Behaviour Based Simulated Low-Cost Multi-Robot Exploration

Jose Manuel Vazquez Diosdado

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2006

Abstract

The use of multiple robots for exploration holds the promise of improved performance over single robot systems. To exploit effectively the advantage of having several robots, the robots must be co-ordinated which requires communication. Previous research relies on a fixed communication network topology, a single lead explorer, and flat communication. This thesis presents a novel architecture to keep a group of robots as a single connected and adaptable communication network to explore and map the environment. This architecture, BERODE (BEhavioural ROle DEcentralized), aims to be robust, efficient and scalable to large numbers of robots. The network is adaptable, the number of explorers variable, and communications hierarchical (local/global).

The network is kept connected by an MST (Minimum Spanning Tree) control network, a subnetwork containing only the minimum necessary links to be a fully connected network. As the robots explore, the MST control network is updated either partially (local network) or globally to improve signal quality. The local network for a robot is formed by the robots that are within a certain retransmission distance in the MST control network. BERODE implements a hierarchic approach to distributing information to improve scalability with respect to the number of robots. The robots share information at two levels: frequently within their local network and less frequently to the entire robot network.

The robots coordinate by assuming behaviours depending on their connections in the MST control network. The behavioural roles balance between the tasks of exploration and network maintenance where the Explorer role is the most focused on the exploration task. This improves efficiency by allowing varying number of robots to take the Explorer role depending on circumstances. The roles generate reactive plans that ensure the connectivity of the network. These plans are based on the imposition of heterogeneous virtual *spring forces*.

Our simulations show that BERODE is more efficient, scalable and robust with respect to communications than the previous approaches that rely on fixed control networks. BERODE is more efficient because it required less time to build a complete map of the environment than the fixed control networks. BERODE is more scalable because it keeps the robots as a single connected network for more time than the fixed control networks. BERODE is more robust because it has a better success rate at finishing the exploration.

Acknowledgements

First of all, I must thank my supervisor Chris Malcolm for his support and guidance while I undertook this work. I have benefited from our discussions over the past four years. I would also like to thank Bob Fisher for his valuable comments.

I am grateful with Mykel J. Kochenderfer and Ernesto Andrade for their feedback that have help me to improve this work.

I wish to thank my family and friends for their constant support. Finally, I especially thank my dear wife Paloma Flores. She has supported me and encourage me all along the way.

This research has been founded by the Research Council for Science and Technology (CONACYT) of Mexico.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jose Manuel Vazquez Diosdado*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the important goals in robotics is the use of robots in hazardous environments. These scenarios are typically partially or completely unknown. Robots working in these scenarios need to build a representation of the environment as they explore it looking for potential hazards (e.g. mine removal, search and rescue, surveillance, etc.).

In recent years, the use of teams of robots in these scenarios has attracted the interest of the research community. The use of multiple robots has several advantages (Burgard et al., 2002); first, the potential to complete the task faster than a single robot. Furthermore, teams of robots are more fault tolerant than a single robot. Finally, overlapping sensory information can reduce uncertainty and make feasible the use of cheaper and noisier sensors than could be tolerated in a single robot. Moreover, in the last five years there has been a growing interest in the development of distributed sensing systems using wireless networks with low power short communication range and the integration of this with robotic platforms. This thesis presents a novel distributed architecture called BERODE (BEhavioural ROle DEcentralized) for incorporating these new technologies in multi-robot exploration tasks.

Important engineering questions arise in the design of a multirobot architecture. From an engineering view point the architecture should be efficient and scalable to large populations of robots. From a more general viewpoint we shall explore the benefits of the application of short range communication technologies to multi-robot scenarios.

The BERODE architecture has been tested in simulation. To improve the realism of our simulations we have measured relevant aspects of the robot sensors and communication devices. Our experiments (Chapter 7) show that the simulated sensor and communication models are reasonable and conservative approximations to the exper-

imental data obtained from the hardware devices. We have built and validated a *low cost* platform that uses sonar and infrared sensors. Our simulated robots are based on this platform. The simulated robots use inexpensive sensors (e.g. sonars, infrared) and have a low power short range communication system.

We tested the BERODE architecture in simulation for *RF* (Radio Frequency) and *LOS* technologies. Most of the current wireless technologies (e.g. Ethernet, Bluetooth) are based on *RF* technologies. The simulated *LOS* and *RF* communication models model the delays caused by retransmissions in the *MANET* (Mobile *ad hoc* network) and the effect of interference. The simulated *RF* model allows some transparency in certain obstacles (e.g. desks, wooden boxes, etc). Part of the signal is absorbed by the obstacles. The attenuation of the *RF* signal for different materials was obtained from experiments with a Bluetooth hardware device and from tables provided by manufacturers of wireless cards at 2.45 GHz (MaxStream, 2003). The effects of multi path reflections are modelled by adding Gaussian Noise (with mean zero) when there is no *LOS* between transmitter and receiver. For radio communication, this model is more realistic than line of sight, but still conservative; e.g. it predicts more limited communication than in general will occur in reality. In the simulated *LOS* model any obstacle in the direct path of the signal blocks the entire signal. The *LOS* implementation is suitable for infrared communication. Several small robot platforms (e.g. swarmbots (McLurkin and Smiths, 2004)) implement this type of technology to exchange information. For this reason, we consider it was important to test our architecture for *LOS* technologies as well as *RF*.

This introductory chapter gives a perspective of the presented work relative to the field of multi-robot coordination. Section 1.1 discusses the benefits of having cooperative robots. Section 1.2 presents the motivation for extending existing research in the area of multirobot exploration. Section 1.3 defines formally the problem of exploring an initially unknown environment using a group of robots with short communication range devices. Section 1.4 presents a general overview of the proposed BERODE architecture to explore environments in a decentralized fashion (Chapter 4 presents the approach in detail).

Section 1.5 presents the goals of this research. The main goal is to present a decentralized architecture for exploration tasks using teams of mobile robots with local communication capabilities which is efficient, robust, potentially scalable to large populations of robots, and suitable for implementation in *low cost* robots. The BERODE architecture is based on behavioural roles that reactively adapt to the dynamic con-

ditions of the communication network formed by the robots. Section 1.6 presents a summary of the contributions of this thesis. The main contribution is the adaptive control network used in the BERODE architecture. Previous approaches rely on *a priori* fixed leader-follower control relations (fixed control networks) whereas in BERODE the control network relations between pairs of robots are modified over time. Our comparisons in simulation of BERODE with previous fixed control networks showed BERODE to be more efficient, scalable and robust with respect to communications. Section 1.7 presents the structure of the thesis.

1.1 The Benefits of Cooperative Robots

The field of cooperative robotics is a relatively new research area, with origins in the 1980s, when researchers began to investigate issues in multiple robot systems (Parker, 2000). Prior to this time, research had concentrated on single robot systems with a variety of sensors (Yamauchi et al., 1998; Thrun et al., 1998b).

The research in multiple robots naturally extends research on single robot systems, but is a distinct topic in its own right due to the complexity of multiple robot interaction and new constraints on robot behaviour derived from this. Distributed systems such as chemical plants or nuclear reactors deal with the real world, but operate under well-known constraints. Multiple robot systems are more complex than these systems because the robots may have to navigate in a largely unknown and often unpredictably dynamic world.

Multiple robots cooperating hold the promise of improved performance and fault tolerance for large-scale problems. Nevertheless, it is difficult to compare the performance of multi-robot architectures because, as is so often the case in robotics, the hardware or the experimental scenario is different. Even more important is the fact that the concept of performance is ambiguous and depends on the particular viewpoint of the designer. Due to the recency of the area, a formal architecture or general principles has not yet been established and all the research done so far is state of art and designed for some specific domain.

However, the researchers have been able to analyze the benefits of multi-robot teams in specific aspects of the task; for instance Jennings et al. (1994) analyzed the equivalence of different types of sensing, communication and amount of communication needed to cooperate in a tightly coupled task.

One of the main goals of our research is to achieve the efficient exploration of

the environment using a team of robots. For this reason in our research the efficiency is measured in terms of the time that the robots require to build a complete map of the environment. In our architecture each robot builds its own feature map of the environment. The robots' maps incorporate observations from all the robots. The feature map is projected into a probabilistic grid map to find unexplored areas. An unexplored area is an area for which there no evidence. The map is considered as completed when the size of the unexplored areas is below a user defined threshold. The exploration process stops once any one robot in the network considers that its map is complete. Communications ensures that the general topology of the robots' maps is the same; however the maps usually have small metric differences due to such things as rounding errors.

1.2 Motivation of the Research

There has been some research in recent years to develop efficient strategies for exploration purposes; most of the approaches are centralized architectures, obtaining solutions close to the optimal; some other approaches are distributed, obtaining suboptimal solutions, but with the advantages of robustness and flexibility.

Efficient strategies minimize task overlapping which can only be achieved by means of coordination. To coordinate a team of robots, implicit or explicit communication is required. Implicit communication occurs through sensing of the world and is usually a side effect of other actions. Explicit communication occurs directly, usually through a wireless medium (e.g. radio frequency, infrared). For instance Mataric (1998) has shown the advantages of explicit communication in improving group behaviour in multi-robot learning.

Previous research in coordination has assumed that robots are able to communicate either with a central system (centralized coordination) or with the rest of the robots in the team at any time in any location without taking in account the range of the signal. In real environments it is important to consider signal limitations. There are advantages of economy in power and size in low power communications. For instance, low power short range communications can ameliorate the interference problem.

The research on multi-robot teams has been concentrated on small teams of robots; another important aspect that has been ignored in exploration tasks is the fact that global communication may not be suitable for larger teams of robots. As the number of robots in a team increases the communication bandwidth becomes a bottleneck for

the system.

In recent years researchers have been developing new robot platforms like RoboMote (Sibley et al., 2002) and Millibots (Navarro-Serment et al., 1999) which are intended to have characteristics of low power consumption, basic sensing capabilities (infrared), small communication range, very small size (47 cm^3), and very low cost (approximately 150 per unit).

One of the main purposes of these platforms is to form Mobile *ad hoc* Robot Networks (Antonelli et al., 2005). An *ad hoc* robot network has to remain connected to achieve coordination between the robots actions.

The main motivation of this research is that our exploration architecture should be suitable for implementation in *low cost* robots (e.g. Millibots). In BERODE the robots are kept as a single adaptable communication network to minimize task overlapping and improve coordination. The control relations between pairs of robots are modified over time rather than relying in *a priori* fixed leader-follower control relations as proposed in previous approaches.

Our comparative simulations suggest that BERODE in practical implementations will be robust to infrequent communication, scalable in terms of communication and number of robots, and will explore the environment more efficiently than networks that rely on fixed control networks.

1.3 Exploration Efficiency for a Robot Network

To explore their environment and to coordinate their actions, the robots need to build a map of the environment as they traverse it. This is a very important aspect in the exploration task since mapping is constantly interleaved with decision making of where to move next. To map an environment, a robot has to cope with two types of sensor noise: Noise in perception (e.g., range measurements), and noise in odometry (e.g., wheel encoders). Because of the latter, the problem of mapping creates an inherent localization problem, which is the problem of determining the location of a robot relative to its own map. This problem is known as the Simultaneous Localization and Mapping (*SLAM*) problem. In this problem the robots build a map and obtain estimates of their location in this map. Chapter 3 presents a discussion about the different approaches to solve this problem and discusses their applicability for *low cost* robots. From this discussion it is concluded that an Extended Kalman Filter (*EKF-SLAM*) approach is the most suitable approach for *low cost* sensor platforms. The *EKF* uses a feature representation of the

environment.

Many strategies can be used to explore an environment using a group of robots. The robots can be kept as a single group forming a robot network, they can be allowed to split into subgroups of variable sizes or each robot can act independently (independent robots).

In previous research in exploration architectures (Section 2.2, page 20) two different approaches have been taken to achieve exploration efficiency. In the first approach the robots are kept within communication range to achieve the coordinated exploration of the environment. The robots' maps integrate the observations from all the robots. In the second approach the robots explore the environment independently and exchange maps when they are within communication range of each other. The efficiency is achieved through the dispersion of the independent robots in the environment. The robots try to move away from the other robots. To insure that the robots integrate the information from other robots about their maps only once they have to keep a history of the exchanged information for each robot.

Unfortunately to the best of our knowledge there is no research that compares the exploration efficiency of these two approaches. There are only comparisons about the benefits of coordination when the communication is assumed to be global (Burgard et al., 2002).

In a decentralized architecture each robot builds its own map. A team of robots builds similar[1] maps when they each integrate each other's measurements only once to their maps. Maps that integrate the same observations (similar maps) are less likely to contain inconsistencies in their general topology. Map inconsistencies can be removed by means of additional exploration to obtain observations that disambiguate the inconsistent information.

