song Ci, and Ning Xi. Multi-robot area exploration with limited-range communications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, **2**, pages 1414–1419, 2004.

[131] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Journal of Robotics and Autonomous Systems*, **54**(12):945 – 955, 2006.

[132] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008.

[133] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087. IJCAI, Inc, 1995.

[134] Reid G. Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan L. S. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 852–858. AAAI Press / The MIT Press, 2000.

[135] N. Sorensen and W. Ren. Rendezvous problem in multi-vehicle systems: Information relay and local information based strategies. In *Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems*, 2006.

[136] Cyrill Stachniss, Óscar Martínez Mozos, and Wolfram Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, **52**:205–227, April 2008.

[137] H.W. Stone and G. Edmonds. Hazbot: a hazardous materials emergency response mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 67 –73 vol.1, 1992.

[138] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1525 –1530, 2008.

[139] SUAAVE:. Sensing, Unmanned, Autonomous Aerial VEhicles, `http://www.suaave.org/`.

[140] Satoshi Tadokoro, Fumitoshi Matsuno, Masahiko Onosato, and Hajime Asama. Japan national special project for earthquake disaster mitigation in urban areas. In *First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, 2003.

[141] O. Tekdas, P. Plonski, N. Karnad, and V. Isler. Maintaining connectivity in environments with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[142] G. Theocharous and S. Mahadevan. Approximate planning with hierarchical partially observable markov decision process models for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **2**, pages 1347 –1352, 2002.

[143] S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **3**, pages 4270–4275, Sept. 2003.

[144] S.B. Thrun. Exploration and model building in mobile robot domains. In *IEEE International Conference on Neural Networks*, pages 175–180, 1993.

[145] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hhnel, Michael Montemerlo, Aaron Morris, Zachary Omohundro, Charlie Reverte, and Warren Whittaker. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, **11**:79–91, 2004.

[146] Y. Tojo, P. Debenest, E.F. Fukushima, and S. Hirose. Robotic system for humanitarian demining. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **2**, pages 2025 – 2030, 2004.

[147] J. Vazquez and C. Malcolm. Distributed multirobot exploration maintaining a mobile network. In *Proceedings of the IEEE International Conference on Intelligent Systems*, **3**, pages 113–118, June 2004.

[148] A. Visser and B.A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *International Sympo-*

sium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT), pages 680–687, 2008.

[149] Arnoud Visser, Gideon Emile Maillette de Buy Wenniger, Hanne Nijhuis, Fares Alnajar, Bram Huijten, Maarten van der Velden, Wouter Josemans, Bas Terwijn, Christiaan Walraven, Quang Nguyen, Radoslaw Sobolewski, Helen Flynn, Magda Jankowsk, and Julian de Hoog. Amsterdam Oxford Joint Rescue Forces - Team Description Paper - Virtual Robot competition - Rescue Simulation League - RoboCup 2009. In *Proceedings of the 13th RoboCup International Symposium, Graz, Austria*, 2009.

[150] Arnoud Visser, Quang Nguyen, Bas Terwijn, Moos Hueting, Robrecht Jurriaans, Martijn van der Veen, Okke Formsma, Nick Dijkshoorn, Sander van Noort, Radoslaw Sobolewski, Helen Flynn, Magda Jankowska, Swaroop Rath, and Julian de Hoog. Amsterdam Oxford Joint Rescue Forces - Team Description Paper - Virtual Robot competition - Rescue Simulation League - RoboCup 2010. In *Proceedings of the 14th RoboCup International Symposium*, 2010.

[151] Arnoud Visser, Tijn Schmits, Steven Roebert, and Julian de Hoog. Amsterdam Oxford Joint Rescue Forces - Team Description Paper - Virtual Robot competition - Rescue Simulation League - RoboCup 2008. In *Proceedings of the 12th RoboCup International Symposium, Suzhou, China*, July 2008.

[152] Arnoud Visser and Bayu A. Slamet. Balancing the Information Gain Against the Movement Cost for Multi-robot Frontier Exploration. In *European Robotics Symposium 2008*, Springer Tracts in Advanced Robotics, pages 43–52. Springer-Verlag, February 2008.

[153] Jijun Wang and Stephen Balakirsky. USARSim V3.1.3: A game-based simulation of mobile robots. Available at http://ovh.dl.sourceforge.net/sourceforge/usarsim.

[154] Richard Welch and Gary Edmonds. Applying robotics to HAZMAT. Technical report, Jet Propulsion Laboratory, California Institute of Technology, 2003. Available at http://hdl.handle.net/2014/36468.

[155] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1160–1165, Sept. 2008.

[156] B. Yamauchi, A. Schultz, and W. Adams. Mobile robot exploration and map-building with continuous localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **4**, pages 3715–3720, May 1998.

[157] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the International Conference on Autonomous Agents (AGENTS)*, pages 47 – 53, New York, NY, USA, 1998. ACM.

[158] Zhenwang Yao and Kamal Gupta. Backbone-based connectivity control for mobile networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, ICRA'09, pages 2420–2426, Piscataway, NJ, USA, 2009. IEEE Press.

[159] J. Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, **8**:7–24, 2000. 10.1023/A:1008984701078.

[160] A. Zelinsky. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, **8**(6):707–717, Dec 1992.

[161] X.S. Zhou and S.I. Roumeliotis. Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1785 –1792, 2006.

[162] V.A. Ziparo, A. Kleiner, B. Nebel, and D. Nardi. RFID-based exploration for large robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4606–4613, April 2007.

[163] R. Zlot, A. Stentz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, **3**, pages 3016 – 3023, 2002.