A group of independent robots can build similar maps by keeping a history of the observations from all the robots in the team. The robots update their information about the observations from all the robots when they become within communication range of another robot. The robots use the new observations to update their map estimates.

Robots that remain as a single connected network can build similar maps by exchanging their observed features periodically. A robot does not have to keep a history of the observations from all the robots in the team. Moreover Losada (Rodriguez-Losada, 2004b) showed that the appearance of errors in the topology of the map is

[1] Similar maps are maps that have the same general topology and contain only small metric differences.

related to the maximum uncertainty in orientation that the robot has at some point in the exploration. The errors in the topology of the map are errors such as the failure to detect open doors. These errors can prevent a robot from exploring an area or in the worst case to return to the initial location. Therefore it is important to keep uncertainty as low as possible all the time. The uncertainty in the robot position increases when a robot traverses areas which contain few features. The detection and removal of topologic errors can be achieved by means of additional exploration.

Robots that move as a single connected network keep their uncertainty lower than independent robots in areas with few features because they periodically receive the observations from all the robots in the team. The uncertainty for independent robots rises when they are unconnected and in areas with a few features.

In the BERODE architecture we decided to keep the robots in a single network because:

- Exploration efficiency can be achieved through the minimization of exploration overlapping. Although robots that act independently could potentially explore the environment more efficiently they are also more likely to explore areas which have been already explored by other robots.
- The robots can build similar maps without having to keep an observation history from all the robots in the network. This is not the case for independent robots.
- The maps are less likely to contain topologic errors because the uncertainty of the robots' orientation is kept lower throughout the exploration than for independent robots.

In terms of exploration efficiency the strategies of keeping a robot network and having independent robots are good solutions. Nonetheless there are practical benefits in keeping a robot network such as the integration of observations from other robots is less expensive in terms of storage and the maps are less likely to contain topologic errors. This is very important for an exploration architecture that is designed to be implemented on large numbers of *low cost* robots such as BERODE.

1.4 Problem Statement and Proposed Solution

The problem is to explore and map an initially unknown environment efficiently using a group of robots which have local communication capabilities and maintain an *ad hoc*

network.

In our problem we have assumed that the robots know their initial relative positions. This was assumed because our architecture contemplates a scenario in which the network of robots is deployed from a single drop off point. This assumption avoids the problem of map merging with relative unknown positions which currently is an ongoing research area.

Our approach tries to improve exploration efficiency through coordination by minimizing task overlapping. We propose to generate coordinated behaviours for the group of robots that allow them to explore the environment while keeping the communication network as a single connected network. The exploratory behaviour of the network emerges from the interaction of the individual behaviours of the robots. The main ideas of our architecture are:

1. Each robot builds and updates its own feature map representation of the environment. The robots' maps incorporate the features observed by all the robots in the network. The feature map is updated using an Extended Kalman Filter (*EKF*) (Smith et al., 1988). The *EKF* produces an estimate (along with its uncertainty) of the position of the features and the robot. The positions of robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[2] number. The robots extract features from their sensor measurements. The extracted features are used to update the feature map. The robots periodically transmit their feature observations. Robots incorporate feature observations of other robots with the same process as for locally extracted features because they share the same global Cartesian frame of reference.

2. The robots are kept as a single adaptable communication network. The communication network is kept connected by building and updating a MST (minimum spanning tree) control network (Cormen et al., 1990). The communication network is kept connected by imposing virtual forces for the connections in the MST control network. The virtual forces are modelled as heterogeneous spring forces. We describe the springs as heterogeneous because they are asymmetric[3] and their free spring length[4] is a range of values rather than a single value.

[2] The robots have an ID number that allows them to identify each other in the network.

[3] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[4] The free spring length is the length for which the spring exerts a null force.

The free spring length is a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection.

3. The robots adopt behavioural roles according to their internal state and their connection state in the MST. The behavioural roles balance between the tasks of exploration and network maintenance.

4. The Robots are attracted to unexplored areas of the environment. Each exploring robot projects its feature map into a probabilistic grid map to find unexplored areas. An unexplored area is an area for which there no evidence. The attractiveness of an unexplored area is a function of its size, its distance, and the predicted communication quality. According to its behavioural role, a robot exhibits some interest or none in the exploration task. As a result a number of robots in the network direct the exploration towards unexplored areas while the rest of the robots ensure the maintenance of communication.

The robots have to operate under the following conditions:

1. **Unknown Environment:** The workspace is unknown *a priori*.

2. **Limited Communication:** The robots have limited communication capabilities.

3. **Limited Communication Bandwidth:** The robots have to share information efficiently to reduce the bandwidth required.

4. **Limited Sensing:** The robots are equipped with *low cost* sensing devices.

1.5 The design of the BERODE Architecture

The BERODE architecture is motivated by the findings of previous related research discussed in Chapter 2. Chapter 3 describes the design of the BERODE architecture in terms of team behaviour. Chapter 4 presents the implementation of the BERODE architecture in individual robots. In the initial simulations we found problems related to occasional shortages of exploring robots (Section 8.6, page 239). A second implementation of the architecture called BERODE-2 incorporated role scheduling mechanisms that addressed these problems (Section 8.7, page 246). This section presents an overview of the general design of the BERODE architecture which is the same for the two implementations.

BERODE is based on behavioural roles such as Explorer and communication Maintainer. These roles reactively adapt to the dynamic conditions of the communication network formed by the robots as they explore an environment. The communication network is maintained as a fully connected network by creating and updating an MST (Minimum Spanning Tree) control network (Cormen et al., 1990). The MST control network is a subnetwork of the communication network containing only the minimum necessary links to have a fully connected network. Thus the robots do not need to try to maintain unnecessary communication links.

In BERODE each robot builds and updates its own feature map representation of the environment using an Extended Kalman Filter (*EKF*) (Smith et al., 1988). The *EKF* produces an estimate (along with its uncertainty) of the position of the features and the robot. The positions of the robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[5] number. The robots extract features from their sensor measurements. The extracted features are used to update the feature map.

The robots periodically transmit beacon signals and their observed features. The beacon signals contain the estimated position of the robot and its uncertainty. The robots use the beacon signals to determine the signal quality. How this is to be measured depends on the hardware implementation. For instance, in the implementation for *RF* technologies we use the *RSSL* (Received Signal Strength Level) which is an available value for this technology (measured in dB) as the signal quality value.

Robots incorporate the features extracted by other robots in the network to their maps using the same process as for locally extracted features because they share same global Cartesian frame of reference.

The MST control network is built at the beginning of the exploration and modified throughout the exploration to improve signal quality. The robots select their behavioural role according to their internal state and their connection state in the MST control network. The interaction between the behavioural roles is achieved through the imposition of virtual forces derived from the connections in the MST control network

The robots exhibit one of the following behavioural roles: Explorer, Maintainer, Pusher and Recoverer. The Explorer and Maintainer behavioural roles are the basic behaviours for the robot in the network. The Explorer robots focus on exploration while the Maintainer robots focus on keeping the communication network connected. Pusher robots help the exploration task when it becomes problematic because opposing

[5] The robots have an ID number that allows them to identify each other in the network.

exploratory directions halt the global exploration process. Recoverer robots help the maintenance task when it becomes problematic because of poor connection quality.

The non-Explorer robots keep the communication network connected by generating plans to move to locations where the energy from the virtual spring forces is minimized. The magnitude and sign of the spring forces is a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection.

The Explorer robots are not directly subject to virtual spring forces; instead these robots monitor the level of safety for their control connection and wait for improvement if necessary to avoid the risk of becoming disconnected. The Explorer robots guide the exploration process by generating a plan to move to the safest unexplored area in terms of communication quality and moving in that direction to the limits of their communication safety.

Robots reselect their behavioural role once an event has occurred. An event occurs when the robot has reached its current goal or modified its local network. When a robot modifies its local network this network is transmitted to the robots inside the local network. The robots inside the local network update their local network and reselect their behavioural roles.

The heterogeneous spring forces disperse the robot network in the exploratory directions when there are no Recoverer robots and contract the robot network when there are Recoverer robots.

The main components of the BERODE approach are briefly introduced in the following subsections; Chapter 4 contains the detailed description.

1.5.1 The Adaptive MST Control Network

Robots with local communication capabilities form mobile *ad hoc* communication networks in which the robots are nodes in the network and can act as routers to relay information through the network; the network is dynamic. In BERODE the *ad hoc* communication network is kept fully connected by creating and updating an MST control network. The MST control network is calculated at the start of the exploration, based on the communication network and signal quality criterion. The signal quality criterion depends on the implementation; for instance for typical radio frequency (*RF*) technologies the received signal strength level (*RSSL*) between a pair of robots is the signal quality. The signal quality is used as the link cost in the calculation of the MST control network. After the initial calculation the robots retain knowledge of their lo-

cal network. The local network for a robot is the network that contains all the robots within a *k-hop* distance. The effect of the size of the local network on the performance of BERODE is investigated in Chapters 8 and 9.

The MST control network can be modified or rebuilt by the robots throughout the exploration. Merging and validation mechanisms are implemented to ensure that the robots maintain the consistency of the MST control network. Robots periodically re-evaluate their local network to improve signal quality; if necessary they modify the local network and inform all the robots within the local network. Additionally, the MST control network is used by the robots to distribute their information among team members (Section 5.10, page 130). Robots periodically transmit beacon signals to their direct connections. The robots determine their safety level using the measured signal quality from the received beacon signals. BERODE implements three levels of safety with the following decreasing order in safety: safe, precautionary and unsafe. A robot is on a specific safety level if all of its connections have at least that safety level.

1.5.2 Goals for the Behavioural Roles

The goals for each behavioural role are:

Behavioural Roles

1. Explorer: Move towards unexplored areas while maintaining a safe connection.
2. Maintainer: Move towards the location that minimizes the spring forces for the control connections.
3. Recoverer: Move towards a location that minimizes the spring forces for the unsafe control connections. An unsafe control connection is a connection with a signal quality below the unsafe threshold (Section 4.3, page 63).
4. Pusher: Move towards the location that minimizes the spring forces for the control connection.

The goal for the Maintainer, Pusher and Recoverer behavioural roles is essentially the same. The difference is in the parameterization of the free spring length values (Explorers are not affected by virtual spring forces). The free spring length for a spring connection is a range of values that depends on the behavioural roles of the robots that form the connection and the quality of the connection. The parameters are set up in such way that the interaction between the roles generates *leader–follower* motions in

the robot network where the Explorers direct the exploration and the Pushers accelerate the movement of the robot network in the exploration directions.

The effect of an unsafe connection on the global behaviour is the contraction of the network until the unsafe connection recovers a high enough quality level to be detected as precautionary or safe, at which point the robots continue the exploration. When a pair of robots detects a connection as unsafe they backtrack their most recent movements to return to a position where the level of safety is not unsafe. If the connection remains as unsafe the robots transition to the Recoverer role and generate a plan to move towards each other's last received positions.

1.5.3 Hierarchical Information Distribution

To achieve coordination and build consistent[6] maps the robots periodically distribute their observed features to the rest of the team. In BERODE each robot is in charge of distributing its information. One of the main goals in BERODE is scalability. To get a good trade-off between coordination and communication costs BERODE implements hierarchical approach for information distribution.

When the robots finish the exploration of the environment they should by then have *consistent* feature maps of the environment. However, we argue that it is not necessary that the robots share consistent feature maps during the whole exploration to achieve the coordinated exploration of the environment. Instead it is only necessary that the robots have commoned[7] feature maps to achieve coordinated behaviours. Robots that are close (based on their *k-hop* distance) need to have more in common between their feature maps than robots that are distant because close robots need to achieve coordinated motion to avoid disconnections. This is not the case for distant robots.

In large networks most robots are distant and communication savings can be made by having less frequent communication between distant robots. This is achieved in BERODE with a hierarchic approach of two levels: local and global. The local level for a robot is formed by the robots inside the local network. The global level is formed by all the robots in the network. We examine the communication cost in Chapter 9 for different local/global ratios.

The features that a robot extracts from its sensor measurements are stored for their

[6] Consistent maps are maps that have integrated the same information about observed features. These maps usually have small metric differences, but the general topology of the map is the same.

[7] Commoned maps are maps that have integrated a common subset of information about observed features.

periodical distribution in a local and a global store. The local and global stores store the same information about the extracted features. The features in the local store are distributed at the local level. The features in the global store are distributed at the global level. The distribution at the local level is more frequent than at the global level to improve scalability with respect to the numbers of robots.

The features that the robot receives from others at the global level are integrated to the robots' maps using the same process as for locally extracted features. This is possible because the robots share the same global Cartesian frame of reference. The features that a robot receives at the local level are useful navigational information for the local robots (robot inside the same local network), but are not used to update the map because that would cause a double update.

The hierarchic distribution of features allows a reduction in the communication cost without sacrificing too much exploration efficiency.

The robots distribute other information such as a change in their role on an event basis. However most of the communication cost is due to periodical transmission of the beacon signals and feature distribution. Most of the communication is retransmitted only at the local level avoiding scalability problems as the number of robots increases.

1.6 Thesis Goals

The overall goal of this thesis is to present and assess BERODE, a decentralized architecture for exploration tasks using teams of mobile robots with local communication capabilities. BERODE is designed to be efficient, robust, potentially scalable to large populations of robots, and suitable for implementation in *low cost* robots (e.g. Millibots). The efficiency of BERODE is assessed in terms of the time that the robots require to build a complete map of the environment.

The two most important questions in assessing the usefulness and applicability of BERODE are as follows:

1. How efficient is a team of robots constrained to maintaining local communication?

2. What are the trade-offs between the number of robots and the communication costs?

With respect to the first question research in distributed exploration systems has tried to keep robots apart in order to avoid exploration overlapping while assuming that

global communication is available. For local short range communication the design of the behavioural roles establishes a balance between the exploration and network maintenance tasks. The exploratory behaviour of the robot network emerges from the interaction of the individual behavioural roles.

Previous research (Burgard et al., 2002) in multi-robot exploration suggests that there is a logarithmic decrease in the exploration time with increasing numbers of robots. In Burgard's architecture a central agent generated plans for all the robots. In his experiments the communication was assumed to be global. The scenarios tested were open spaces and structured environments (office like environments). Unfortunately for robot networks of more than a few robots with local communication, the performance in realistic conditions is unknown. For instance, Burgard et al. (2006) limited his experiments to three robots. They conducted simulations for more robots (up to 20 robots) but their communication model assumed no interference from the obstacles or environmental structure (Burgard et al., 2006).

One of the important aspects to achieve multirobot coordination is the communication range. Although having robots with an effectively unlimited communication range may seem in principle attractive it is undesirable for the purposes of scalability. The bandwidth required increases as more robots are added to the network. Moreover when power consumption is a constraint in the system it is desirable to restrict communication to short distances because the amount of power required for transmitting a signal in free space is a function of the square of the distance. In indoor environments the situation is much worse because of the structure and the obstacles.

We are particularly interested in determining the performance of BERODE in indoor environments because they are cluttered and contain structures (rooms, corridors) inside that complicate the navigation of the robots and block communication. Our simulations therefore incorporate typical measured aspects of communication (e.g. signal absorbed by obstacles). Our research is interested in providing more insight about the performance of the teams of robots with local communication by using simulation models that better approximate the characteristics of signal propagation in indoor environments.

With respect to the second question, in our BERODE architecture the coordination of the robot network is achieved by means of the exchange of information. To get a good trade-off between coordination and communication cost we propose the hierarchical distribution of information at two levels: local and global. Information is distributed more frequently at the local level than at the global level to reduce the

communication cost while keeping exploration efficiency.

To assess the scalability and robustness of BERODE with respect to communication we analyse the effect on communication cost of varying the size of the local level and the frequency of the distribution at the local and global levels.

1.7 Contributions

As explained in Section 1.4 we developed two implementations of the BERODE architecture. The second implementation of BERODE (called BERODE-2) was used to assess the properties of the proposed architecture.

The main contribution of this thesis is the description of a new architecture for multi-robot exploration that is more efficient, scalable and robust with respect to communications than the previous fixed control networks (Section 9.6, page 295). In our comparisons in simulation of BERODE-2 with previous fixed control networks we found that BERODE took less time to build a complete map of the environment, maintained communications better and failed less often.

In our simulations from Section (9.4, page 277) we found that the hierarchical (local/global) communication approach scales better to larger numbers of robots, than non-hierarchic communications (Section 9.3, page 270 and Section 9.4.1, page 286). When non-hierarchic communications are used in large robots networks ($n \geq 40$) there is an accelerated increase in the communication bandwidth. When the ratio between local and global networks is small, there is a linear increase in the bandwidth with respect to the number of robots. More over, when communications are only local, low power communications can be used with benefits to power consumption and interference. In our simulations (Section 9.5, page 289) with the *RF* model we identified a communication range threshold above which the exploration efficiency ceases to improve for a robot network. This finding is important because communication can be restricted to short distances to reduce power consumption without decreasing the exploration efficiency.

Our simulations confirm the findings of Fenwick et al.'s (2002) that the uncertainty can be reduced by combining information from multiple robots. This confirms our suggestion that large teams of expendable *low cost* robots can be used to produce maps which are useful not only for the robots own navigation, but for human use.

We can't conclude that these properties and findings would necessarily transfer to a real world implementation, but we have taken steps (Chapter 7) to make our sim-

ulations as reasonable and conservative in the relevant aspects. Section 11.5 (page 352) discusses details on how the BERODE architecture can be implemented using real robots. A more detailed assessment of the contributions can be found in the Conclusion chapter (page 347).

1.8 Structure

This thesis consists of eleven chapters:

- Chapter 1 presents the problem of decentralized coordination, the proposed approach, the objectives of the thesis and its contributions.
- Chapter 2 presents a review of the state of the art in multi-robot exploration and sensor networks, discusses their integration in distributed multi-robot architecture, and outlines the problems in terms of computational and communication costs.
- Chapter 3 presents a review of map building approaches and discusses their suitability for BERODE. The suitability of an approach is decided on computational and economic cost that allows the potential scalability to large populations of robots. An *EKF* approach is found to be the most suitable approach for BERODE.
- Chapter 4 presents the general design of the BERODE architecture in terms of what the team of robots is expected to accomplish. Several algorithms to calculate the MST control network are proposed. The robots adopt a behavioural role according to their internal state and their connection state in the MST control network. The exploratory behaviour of the robot network emerges from the interaction of the individual behaviours of the robots.
- Chapter 5 presents the implementation of the BERODE architecture in individual robots. The architecture has been implemented using modules that are sequentially executed. The implementation and parameterization of some of the modules depends on the behavioural role.
- Chapter 6 presents the Map Building Module. The environment is represented as a set of line and point features. The module implements an *EKF* to estimate

the position and uncertainty of the robots and the features. Models for the robot and the features are described.

- Chapter 7 presents the simulator that we use to test BERODE and describes the *LOS* and *RF* communication models used in our simulations. Experiments that show that the simulated sensors and communication devices are reasonable and conservative approximations to the experimental data are presented. The implementation of the Map Building Module for a *low cost* platform that uses infrared and sonar sensors is presented. Investigations to assess the *low cost* platform in a real environment are presented and analyzed.

- Chapter 8 presents the implementation of the approach for *LOS* and *RF* technologies. Initial simulations revealed poor performance in particular conditions because the exploration temporarily ceased. An improved mechanism for scheduling exploration was added. The simulations show that the improved version explores the environments faster than the original architecture.

- Chapter 9 presents simulations that analyze the robustness, the scalability and the efficiency of BERODE. The simulations show that BERODE is robust to infrequent communication, is scalable in terms of communication and number of robots, and explores the environment more efficiently than networks with fixed control networks.

- Chapter 10 presents simulations that analyze the consistency of the maps built by the robots in BERODE. The simulations show that as more robots are added to the network the maps become more consistent at the feature, metric and topologic levels of the map. A scalability problem with respect to the amount of map features in the current implementation is discovered and a solution is investigated.

- Chapter 11 presents the conclusions and contributions of thesis. Future extensions addressing the limitations of the current implementation are described.

Chapter 2

Multi-Robot Exploration Architectures

2.1 Introduction

This chapter presents a review of research relevant to this thesis in the areas of multi-robot exploration, mobile sensor networks, and communication protocols for mobile *ad hoc* networks.

In recent years several Multi-robot architectures for exploration purposes have been proposed. These architectures are usually classified as centralized and decentralized (Section 2.2). Centralized architectures are suitable for a small number of robots. Decentralized architectures are flexible, scalable and robust, but frequently achieve this at the cost of suboptimal solutions compared to those of centralized approaches (Burgard et al., 2002).

Centralized architectures assume global communication while decentralized architectures assume local communication and can be classified as insensitive or sensitive (Section 2.2.2). Communication insensitive architectures allow robots to explore the space on their own, while communication sensitive architectures try to maintain the robots within communication range.

A team of robots maintaining communication range form a mobile *ad hoc* communication network. The emergent area of mobile sensor networks shares the same objective as that of mobile *ad hoc* networks; to maintain all its members within communication range. Their difference is that mobile sensor networks frequently rely on a fixed *a priori* placed wireless communication structure or create one. Frequently the objective of the mobile sensor networks is the deployment of a communication infrastructure for service purposes, search and rescue and static deployment. Section 2.3 presents a review of relevant applications in this area and discusses their similarities

with the exploration task.

Mobile sensor networks are based on decentralized control methods designed to maintain groups of robots within some range by means of simple local interactions. A variety of approaches such as rigid formations, *virtual physics* and potential fields have been proposed. Section 2.4 presents these approaches and discusses their applicability for the proposed approach.

In a decentralized approach communication between robots is necessary to achieve coordination, minimizing task overlapping. In a decentralized approach each robot is in charge of distributing its information. Within the robot network, information is distributed to the point where all the robots share a common model of the environment. Section 2.5 presents a review of communication protocols for mobile *ad hoc* networks.

The objective of the BERODE (BEhavioural ROle DEcentralized) architecture is to build a common representation of the environment. Section 2.6 introduces the types of map representations commonly used in multi-robot exploration. The applicability of the representations to multi-robot scenarios is briefly discussed in this section. Chapter 3 provides a more detailed discussion about the map representations and their applicability to multi-robot scenarios using *low cost* robots. The localization problem within these maps in the presence of uncertainty in the sensors is also discussed.

The chapter ends with a summary of the current state of the art in these areas and the expected contributions of the approach to these areas.

2.2 Exploration Architectures

The problem of exploration of an unknown environment has been extensively studied; first using single robot systems with a variety of sensors (Moravec and Elfes, 1985; Shatkay and Kaebling, 1997; Yamauchi, 1998) and later using teams of robots. The first implementations of multi-robot exploration systems were simple extensions of the single robot implementations (Yamauchi et al., 1998).

Multi-robot exploration extends the research on single robot systems, but it is also a topic in itself, because of the complexity of dealing with multiple robots and new constraints. Multiple robot systems are more complex than other distributed systems because they have to deal with uncertainties of a real environment.

Multi-robot systems are usually classified as centralized and decentralized; centralized systems obtain solutions close to the optimal[1] but are computationally intensive

[1] For exploration tasks optimality is defined in terms of the time to completely explore an environ-

and have a single point of failure; on the other hand, decentralized systems are flexible and robust, but frequently achieve considerably sub optimal solutions compared to those of centralized systems (Burgard et al., 2002). Centralized systems require a global communication system, while decentralized exploration has been implemented in systems with local communication.

A review of relevant centralized and decentralized exploration architectures is presented in the following subsections.

2.2.1 Centralized approaches

Thrun et al. (2000) designed an algorithm using a Monte Carlo localizer[2] and maximum likelihood on grid maps. The maximum likelihood function determines the best alignment of laser scanned data. A team leader merges the maps and shares the grid map with the rest of the team. In the experiments the team consisted of two members.

Simmons et al. (2000) implemented an algorithm to merge maps and coordinate the robots using a central unit. The algorithm tries to maximize the overall utility by minimizing the potential overlapping of information. Each robot creates a local map, and sends the information to a central mapper which improves the map by iteratively combining data from the robots. The global map is used to decide where the robots should explore in the next time step. The experiments were developed using a team of three robots.

Burgard et al. (2002) presented a similar approach to Simmons' approach. In Burgard's approach the central system computes the frontier cells based on a utility value for each location. The utility value is determined by the expected travel cost and the information gain. The information gain is the estimated number of unknown map cells within a radius at the location. When one robot is assigned to some location, the information gain of the location cells is decreased. Based on this approach Solanas and Garcia (2004) proposed a clustering technique that allows faster partial coverage of the environment.

Dias (Dias and Stentz, 2001; Dias et al., 2002) proposed a market-based approach. In this approach each robot establishes a set of goals in areas where little information is known; based on these goals each robot produces a tour containing several goals. The tours are auctioned and the robots submit bids based on the distance that they

ment.

[2] A Monte Carlo localizer represents the belief by a set of samples, drawn according to the posterior distribution over robot poses.

have to travel to cover the tour. The tours are refined through continuous inter-robot negotiation. Robots periodically send their recently explored area to a central agent. The central agent merges the areas and creates a global map that is sent to the robots. Burgard et al. (2006) discovered that frequently a robot would generate a tour for itself because it was individually the best suited bidder for each area. This could lock out other robots and lead to an unbalanced assignment of tasks and an increase in the exploration time.

Centralized approaches try to disperse the robots throughout the environment to achieve efficiency; robots stay apart minimizing interference (Simmons et al., 2000). Dispersion is suitable for small teams when global communication is available. In real environments it is important to consider signal limitations; robots might be unable to communicate with each other if they disperse without regard to their communication range.

In all these approaches robots frequently send grid maps to a central agent, relying strongly on the communication system and having a single point of failure. Moreover, as the number of robots in a team increases the communication bandwidth becomes a bottleneck for the system.

2.2.2 Distributed approaches

Previous research in distributed approaches for exploration can be classified as communication insensitive and sensitive. Communication sensitive approaches are those where the robots try to remain in communication range with at least one robot during all the exploration process. Communication insensitive approaches are those where robots autonomously explore the environment and environmental features are exchanged when communication is possible.

2.2.2.1 Communication Insensitive

In Yamauchi's (1998) approach, robots exchange their grid maps and continuously update their own map, merging the map received with their local maps. The approach uses frontier-based exploration to direct robots to the areas that are likely to provide the most new information about the world. A minimal degree of coordination is achieved since robots only exchange their maps. Multiple robots end up moving to the same frontiers, which is highly inefficient.

Konolige et al. (2004) proposed an approach where the robots initially explore the

environment by themselves. When two or more robots come in range of each other (form an exploration cluster), their maps are merged and a leader robot emerges. The leader coordinates all the robots, builds a complete map that represents the data collected by all the robots, and broadcasts the map frequently to all the robots in range. When a robot loses communication with its current exploration cluster it keeps exploring the space in an autonomous fashion. Konolige remarks that maps are built less efficiently when robots can not communicate.

These authors remark that the benefit of having multiple robots to explore an environment is maximized if they can coordinate; to achieve coordination robots have to be able to communicate.

2.2.2.2 Communication Sensitive

Communication sensitive approaches where robots methodically sweep the unknown environment by advancing in close line formations have been proposed by Singh and Fujimura (1993), Latimer et al. (2002), Min and Min (1998) and Reikleitis et al. (2004). Formations are broken and restored when obstacles are encountered. Strategies where robots keep track of their absolute positions through mutual observation are proposed by Rekleitis et al. (2000) and Grabowski et al. (1999). While some robots move, their relative coordinates are calculated by other stationary robots that observe them.

All of these approaches require very tight coordination among all the robots and have been tested only in environments with small obstacles or free space environments. Moreover, methodical sweep techniques are not suitable for structured environments. Structured environments are full of potential traps for these techniques.

Arkin and Diaz (2002) developed algorithms to maintain line of sight communications between robots while searching for a hazard with different degrees of *a priori* knowledge of the environment. One robot remains static serving as a base for the rest of the robots. The exploration area is limited by the total communication range. Robots move one at a time. As a consequence having several robots to explore an area does not decrease significantly the time required to find the hazard.

The research of Antonelli et al. (2005), and Pimentel and Montenegro (2002) is similar to that of Arkin and Diaz but the robots move simultaneously, and a simple idealised *RF* communication model is assumed. In the approach of Pimentel and Montenegro the robots coordinate by means of predicting other robots locations, while in Antonelli et al.'s approach an expected signal is predicted for close locations. The envi-

ronments in both cases contain only a few randomly placed obstacles and the explored area is limited by the total communication range. Moreover, the idealised *RF* communication model assumption is strong and unsuitable for real indoor environments which are complex *RF* environments with *hot spots*[3], interference, etc. In our simulations we implement a more realistic communication model which as been validated through experimentation with *RF* communication devices (Section 7.3.3, page 169).

2.3 Mobile Sensor Networks

In the last five years there has been a growing interest in developing distributed sensing systems using local wireless networks with a local range of communication. The main characteristics of these devices are low power consumption, size, and cost which make them particularly suitable for larger numbers of cheap robots.

Recent platforms such as Millibots (Navarro-Serment et al., 2002) and Robomote (Sibley et al., 2002) are envisioned as mobile sensor networks. These new platforms focus on the formation of cooperative large teams and are meant to be deployed in hazardous unknown scenarios.

Robomote is a test platform for mobile sensing using local communication (Sibley et al., 2002). Based on this platform, algorithms for incremental sensor deployment have been developed (Howard et al., 2002a; Howard et al., 2002b). The issue of exploration has not been addressed yet, but it is one of the final goals of the project.

Research on the Millibots project has been oriented to the cooperative localization of the robots using a beacon system (Grabowski et al., 1999). The robots explore the environment following a leap frog sequence to maintain good position estimates while exploring the environment.

Mobile Sensor Networks have been implemented for two types of communication: Radio Frequency and Line of Sight. Typically in Radio Frequency technologies systems the *RSSL* (Received Signal Strength Level) is available, while in Line of Sight technologies it is not available.

Research on radio frequency communication models has been addressed by Wagner and Arkin (2003), Powers and Balch (2004), Thibodeau et al. (2004), Ulam and Arkin (2004), and Kantor et al. (2003).

[3] A *hot spot* is referred to as a local place where the signal strength is considerably larger than at its surroundings.

Wagner and Arkin (2003) developed algorithms to cover an area. He identified trade-offs between area coverage and communication safety concluding that coverage degrades when plans are communication-focused. Powers and Balch (2004) proposed a navigation behaviour denominated VBCP (Value-Based Communication Preservation) to preserve communication while traversing an environment. VBCP calculates movement vectors using the *RSSL* (Received Signal Strength Level) from the robots, the robots' positions and map-based predictions of the *RSSL* for nearby positions to the robot. Powers and Balch's experiments on real outdoor scenarios do not entirely demonstrate that it is feasible to use the *RSSL* to maintain communication for teams of robots. In indoor scenarios *RF* signals may create *hot spots* due to reflections, temporary disturbance, problems which Powers did not consider. Ulam and Arkin (2004) proposed a reactive approach to recover communication in a surveillance mission. Several recovery strategies were proposed. He concluded that there are still remaining issues to determine the best strategies to recover communications in large-scale teams. Thibodeau et al. (2004) developed an adaptive topology algorithm for an exploration task where a leader robot frequently builds an MST control network. Based on this network *leader-follower* relations are imposed on all the robots with the exception of the leader. As a result the leader directs the exploration task while the rest of the team follows it. Thibodeau et al. compared his approach against fixed configuration topologies (e.g. chain, fixed tree). The comparison was based on the time to build a complete map. Thibodeau et al.'s approach performs better than fixed configurations. Kantor et al.'s (2003) work focuses on surveillance where navigation is achieved by relying on a *RF* network deployed *a priori*. Navigation was successfully achieved but the authors remarked that the minimal density of sensors required to achieve optimal navigation was unclear.

Research on line of sight (*LOS*) communication models has been addressed by Nguyen et al. (2004), Pezeshkian et al. (2003), Verma et al. (2002), Carpin and Parker (2002), Kannan et al. (2003) and Sweeney et al. (2002) where *a priori* leader-follower relations are established. The team of robots forms a column formation. These approaches are focused on surveillance missions where one robot serves as a base. The leader robot is in charge of building the environmental representation. All of the previous *LOS* approaches have been implemented with real robots and tested in real environments excepting the work of Sweeney et al. In the experiments the teams of robots were successfully maintained within communication range.

Anderson et al. (2003) developed an algorithm to maintain the *LOS* communica-

tions for a team of robots. The algorithm plans movements for all the robots. The drawback of this algorithm is that it is centralized and assumes that an *a priori* map is available.

Fixed leader-follower relations among team members are not desirable because the robot network tends to maintain a particular fixed topology; moreover, according to Thibodeau's research (*op. cit.*) fixed topologies are less robust to robot failures than adaptive topologies. To the best of our knowledge, all the research where robots act as mobile sensor networks implements *a priori* control relations between pairs of robots.

Although communication sensitive architectures have been successfully implemented, with the exception of Thibodeau's approach (*op. cit.*) all these architectures rely on robot leader-follower relations defined *a priori*. Thibodeau's approach performs better than fixed *a priori* leader-follower relations. Thibodeau argued that *a priori* relations over constrain the mobility of the network. Thibodeau's research was carried out in large open spaces with small obstacles. We believe Thibodeau's argument against *a priori* relations will hold in real indoor environments; moreover it is very likely that the constraining effect of *a priori* communication relations will be much worse.

For such reasons, this work proposes that the relations between team members should be determined in a dynamic fashion to gain flexibility and more efficiently achieve their common goal of exploration. A pair of robots has a relation if they have to remain within communication range. Robots only have to maintain certain relations to ensure that the robot network remains fully connected, and these relations can change.

As in Thibodeau's approach (*op. cit.*) this thesis proposes the calculation of an MST (Minimum Spanning Tree) control network. Based on the MST control network relations among pairs of robots are determined. The MST control network is calculated based on the communication network and a signal quality criterion. For instance for radio frequency communication systems the *RSSL* (Received Signal Strength Level) is used as the signal quality. The MST control network dynamically changes as robots traverse the environment.

In the previous research on network maintenance there is only one robot that executes the desired task (leader) while the rest of the team (followers) maintain the network. This type of approach is suitable for surveillance missions or convoy transportation but is highly inefficient for exploration purposes. The benefits of having multiple robots to complete the exploration are reduced. Robots end up following the

leader without collaborating in the exploration task. For this reason, this thesis proposes a behavioural role approach where more than one leader may be active at the same time. The proposed behavioural roles are Explorer, Maintainer, Recoverer and Pusher. The Explorer role has as its main task the exploration of unknown areas, referred to in previous research as the leader. The Maintainer role has as its main task the maintenance of the network, referred to in previous research as the follower. The Pusher behavioural role helps the exploration task when it becomes problematic; for instance when two Explorer robots pull the robot network in opposing exploratory directions the exploratory behaviour of the robot network is halted. One of the Explorer robots gives up its exploration task and transitions to the Pusher behavioural role. The Pusher robot then induces movement in the robot network by pushing its direct connections away from it. The Recoverer robot helps the communications maintenance task when it becomes problematic. A robot exhibits this role when the level of safety is unsafe. The Recoverer role tries to improve the signal quality for its unsafe connections. A robot determines its behavioural role based on its number of connections on the MST control network and its internal state.

2.4 Control in Mobile Robot Networks

A variety of approaches such as rigid formations, *virtual physics* and potential fields have been proposed to create global behaviour in a group of mobile robots by means of local interactions among the robots.

Fredslund and Mataric (2002a; 2000b), Desai et al. (1998) proposed approaches where robots use other robot(s) as reference and establish a rigid formation. A rigid formation is a formation in which robots have predetermined neighbours and use them as references to determine their movements. Robots use only local sensing and minimal communication to maintain a geometric shape such as lines, columns, and diamonds. A robot is designated as the leader and follows preplanned paths while the others maintain their relative positions.

Distributed control based on *virtual physics* emulates physical forces such as electric charges, springs, and gravitational forces among robots. Howard et al. (2002a) modelled robots as electric charges to generate the uniform deployment of the robots into an unknown enclosed area. Gordon et al. (1999) proposed a gravitational force model where robots repulse at close range. McLurkin used a partially connected graph with a spring model to produce uniform deployment within a limited indoor environ-

ment (McLurkin and Smith, 2004). Shucker and Bennett (2005) extended McLurkin's research adding an acute-angle[4] test to achieve a faster dispersion of the robots in the environment.

The use of social potential fields to achieve distributed control was introduced by Reif and Wang (1995). Previously potential fields were used for planning purposes. Potential fields are inspired by collective animal behaviours such as flocking. The concept underlying potential fields is that an agent is influenced by its immediate neighbours. Vector forces represent the influence exerted by an agent's neighbours. Forces are either attractive or repulsive depending on the distance between the agents. Yamaguchi (1997) generated formations based on potential fields. Balch and Arkin (1998) developed a behaviour based architecture based on the integration of potential fields and navigational behaviours to control formations. Dudenhoeffer and Jones (2000) added the concept of neutral zone to eliminate the backlash effect when a robot executes other behaviours while keeping an appropriate distance from the rest of the group.

Research on distributed control has demonstrated the feasibility of controlling robots using local information. Rigid formations successfully control the robots' relative positions. This approach is not adequate to keep the robots within communication range in an exploration task because the maintenance of rigid relative positions tends to impair the exploration process.

The basic idea behind *virtual physics* and potential fields is the same. In *virtual physics* a certain physics model is implemented, while in social potential fields the attractive/repulsive forces are user determined functions. These approaches are a useful tool that combined with navigational behaviours maintains robots connected without constraining them to achieve rigid relative positions. Robots maintaining an *ad hoc* network while moving throughout the space face a similar problem to that of static deployment due to the maintenance of certain neighbourhood relations. With the exceptions of McLurkin (*op. cit.*), and Shucker and Bennett (2005) in all the previous approaches a robot exerts a force on all the robots that are within its influence range; this over constrains the system and frequently leads to deadlocks.

In *virtual physics* approaches the force model emulates a certain physical force. In potential fields the attraction/repulse force models are user determined. We propose the use of a *virtual physics* approach based on the simulation of *spring forces*. This model

[4] The acute-angle test adds the link between robots A and B to the connectivity graph if for all the neighbours C of A the interior angle $\angle ACB$ is acute.

has been proved to converge to a stationary state (Volpe and Khosla, 1990). This is not the case with user determined functions. For this reason this research implements a *virtual physics* model based on *spring forces*. In previous research the free spring length of the *virtual springs* was the same for all the *springs*. Our model implements heterogeneous spring forces. We describe the spring forces as heterogeneous because they are asymmetric[5] and their free spring length[6] is a range of values rather than a single value. The free spring length is a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection.

This thesis proposes an approach similar to McLurkin's (McLurkin and Smith, 2004). In McLurkin's work on partial graphs pairs of robots exert forces on each other only when their connection is part of the partial graph. The use of partial graphs avoids over constraining. We propose the use of an MST control network. The MST control network is a partial graph of the communication network that contains the minimum necessary links to have a fully connected network. Only the connections that are part of the MST control network exert forces.

Moreover, in the previous approaches robots exert homogeneous forces between each other. This research proposes the implementation of heterogeneous forces that depend on the behavioural roles (explained in the previous section) exhibited by the pairs of robots. We argue that the use of heterogeneous forces aids the exploration process. For instance it could be desirable to maintain an Explorer robot at a closer distance or as explained in Section 1.5 (page 9) a Pusher robot can be used to accelerate the movement in the exploration directions. This can be easily achieved in a decentralized fashion with heterogeneous forces.

To the best of our knowledge there is no previous research that implements such heterogeneous forces. Heterogeneous forces generate a *leader–follower* motion effect where robots in the Explorer behavioural role direct the exploration, and the robots in the Pusher behaviour role accelerate the movement in the exploration directions.

2.5 Communication in Mobile AD HOC Networks

Communication is an important feature in multi-robot systems. Communication is necessary to achieve coordination, minimizing task overlapping. To coordinate a team of

[5] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[6] The free spring length is the length for which the spring exerts a null force.

robots implicit or explicit communication is required. Implicit communication occurs through sensing of the world, and is usually a side effect of other actions. Explicit communication occurs directly, usually through a wireless medium (e.g. radio frequency, infrared). Research (Mataric, 1998) has shown the advantages of using explicit communication to improve group behaviour in multi-robot applications.

In previous research it is frequently assumed that global communication among all the robots is available; this assumption in real world environments is unrealistic. Moreover, global communication may not be suitable for large teams of robots, because as the number of robots in a team increases the communication bandwidth becomes a bottleneck for the system.

This research uses local communication systems to coordinate the team of robots. The robots form a wireless multi-hop *ad hoc* network as they explore the environment. These networks where the locations of the nodes (robots) changes over time have dynamic topologies and are referred as *MANETs* (Mobile *ad hoc* Network). Each node in a *MANET* functions as both a host and a router, and the control of the network is distributed among the nodes.

Routing protocols for *MANETs* are classified as *table driven* (Perkins and Bhagwat, 1994) and *source initiated on demand driven* (Perkins and Royer, 1999). In the *table driven* approaches each node of a network maintains a table that encodes the network topology. Messages contain a destination node identifier as part of the message. When a node receives a message, it forwards the message to the preferred neighbour for its destination. The forwarding process continues until the message reaches its destination. The manner in which routing tables are constructed, maintained and updated differs from one routing method to another. Popular routing methods, however, attempt to achieve the common objective of routing messages along the optimal path. Nodes periodically communicate to update the table as the topology changes. In the *source initiated on demand driven* algorithms, routes are found as they are required. Each time a message needs to be sent a new route has to be discovered. The route discovery process has two stages: a forward path stage and a reverse path stage. These stages involve several queries and reply messages to establish the route. This query-reply process is high in terms of communication bandwidth. *Source initiated on demand driven* protocols are therefore suitable for sporadic communication.

In the proposed decentralized approach robots maintain their direct connections by periodically sending beacon signals. To achieve coordination robots have to exchange information. Each robot in the decentralized approach is in charge of distributing its

information. A *source initiated on demand driven* protocol is not suitable for our approach because the robots have to exchange information frequently. A *Table driven* routing algorithm is the most adequate way to distribute the information in our approach.

Researchers in the area of *MANETs* have proposed several routing algorithms based on *MSTs*. For instance, Cai et al. (2005) proposed a 1-*hop* leveraging scheme that reduces unnecessary broadcast transmissions for *MSTs*. The scheme partitions each host's radio broadcast coverage. Based on the partition each node can determine on its own whether or not to forward a message. A node does not transmit the message when its 1-*hop* neighbours can receive the same message from other nodes.

Although in this thesis the *Table driven* routing protocol is not implemented, for realism in the simulations information from *Table driven* routing algorithms about the delays, error rates, etc. was taken into account.

2.6 Building Maps with Multiple Robots

Exploration architectures build a representation of the environment as they traverse it. Two types of representations are used, grid maps and feature based maps. All of the reviewed approaches build grid maps for coordination purposes. Grid maps are convenient to coordinate robots by distributing them among frontier cells. A frontier cell is defined as an unexplored cell adjacent to a free cell. Frontier cells are evaluated according to the number of unexplored cells in the surrounding area, their distance to the current robot position, and in this work the predicted signal quality based on the current robot positions.

Probabilistic grid maps in real world implementations are often built using laser sensors, and simulations often assume laser sensing. In probabilistic grid maps the cells are assumed to be independent. Unfortunately probabilistic grid maps are not suitable for *low cost* sensor platforms because the assumption of independence between cells is less likely to hold. For instance, a sonar measurement covers an entire cone in the grid map representation, and from a single sonar measurement it is impossible to determine the location of the object in the sonar cone. A single sonar measurement is then associated with a much larger number of cells than in the case of a laser measurement. Laser sensors are expensive and cannot be carried by small robotic platforms. Moreover, the frequent exchange of grid maps among large teams of robots is impractical due to the high communication cost.

Because of their geometric representation feature maps require less storage space and have a lower communication cost than grid maps; moreover, feature maps have been implemented in several *low cost* sensor platforms (Tardos et al., 2002b; Leonard et al., 2002; Wijk and Christensen, 2000).

This research does not intend to propose a new solution for the localization problem in a team of robots; however the assessment of the effects of uncertainty in robot locations on the coordination process is important to validate the algorithm in real world scenarios. This problem where the robot has to concurrently deal with uncertainty in its movements and its sensing has been the focus of extensive research and is referred as simultaneous localization and mapping (*SLAM*).

Chapter 3 describes the state of the art in *SLAM* approaches and discusses their applicability to *low cost* sensor platforms. From this discussion it is concluded that an Extended Kalman Filter (*EKF-SLAM*) approach is the most suitable approach for *low cost* sensor platforms. The approach has been previously implemented in *low cost* platforms and its consistency and convergence properties have been proved. This thesis implements an *EKF-SLAM* approach for the map building management module of the robots. The robots build a feature map for localization purposes and project this map into a grid for planning and coordination purposes. Chapter 6 presents the implemented map building management module.

2.7 Conclusions

In recent years several centralized and decentralized approaches to explore unknown environments using teams of robots have been proposed. Centralized approaches are based on global communication to achieve coordination. These approaches have several disadvantages (e.g. single point of failure, scalability problems). Decentralized approaches rely on local communication. These approaches are either sensitive or insensitive to communication maintenance. In communication insensitive approaches robots tend to overlap tasks which is inefficient. In communication sensitive approaches robots try to maintain communication links between pairs of robots. Pairs are established *a priori* and remain static in the exploration process. *A priori* relations tend to over constrain the mobility of the network slowing down the exploration task. This work proposes a novel approach for exploring environments based on a communication sensitive distributed approach. We have conducted simulations to test our approach. To improve the realism of the simulations, models for the robot's sensors, odometry,

and communication were derived from experiments with hardware devices. Chapter 7 presents these models and their experimental validation with real measurements from the hardware devices. The approach tries to improve the *efficiency* of distributed approaches, where *efficiency* is the time to build a complete feature map representation. A feature map is considered to be complete once any one robot has projected it into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a used defined threshold (Section 8.6, page 239).

The proposed solution consists of the imposition of dynamic control links between pairs of robots. Control links are communication links that pairs of simulated robots have to maintain. These links form the MST control network. The MST control network is calculated based on the existing communication links and a signal quality criterion. It was hoped that dynamic control links would give more freedom to the mobility of the simulated robots and the exploration task would be achieved more efficiently. Our simulations suggest that this is the case.

Decentralized control is based on the local interactions among robots to generate global behaviour. Based on existing research in this area, we propose a novel *virtual forces* approach to maintain the control links within communication range. The approach weights the forces exerted on each other by pairs of simulated robots according to their behavioural roles. Similar to McLurkin and Smith's (2004) work pairs of simulated robots exert forces on each other only when their connection is part of the MST control network.

In our approach each simulated robot has to distribute its information. The MST serves as the basis for the routing protocol to distribute information. The protocol allows the distribution of information in an intelligent localised fashion reducing the amount of broadcasted information.

The simulated robots build a feature based representation of the environment. An *EKF-SLAM* approach is used to build the representation. The simulated robots have a Map Building Module to integrate local and external features. Local features are extracted from raw sensor data. External features are the extracted features that a simulated robot receives from the other simulated robots in the network.

The main contributions of the work to the reviewed areas are:

- The dynamic assignation of control links among pairs of robots for network maintenance purposes. Most of the previous approaches rely on *a priori* control relations to keep the network fully connected. The use of dynamic control links

allows the improvement of the communication network with respect to signal quality and local geography.

- The novel *virtual forces* approach for the MST control network. The MST control network is a subnetwork of the communication network. Most of the previous approaches impose *virtual forces* on the entire existing communication network thus over constraining robot movement.

- The heterogeneous forces model. The forces are described as heterogeneous because they are asymmetric[7] and their free spring length[8] is a range of values rather than a single value. The forces depend on signal quality and the behavioural roles of the pair of robots that form the communication link. Heterogeneous forces aid the exploration process because they induce a movement effect towards the unexplored areas of the environment.

[7] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[8] The free spring length is the length for which the spring exerts a null force.

Chapter 3

Building Maps in Multi-Robot Architectures

3.1 Introduction

Over the past two decades, there has been a flurry of work on map building by mobile robots. The problem of building a map is that as the robot moves through the environment it has to concurrently build the representation (map) and localize itself within this representation. This problem is commonly referred as Simultaneous Localization and Mapping (*SLAM*). To solve the *SLAM* problem a robot has to obtain information from sensors. In a typical *SLAM* problem the robot starts with an empty map and the position of the robot as it moves is defined with respect to the initial robot position. The difficulty of the problem is to cope with the growing uncertainty due to the noise in the sensors and approximate mathematical models to estimate the position of the robot and the features as the robot moves around the environment.

This chapter reviews the current state of the art in solving the *SLAM* problem. First the *SLAM* problem is introduced and its characteristics are identified (Section 3.2). Afterwards a description of the types of maps used in *SLAM* approaches is presented (Section 3.3).

In recent years several approaches to solve the *SLAM* problem have been proposed and have achieved exceptional results for single robots. All of these approaches are based on probabilistic representations and some of them have been extended for multi-robot scenarios. *EKF* (Extended Kalman Filters), *EM* (Expectation Maximization) and Scan-Matching approaches are the most commonly used. Section 3.4 describes the fundamentals of these approaches and discusses their advantages and disadvantages.

This thesis implements a *SLAM* algorithm in the Map Building Module. The suitability of an approach for a particular implementation depends mainly on the quality and range of the sensors, the computational cost, and the type of environment. In this thesis we propose the use *low cost* robots for the exploration task. *Low cost* robots are robots that use inexpensive sensors such as sonar and infrared sensors. It is argued that larger teams of *low cost* robots may be a more suitable solution for some exploration tasks than small teams of expensive robots because they are more expendable.

Section 3.5 presents a discussion of the map building approaches implemented in multi-robot systems. In the case of multi-robot scenarios it is important to determine what type of information is shared and its communication cost. The suitability of a particular approach is based on its implementation in *low cost* robots. The suitability is based on the current proofs of convergence, consistency of the approach as well as the communication costs incurred when the maps or parts of them are exchanged between the teams of robots.

Section 3.6 presents a discussion about *low cost* sensors for mapping purposes. Section 3.7 presents a review of implemented mapping systems based on *low cost* sensors. Section 3.8 ends with a summary and discussion of the current state of the art in *SLAM* using *low cost* sensors. The particular difficulties with *low cost* sensors are noise and lack of precision, which render some mapping and localisation approaches unfeasible which are successful with better and more expensive sensors.

3.2 The Problem of Building a Map

The problem of building a map as the robot moves through the environment is a concurrent estimation process commonly referred as *SLAM*. The difficulty of the problem is to cope with the noise in the sensors (external and internal), approximate mathematical models and the unknown robot position.

To acquire a map a robot has to have external sensors. External sensors give information that is relative to the external environment. For instance the scanning laser sensor measures the travelled distance of reflected light beams. By using this information it is possible to measure the distance and orientation with respect to external objects. The most commonly used sensors are radar, sonar, laser, compass, GPS and video camera. Internal sensors are not referred to the external environment; they are referenced with respect to a local coordinate frame system. Internal sensors provide measurements related to the state: acceleration, speed and incremental distance. The

most commonly used are: accelerometers, speedometers, odometers, and steering sensors. Internal sensors are not essential but their use aids the map building process. Information from internal sensors is integrated by means of a kinematic model. The kinematic model is used to predict changes in the robot position. Usually the kinematic model is a simplified approximate model.

In the process of building a map the robot starts from an initial known position. The robot moves through the environment obtaining sensor measurements at certain positions. The map is incrementally built at each position. The map built with data from noisy external sensors is inexact. The robot estimates its position as it moves, and this estimation is an approximation. As new measurements are obtained, the estimation of these measurements is a function of the uncertainties in the position of the robot and the measurements. Because of this the robot localisation and the process of building the map are coupled problems. Therefore, the goal in the *SLAM* problem is to process the sensor data to produce an estimate of the trajectory of the robot while concurrently building the map of the environment.

The main difficulties in building a map without regard to the approach used are:

1. Concurrency: The problems of robot localization and map building have to be solved concurrently. A *SLAM* algorithm has to concurrently minimize the uncertainty in the robot location and the map.

2. The environment size: Small size environments such as a small office or a room can be mapped easily. The accumulated uncertainty is small and the size of the map is not an issue in terms of computational cost. Large size environments such as buildings, museums, etc. are difficult to map. The further the robot moves the larger the uncertainty becomes. Additionally the computational cost increases as the size of the map increases.

3. The dynamic nature of the environments: Most of the approaches assume that the environment is static. Real environments are dynamic.

4. The revisiting problem: The decision of whether or not a robot has, after it has travelled an arbitrary distance, returned to a previously explored area is referred as the revisiting problem. This is a difficult problem for large size environments. All the approaches fail to solve this problem in a general principled way. Most of the approaches implement special purpose routines to solve this problem.

3.3 Types of Maps

A map is a representation that describes the environment in terms of the type of objects that can be detected by the external sensors and that are appropriate for the purpose of localization. The map has to contain at least the *landmarks* required for localization purposes. A *landmark* is an object identifiable by the sensors. Many objects can exist in environments but they can be excluded from the map representation if they are not useful for localization purposes.

Chatila and Laumond's work (1985) was one of the first successful approaches to build accurate maps of the environment, but the most important legacy from this research is the definition of the three possible levels of maps: metric, topologic and semantic.

Metric maps are the most common type of representation adopted and the only one implemented in real world multi-robot scenarios. Metric maps represent numerically the coordinates and properties of the robot and objects inside the environment. Metric maps are represented by features or probabilistic grids. In a feature based map the objects are represented using a geometric model. Walls are represented as line segments; corners are represented as points, etc. The feature map contains the coordinates of the features and the robot with respect to a frame of reference. It is common in map building to use the initial position of the robot as the frame of reference. In grid maps the space is divided in cells. Each cell has a value that represents the probability of being occupied. There is a trade-off between the size of the cells and the complexity of managing large numbers of cells. Small size cells improve the quality of the map but are not practical for localisation because of their computational cost.

Topologic maps represent relations between *landmarks* without using metric information. Common *landmarks* in topologic maps are corridors, rooms, corners, etc. Topologic maps are usually represented with Voronoi graphs (Choset and Nagatani, 2001). Approaches using metric maps are vulnerable to inaccuracies, even when all the relationships between features and the robot itself are taken into account. Topological approaches handle this problem better because they only have to maintain topological global consistency, not metric. Their main drawback is the difficulty of extracting the complex features represented on these maps.

Semantic maps discard all the metric information and represent the map by means of a graph of significant places and their relationships.

Topologic and semantic maps are suitable for high level task planning, such as

multi task scenarios. Moreover, these types of representations can be extracted from metric maps (Thrun, 1998b). To the best of our knowledge, a *SLAM* approach based purely on topologic or semantic maps in real world scenarios has not been implemented. For this reason in our research we implemented a feature based map which is a metric map that contains the positions of the features with respect to a frame of reference. In our implementation the frame of reference is the initial position of the robot. In the simulations with multiple robots the simulated robots share a common global coordinate system based on the initial position of the simulated robot with the lowest ID number. The relative positions of the other simulated robots are known a priori with a small uncertainty.

Most of the approaches to build maps implementing metric representations are used mainly because they are easy to understand for humans. Their representation is the richest, facilitating their construction and their usefulness for robust navigation for the robots. Moreover, most if not all of the multi-robot architectures employ metric representations.

3.4 Stochastic SLAM

As previously defined the problem of building a map is a concurrent estimation problem. For this reason most of the *SLAM* approaches are based on probabilistic representations. These approaches have been the most successful. The main approaches are: Scan-Matching, Expectation Maximisation (*EM*) and Extended Kalman Filter (*EKF*). The following subsections present a discussion of the most relevant works for these approaches.

3.4.1 Scan-Matching

Scan-matching algorithms are based on the alignment of neighbouring sensor scans, e.g. from a laser scanner, to estimate the relative translations and rotations of the robot between scans. The matching process aligns the overlapping segments of the scan set by minimizing some distance metric between inter-scan primitives or raw data. Most of the scan-matching approaches are derived from the *Iterative Closest Point* algorithm (Besl and McKay, 1992). This algorithm iteratively refines an initial robot position estimate obtained through odometry, which limits the search space. It is assumed that the displacement between the initial estimate and the robot's true position is small

enough to obtain the optimal solution.

Scan-matching approaches differ in the primitives selected for the matching process, the distance metric and the weighting of correspondences. Lu's approach successfully matched points to points in an *a priori* unknown environment (Lu and Milios, 1997). The approach is robust because it does not rely on the uniqueness of landmarks but it is computationally expensive since it processes raw data directly. Lee et al. (2002) among others matched lines to lines. The approach is suitable for structured environments and is computationally efficient. Gutmann et al. (2001) combined the two previous approaches taking advantage of the robustness of Lu's approach and the efficiency of Lee's approach. Additionally, Guttmann's matching process takes in account the topological relationships between neighbouring positions, associated by odometry, to maintain global consistency.

Another type of scan-matching approach is based on finding statistical correlations between scans. Biber (2003) proposed a *normal distribution transform*, and Weiss and von Puttkamer (1995) used histograms. These approaches rely on the chosen statistical criteria rather than on correspondences between individual scan elements. The approaches were successfully applied for indoor environments, but the authors remarked that it was not clear that everything can be modelled well with this approach.

There are some hybrid approaches that combine scan-matching with other *SLAM* techniques but they are not suitable for online mapping. For an instance, Haehnel et al. (2003) combined *FastSLAM* (Montemerlo and Thrun, 2003) with scan-matching to minimize odometry error. The approach is based on the random creation of particles (Monte Carlo approach). Each particle has an associated map. The matching process is carried out for each map. The approach obtains very accurate maps but is not suitable for online mapping because of the time required to solve the *SLAM* problem for each particle.

The main drawback of these approaches is their computational complexity and their susceptibility to getting stuck in local minima. If the search space does not contain a solution at least close to the optimal the method fails to build a consistent map. For instance, when the robot revisits previously mapped areas the matching process fails to identify this situation and an inconsistent map is built. Moreover, the properties of convergence and consistency of this approach have not been proved.

Although scan-matching approaches have been successfully implemented with laser scanners and vision systems, they have not been implemented for sonar. We argue that a scan-matching approach alone is not suitable for *low cost* sensors (e.g.

sonar, infrared) because these sensors do not provide enough information about the surroundings. A hybrid approach that combines local maps and a scan-matching approach could be used instead. Local maps are created by storing sensor measurements from several close positions. Local maps are then matched using a scan-matching approach.

3.4.2 Expectation Maximisation

The Expectation Maximisation (*EM*) approach estimates the most likely map, along with the most likely path taken by the robot. The algorithm contains two stages: Expectation and Maximization. In the expectation stage the most likely path for a given map is calculated using a hill climbing technique. In the Maximisation stage the most likely map for the obtained path at the Expectation stage is calculated. At this stage the algorithm generates a sequence of maps with monotonically increasing likelihood until a local maximum is reached. Due to the high dimensionality of the space, commonly the maps are represented by probabilistic grid maps where the cells are assumed to be independent.

This algorithm does not require the unique identification of *landmarks*; thus, there is no data association problem. Data association is performed through gradual reinforcement or degradation of matching probabilities because all the data is considered in the iterative process. This algorithm is suitable for offline processing because it processes the entire data set multiple times. The solution is not incrementally built as new data is processed.

Thrun et al. (2000) and Burgard et al. (1999) proposed an online version of the *EM* algorithm. Thrun et al.'s algorithm sacrifices the robustness in the data association process to diminish the computational cost. In Burgard et al.'s algorithm local maps are created from the new data. This approach assumes that the odometric error is zero. Based on this assumption the local maps are processed once rather than iteratively. The main drawback of these algorithms is that they achieve suboptimal solutions. As a consequence the maps may contain large inconsistencies making the map unsuitable for the purposes of navigation.

EM approaches build probabilistic maps and have been implemented using laser scanners. Online versions of this type of approach are not robust. We argue that they are not suitable for *low cost* sensors because of the assumption of independence between neighbouring cells in the probabilistic grid maps. For instance, a sonar mea-

surement covers an entire cone in the grid map representation, and from a single sonar measurement it is impossible to determine the location of the object in the sonar cone. A single sonar measurement is then associated with a much larger number of cells than in the case of a laser measurement. The assumption of independence is then less likely to hold in the case of the sonar sensors.

3.4.3 Extended Kalman Filter

Smith et al. (1988) proposed the implementation of an Extended Kalman Filter approach to the *SLAM* problem. In the *EKF* approach a stochastic map of relationships is built. The stochastic map contains *landmarks* and the robot position. The consistency is achieved through the maintenance of the correlations between the *landmarks*. A feature metric map is then required to represent the *landmarks*. The types of features used depend on the available sensor for the implementation.

The *EKF* consists of two stages: prediction and update. The prediction stage consists of the estimation of the robot and features locations based on a predictive model. At the update stage the re-observations from features are used to improve the estimation of the state. An external data association process is required to identify the correspondence between the features from the new observations and the features in the map. The *EKF* is based on the assumption that the noise in the prediction and observation models is a Gaussian distribution with central zero.

The main advantage of the *EKF* approach is that is a recursive estimator. The filter estimates the *full posterior* over maps recursively. The *full posterior* is defined as the most likely positions for the robot and features along with the uncertainty among the features in the map. The only other approach that maintains the full posterior is *FastSLAM* (Montemerlo et al., 2003). The advantage of estimating the full posterior is the proven convergence of the algorithm with probability one to the true map and robot position, up to a residual uncertainty distribution (initial uncertainty in the robot position). Newman (1999) proved the convergence and consistency properties under the assumption of Gaussian noise.

The disadvantages of this approach are the computational cost, the data association problem and Gaussian noise assumption. The $O(N^2)$ (where N is the number of features) computational cost limits the size of the maps. The data association process is not solved by the filter; an external process is required to solve this problem. The correspondence between features from the new observations and the features in the

map becomes unreliable as the uncertainty in the robot position increases. In cluttered environments it is difficult to determine this correspondence. Incorrect data association can be tolerated by the filter up to a certain point, after which the filter diverges. The Gaussian noise assumption is restrictive because it typically does not hold. Sensors are noisy but do not exactly fit Gaussian distributions.

In spite of its disadvantages, to date *EKF* approaches are the most popular because of the proven convergence properties, the recursive estimation, and the capability of solving the revisiting problem. To overcome the disadvantages of the filter, researchers have proposed several improvements to the basic *EKF* algorithm.

Several researchers have developed approaches that reduce the $O(N^2)$ computational cost. The basic idea in these approaches is to maintain the optimal estimation in real time for a subset of features, and to accumulate the pending correction, so that it can be transferred to the complete map when required. The approaches mainly differ in the criteria for selecting the subsets. Guivant and Nebot's (2001) compressed filter selects a subset that contains the closest features to the current robot position. Knight et al.'s (2002) postponement approach selects the subset based on the currently observable features according to the sensors' range. Tardos et al. (2002a) and Williams et al. (2002) proposed local map sequencing techniques. These techniques fuse independent local maps when a common reference is found. Repeated features are identified and used to improve the map estimates. Afterwards these features are removed from the map. Other researchers have proposed similar techniques. For an extensive review consult (Rodriguez-Losada, 2004b).

The data association problem was commonly solved using a *gated nearest neighbour*[1] algorithm (Bar-Shalom and Fortmann, 1988). The data associations with this approach are unreliable in cluttered environments or when the uncertainty in the position of the robot is large. Recently more robust techniques such as the *joint compatibility test* (Neira and Tardos, 2001) or the *graph theoretic approach* (Bailey et al., 2000) were proposed. These techniques test the compatibility of sets of observed features to mapped features, instead of individual tests. A set of features is typically composed of features that are in close locations. The techniques work successfully in indoor cluttered environments.

[1] The gated nearest-neighbour algorithm sequentially matches a set of new features against a previous set of features. Each new feature is compared against all the previous features to determine the best match (according to a similarity criterion). If the similarity between the features is below a certain threshold the features are matched. Matched features are not used in subsequent matches. The algorithm stops once all the new features have been processed.

EKF approaches have been successfully implemented using laser, vision and sonar sensors. We argue that an *EKF* is the most suitable approach for *low cost* sensors and it is the approach we will use. The approach is suitable for dynamic environments because of the feature representation. The feature extraction process filters the noise from the raw measurements. For instance, persons walking in corridors are detected as noisy measurements and are not integrated into the map. Moreover, the consistency and convergence properties of the algorithm have been proved for a single robot (Newman, 1999) and multiple robots (Fenwick et al., 2002).

3.5 Multi-Robot Map Building

To assist us in coming to design decisions there are unfortunately only a few implementations of multiple robots mapping real environments so far. Grid maps and feature based maps have been used to build maps with multiple robots. The approaches of Thrun (Thrun et al., 1998a; Thrun, 2001), Simmons et al. (2000) and Konolige et al. (2004) are based on grid maps and employ laser sensors.

In (Thrun et al., 1998) an extension of the *EM* (Expectation Maximization) algorithm for multiple robots is presented. This extension is used in (Thrun, 2001) where robots cooperate to build maps. A robot is designated as the team leader. The team leader merges the representations and transmits them to the robots. Simmons uses a similar approach to Thrun. Each robot creates local maps, and sends the information to a central agent which improves the map by iteratively combining data from the robots. The global map is used by the agent to decide the place that the robots should explore in the next time step. In Konolige's work when two or more robots come within range of each other (form an exploration cluster), their maps are merged and a leader robot emerges (Konolige et al., 2004). The leader coordinates all the robots, builds a complete map that represents the data collected by all the robots, and broadcasts the map frequently to all the robots in range.

Approaches based on grid maps are not suitable for *low cost* sensor platforms because the use of expensive sensors such as lasers is necessary to build this type of map. Moreover, the frequent exchange of grid maps among large teams of robots is impractical due to its high communication cost. The convergence and consistency properties of grid map based approaches have not been proved. Thrun (2001) remarked "it is unclear how the performance of our approach degrades with inaccuracy of the sensors. For example, it is unclear if sonar sensors are sufficiently accurate to yield good re-

sults". Tardos suggests that grid map based methods for localization rely on the high quality of laser scanned data (Tardos et al, 2002b).

The approaches of Fenwick et al. (2002), Williams (Williams, 2001), Newman (Newman et al., 2002b), Madhavan et al. (2004) and Rodriguez-Losada and Matia (2004) are based on feature maps and implement *EKFs*. Robots observe their relative positions. The relative positions serve as relative frames of reference for the exchange of features.

In Fenwick's research the consistency and convergence properties of the *EKF* are proved (Fenwick et al., 2002). Moreover, the results show that the convergence is accelerated due to the combined information from multiple robots. Williams (2001) extended and generalized his algorithm *CLSF* to multi-robot scenarios. Newman et al. (2002) proposed a technique to combine sensor readings from multiple robots. Madhavan's research used heterogeneous robots in outdoor experiments (Madhavan, 2004). The *EKF* builds a feature map. This type of map requires less storage space and has a lower communication cost than grid maps; moreover, feature maps have been implemented in several *low cost sensor* platforms (Tardos et al., 2002b; Leonard et al., 2002; Wijk and Christensen, 2000).

In conclusion, we argue that feature maps based on an *EKF* are the most suitable approach for our multi-robot map building approach. Previous research has proved the consistency and convergence properties of the filter, and has successfully implemented it in systems with *low cost* sensors.

3.6 The Influence of the Hardware Cost

Many factors have to be taken into account in the design of a robotic platform. One of the main factors is the cost of the components. When developing robot architectures for future implementation it is useful to consider the different rates at which the cost of components are changing. For example, computers are getting cheaper much faster than motors. So the costs of the components for a robot can be usefully divided between: electromagnetic, sensors, and computational. Electromagnetic components are the motors and other elements that allow the robot to move through the environment. Sensor components are the components that allow the robot to observe and interpret the environment. Examples of such sensors are laser, sonar, encoders, GPS, etc. The computational component is the brain of the robot. The brain of the robot is typically a CPU. The exploration task is computationally expensive. Robots have to create

and store a representation of the environment. In a decentralized approach each robot builds its map and integrates information from the other robots. A powerful CPU is required to achieve this task. Often, in *low cost* robots used for exploration purposes the costs in decreasing order are: electromagnetic, sensor and computational costs. Most of the robots used in exploration use expensive sensors (mainly laser). We argue that these sensors are unsuitable for large teams of small size robots because of their size and cost. Small robots can not carry these sensors because of their size and their power consumption requirements.

Low cost small robots are suitable for some exploration tasks because they are expendable. A *low cost* robot is a robot that uses inexpensive sensors such as sonar, infrared, bumpers, etc. Small robots are desirable because large numbers of them can coexist without disturbing each other's tasks (e.g. colliding into each other). In a rescue mission small robots can easily traverse an environment while in a patrolling mission small robots are less noticeable.

As previously mentioned it is undeniable that the exploration task is computationally expensive. A small robot used for exploration purposes therefore has to carry a powerful CPU. To this day there is no small robot with this capability. However because of the current trends in microprocessor technology we believe that such computer power will be available in the near future at a very low cost. For instance Jantz and Doty (2002) used a PDA in a *low cost* small robot for tracking purposes in a vision system. The robot successfully tracked a target in real time. PDA devices these days are cheap and very powerful devices with wireless capabilities and can be easily incorporated in small robot platforms (Williams, 2003). This suggests that it is now becoming economically feasible to use these devices in exploration tasks.

The motivation of this work is that the proposed architecture should be suitable for implementation in *low cost* robots such as Robomote (Sibley et al., 2002), Swarmbots (McLurkin and Smiths, 2004) and Millibots (Grabowski et al., 1999). The motivations to build these platforms are scientific and economic. From a scientific view point these platforms help to understand better collective intelligent behaviour exhibited by biological "systems" such as flocks of birds, schools of fish, and colonies of ants, termites, and bees. From an economic point of view it is important to consider how the task assigned to the robots can be achieved effectively given a certain budget; having many very simple and *low cost* robots or fewer more complex and expensive robots? Brooks and Flynn (1989) addressed this issue for a space exploration scenario. They argued that large numbers of robots can change the trade-off between reliability of

individual components and overall mission success. Gage (2002) pointed out that a robotic system that consists of a large number of identical robots places a premium on lowering the cost of each unit, and, in fact, justifies significant "up front" investment for this purpose, since this investment is amortized over many units. For these reasons we argue that large numbers of *low cost* robots are a good solution for exploration tasks.

The cheapest sensors are bumpers and short range infrared sensors. As later discussion will show these sensors are not suitable for mapping tasks in an office environment because of their small range (up to 0.30m). A system based on this type of sensors requires an external localization system that increases the cost of robot. For instance, in the RoboMote project (Sibley et al., 2002) robots obtain relative locations with respect to each other from wireless communication signals. The goal of the RoboMote research is the deployment of the robots as a mobile sensor network.

Sonar sensors and infrared sensors (Triangulation-Based) are cheap, small and provide a good sensing range (up to 6m and 1.5m respectively). They have been used for exploration purposes in the Millibots project. Millibots use sonar sensors for localization and mapping. For localization the robots perform trilateration through the use of distance measurements obtained from sonar sensors to another three Millibots. The Millibots leapfrog each other to maintain good position estimates as they traverse unknown terrain. Millibots create a grid map representation of the environment. Infrared sensors based on triangulation have been developed in the last three years. These sensors have a narrow beam width (2°). The range of these sensors is shorter than that of sonar sensors. As previously discussed laser sensors are not suitable because of their size and cost for large teams of small size robots.

This thesis is interested in the exploration of structured environments like office environments. Typical office environments are formed by corridors and offices. Corridors have a typical width of 2 to 4m. Offices typically have desks, cabinets, etc. Mapping offices with sonar sensors is difficult because of the wide beam-width associated with sonar measurements. The use of infrared sensors can simplify the mapping task without incurring high costs: computational and economic. We believe that the combined use of sonar and infrared sensors presents a suitable alternative to lasers for a *low cost* robot platform. The next section presents a discussion of previous research using sonar sensors for mapping purposes.

3.7 Feature Extraction for Sonar Sensors

Researchers agree that building maps using sonar sensors is far more difficult than with laser data. Sonar measurements have one *DOF* (distance to the closest obstacle). Their angular precision is typically in the range of 20 to 45 degrees. Features such as points and lines have two *DOF*. Despite the increased difficulty of extracting features from these sensors their advantage is their low cost. Laser sensors are much more expensive than sonar sensors.

Various approaches have been proposed to overcome the limitations of sonar sensors. Some approaches use arrays of sensors that allow feature detection and discrimination. Kleeman and Kuc (1995) achieved very precise localization and classification of features from a rotating sensor array. Barshan and Kuc (1990) developed an intelligent sonar sensor that disambiguates reflections from planes and corners using time of flight at different frequencies. These approaches are based on carefully engineered configurations with known baselines, the same principle as is used in classical stereo vision (Faugeras and Luong, 2001).

Other approaches use sonar data from multiple locations. Wijk and Christensen (2000) developed a triangulation technique to model points. The technique is based on the intersection of arcs of likely localization from several sonar readings. The robot takes some readings from one position and stores them in a temporary buffer. It then moves some small distance and takes more readings. A simple voting scheme generates hypotheses of possible targets. The drawbacks of the approach are that it only models point features and its computational cost. Every new reading produces a hypothesis that is tested against all the rest.

Tardos et al. (2002b) proposed a similar triangulation technique based on the buffering of readings. This model incorporates point and line features. A Hough transform approach obtains the parameters of the features. The feature parameter space is discretized by means of a grid. Each echo is projected into the parameter grid, accumulating votes for possible causative grid cells. The grid cells with the most votes indicate the parameters of the features that explain the data. The drawback is that the entire parameter space is represented in a grid which imposes a trade-off between resolution and computation time.

Recently in Leonard's research a random sample and consensus (*RANSAC*) approach has been used to extract features from multiple vantage points (Leonard et al., 2002). This is similar to Tardos' approach since each random sample determines

a point in the feature space that is voted for by the remaining echoes. The data is pre-processed as it arrives to estimate which echoes may match. This has achieved remarkable results in localizing sonar-based robots. The drawback of the approach is the delay in feature incorporation into a map.

The use of additional *low cost* sensors may improve the map building process. This thesis proposes the use of infrared sensors to improve the localization estimates. A novel *low cost* platform based on these sensors was proposed and built. Section 7.7 (page 207) presents the designed platform. Experiments to validate the platform and compare the performance against sonar alone are also presented.

3.8 Summary

The problem of building a map is that as the robot moves trough the environment it has to concurrently build the representation (map) and localize itself within this representation. This is referred to as *SLAM* (Simultaneous Localization and Mapping). This problem has been successfully solved with probabilistic techniques, based on Bayes Theorem. Two types of probabilistic representations are used to represent the environment, grid cells and feature maps. In a feature based map the objects are represented using a geometric model. Walls are represented as segments; corners are represented as points, etc. In grid maps the space is divided into cells. Each cell has a value that represents the probability of being occupied. Because of their geometric representation feature maps require less storage space and have a lower communication cost than grid maps.

Various approaches to solve the *SLAM* problem based on Bayes Theorem have been proposed. The approaches differ in their assumptions and simplifications of the Theorem; each approach has his advantages: *EKF* (Extended Kalman Filters), *EM* (Expectation Maximization) and Scan-Matching.

The *EM* and Scan-Matching algorithms have been implemented with laser sensors. Real time implementations of these algorithms sacrifice robustness. Their consistency and convergence properties have not been proved.

We argue that an *EKF* approach is the most suitable solution for the map building problem. Its consistency and convergence properties have been proved for a single and multiple robot scenarios. The approach is suitable for dynamic environments because of the feature representation. The feature extraction process filters the noise from the raw measurements. Its disadvantages are its high computational cost and the data asso-

ciation problem. However these disadvantages have been addressed in recent research. The high computational cost is reduced using local maps that are incorporated in the global map as required. Data association techniques such as the joint compatibility test are robust against clutter and large uncertainty in the location of the features.

One of the main goals of this work is to enable groups of *low cost* robots to build maps of the environment. A *low cost* robot is a robot that uses inexpensive sensors such as sonar, infrared, bumpers, etc. In previous research by others (Leonard et al., 2002), the *EKF* approach has been successfully implemented with *low cost* sonar sensors. Building maps using sonar sensors is far more difficult than with laser data because sonar measurements have limited angular precision and reflection problems. The use of additional *low cost* sensors can improve the map building process. We propose the use of infrared sensors to improve the sonar localization estimates. A novel *low cost* platform based on these sensors was proposed and built. Chapter 7 presents the design and testing of the *low cost* platform proposed. Our simulations with multiple robots are based on this platform.

There are two main problems to address in this project. The first problem is how an individual robot with *low cost* sensors can localise itself and build a map. The second problem is how to coordinate the map building process of several robots. Having decided how to approach both problems this is a good point at which to review the progress through the argument of this thesis.

Chapters 1 and 2 presented a discussion of previous approaches to explore environments using multiple robots. Chapter 1 described the distinctive features of the architecture presented in this thesis called BERODE (BEhavioural ROle DEcentralized). This chapter discussed the suitability of previous approaches to build maps for the BERODE architecture. It was concluded that the most suitable approach was an Extended Kalman Filter Approach. Chapter 4 describes BERODE in terms of global team behaviour. The goal for the team of robots is to explore environments while remaining as a single connected communication network. In the approach each robot builds a feature map representation. Robots periodically exchange the features that they extract locally to keep a common representation of the environment. Chapter 5 presented the implementation of the BERODE architecture in individual robots. Chapter 6 describes the Extended Kalman Filter used by the robots to create and update their maps. Chapter 7 presents experiments that show that the sensors and communication models used in our simulations are reasonable and conservative approximations to data obtained from hardware devices. Chapter 8 presents the implementation of

BERODE for *LOS* and *RF* technologies. Chapter 9 presents simulations that show the robustness, the scalability and the efficiency of BERODE. Our simulations show that the approach is robust to infrequent communication, scalable with respect to communication and number of robots, and more efficient than approaches with fixed control topologies. We can't conclude that these properties would still transfer to a real world implementation, but we have taken steps (Chapter 7) to make our simulations as realistic as possible in the relevant aspects. Chapter 10 presents simulations that analyse the consistency of the maps built by the simulated robots in multi-robot scenarios. Chapter 11 presents the conclusions and contributions of this thesis.

Chapter 4

Purposes and Design of BERODE

4.1 Introduction

In recent years several multi-robot exploration architectures have been proposed. These architectures require some degree of centralization to achieve coordination (Thrun, 2002). Most of them rely on a single centralized agent that plans and coordinates the team (Simmons et al., 2000). These architectures are suitable for a small number of robots but they do not scale to large numbers of robots because of their computational and communication costs. The computational cost of centralized planning typically increases combinatorially with the number of robots. The communication bandwidth becomes a bottleneck for the system because information has to be gathered by the central agent.

Decentralized architectures on the other hand are robust, flexible and suitable for a large number of robots. These architectures have achieved less efficient[1] solutions compared to those of centralized architectures (Yamauchi et al., 1998). These less efficient solutions are typically due to lack of coordination between agents. To coordinate agents have to communicate. Low power communication systems have a limited range. This thesis proposes a novel architecture where the coordination of exploration is achieved by keeping the robots as a single connected and adaptable communication network. The likelihood of having overlapping tasks increases when the robot network splits into groups. Our architecture tries to minimize task overlapping to improve efficiency through coordination. To achieve coordination our architecture requires that at

[1] By efficient we refer to the time used to build a complete map. We consider that a map has been completed once any one robot has projected its feature map into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a user defined threshold (Section 8.6, page 239).

the start of the exploration the robots know their relative locations.

This chapter describes the features of our architecture called BERODE (BEhavioural ROle based DEcentralized) and explains how the exploratory behaviour of the network emerges from the interaction of the individual behaviours of the robots. The following chapter presents the detailed implementation of the BERODE architecture in the robots.

BERODE is based on behavioural roles such as Explorer and communication Maintainer. These roles reactively adapt to the dynamic conditions of the communication network formed by the robots as they explore an environment. The communication network formed by the robots is kept fully connected by creating and maintaining an adaptive control network. The control network is a subnetwork of the communication network containing only the necessary connections to maintain the communication network connected. This abstraction improves exploration efficiency. To improve signal quality the robots are allowed to modify or rebuild the communication network.

As the robots explore the environment they build a feature based map to represent it. The positions of features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[2] number. Each robot builds and updates its own map using an Extended Kalman Filter (*EKF*). The *EKF* produces an estimate (along with its uncertainty) of the location of the features and the robot. As the robots explore the environment they extract new features from their sensor measurements. These features are used by the *EKF* to update the estimated locations. The extracted features are periodically distributed among the team of robots. Robots incorporate the received features to their maps using the same process as for locally extracted features. This is possible because the robots use the same global Cartesian frame of reference.

The robots periodically send beacon signals that contain their estimated position and its uncertainty. The robots adopt behavioural roles based on their state[3] in the control network. The behavioural roles balance between the tasks of network maintenance and exploration. According to its behavioural role, a robot exhibits some interest or none in the exploration task. As a result a number of robots in the network direct the exploration towards unexplored areas while the rest of the robots ensure the maintenance of communication. The behavioural roles differ in the implementation of some of their modules and their parameterization. The robots reselect their behavioural role

[2] The robots have an ID number that allows them to identify each other in the network.

[3] The state in the control network for a robot is the number of connections and their quality.

once an event as occurred. An event is generated when a robot has reached its current goal or modified the control network. When a robot modifies the control network the modified control network is transmitted to the robot network. The robots then update their control network and reselect their behavioural roles.

The roles generate plans to keep the robot's direct connections in the control network within communication range. The plans of the non-Explorer robots are based on the imposition of *virtual forces* by the robot's direct connections on the control network. *Virtual forces* are attractive/repulsive relations between pairs of robots. These forces are modelled as *virtual springs* where the free spring length is a function of the behavioural role of the robot and its connections. The Explorer robots generate plans to move towards attractive unexplored areas. The attractiveness of an unexplored area is a function of its size, the path length and the predicted communication safety at that location. The predicted communication safety is the expected signal quality based on the estimated positions of the robots. The position estimates are obtained from the beacon signals that the robots transmit periodically.

To improve scalability with respect to communication BERODE implements a two level hierarchic approach to distribute information. The levels are: local and global. Information is shared more frequently at the local level than at the global level. The local level for a robot is the network that contains all the robots within a *k-hop* distance. The global level is formed by all the robots. The robots have to cope with the delays in the reception of information. Large delays deteriorate the performance of the network.

The chapter is structured as follows:

- Section 4.2 presents the formal definition of the problem and describes the exploratory behaviour of the robot network which emerges from the interaction of the individual behaviours of the robots.

- Section 4.3 presents the formal description of the state of the communication network, and the *virtual spring* model used to generate *virtual forces* to keep the control network connected. The state of the network is represented by the current topology of connections in the communication network. The terminology used throughout the rest of the chapter is also defined in this section.

- Section 4.4 presents the algorithms used by the robots to create and update the control network. Robots can modify or rebuild the control network. This network is used for two purposes: control and information distribution. For the purposes of control it is used to keep the robots as a fully connected network.

For the purposes of information distribution it is used to efficiently distribute knowledge (e.g. robot location, extracted features, goals, etc.) relevant to the other robots in the network.

- Section 4.5 presents the proposed behavioural roles: Explorer, Maintainer, Recoverer and Pusher. The tasks and goals for each behavioural role are described in this section. The roles are based on previous research in decentralized control (Thibodeau et al., 2004) and on the experimental findings encountered in the design of the architecture (Vazquez and Malcolm, 2004).
- Section 4.6 presents a summary of the contributions of the proposed architecture.

4.2 Problem Definition

The previous chapters introduced and discussed the problem of decentralized coordination for multiple robots. The problem is to explore an initially unknown environment efficiently using a group of robots that that start off at known locations. The robots have sensors to build a representation of the environment and a communication system to communicate periodically. The robots estimate their position using the information acquired from their sensors. Our approach to solve this problem is:

> *To generate coordinated behaviours for the group of robots that allow them to explore the initially unknown space while meeting the constraints imposed by a communication system.*

4.2.1 Proposed Solution

The general idea of the proposed solution is that the exploratory behaviour of the network should emerge from the interaction of the individual behaviours of the robots. The behaviours generate reactive plans that explore the unknown environment while avoiding collisions with obstacles and other robots in the environment. Robots adopt their behavioural role according to their state in the robot network. This state is determined by the number and the signal quality of the connections of the robot in the control network. The control network contains the minimum necessary links to keep the communication network connected. To improve signal quality the robots are allowed to modify or rebuild the communication network.

In our approach each robot builds and updates its own feature map representation of the environment. The robots' maps incorporate the features observed by all the

robots in the network. The feature map is updated using an Extended Kalman Filter (*EKF*) (Smith et al., 1988). The *EKF* produces an estimate (along with its uncertainty) of the position of the features and the robot. The positions of robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[4] number. The robots extract features from their sensor measurements. The extracted features are used to update the feature map. The robots periodically transmit their feature observations. Robots incorporate feature observations of other robots with the same process as for locally extracted features because they share same global Cartesian frame of reference.

4.2.2 The Exploratory Behaviour of the Robot Network

The exploratory behaviour of the robot network emerges from the interaction of the individual behaviours of the robots. This interaction is achieved through the imposition of virtual spring forces derived from the connections in the control network. The control network contains only the necessary connections to keep the communication network connected. This network is calculated at the beginning of the exploration by the robot with the smallest ID. To improve signal quality the control network can be modified or rebuild by any robot. The robots periodically transmit beacon signals. The beacon signals contain their estimated position.

The robots exhibit one of the following behavioural roles: Explorer, Maintainer, Pusher and Recoverer. The Explorer and Maintainer behavioural roles are the basic behaviours for the robot in the network. The Explorer robots are the robots with one connection in the control network while the Maintainer robots are the robots that have two or more connections in the control network. Because of this the Explorer robots focus on exploration while the Maintainer robots focus on keeping the robot network connected.

The Pusher robots have one control connection therefore the Pusher-Explorer connection is not possible for a robot network bigger than two robots. The purpose of the Pusher behavioural role is to help the exploration task when it becomes problematic; for instance when two Explorer robots pull the robot network in opposing exploratory directions the exploratory behaviour of the robot network is halted. One of the Explorer robots gives up its exploration task and transitions to the Pusher behavioural role. The Pusher robot then induces movement in the robot network by pushing its

[4] The robots have an ID number that allows them to identify each other in the network.

direct connections away from it.

The Recoverer robots have one or more connections in the control network. The Recoverer robot helps the communications maintenance task when it becomes problematic. The Recoverer role tries to improve the level of safety for unsafe control connections. A connection is unsafe when its signal quality level is below an unsafe threshold (user defined). The measure of the signal quality for the connection depends on the hardware implementation. For instance, in the implementation for *RF* technologies we use the *RSSL* (Received Signal Strength Level) which is an available value for this technology (measured in dB) as the signal quality value. The robots transition between behavioural roles in response to the changes in the control network topology and the signal quality of the connections.

The non-Explorer robots keep the communication network connected by generating plans to move to locations where the energy from the virtual spring forces is minimized. The Explorer robots are not directly subject to virtual spring forces; instead these robots monitor the level of safety for their control connection and wait for improvement if necessary to avoid the risk of becoming disconnected. The Explorer robots guide the exploration process by generating a plan to move to the safest unexplored area in terms of communication quality and moving in that direction to the limits of their communication safety.

The virtual spring forces are described as heterogeneous because they are asymmetric[5] and their free spring length[6] is a range of values rather than a single value. The magnitude of these forces is a function of the quality of the connection and the behavioural roles of the robots that form the connection. For instance, for a Maintainer-Pusher connection the Maintainer can have a repulsive force from a Pusher robot while the Pusher robot can have an attractive force from the Maintainer robot.

The thresholds for the attractive and repulsive heterogeneous spring forces are set up to disperse the robot network towards the unexplored areas when there are no Recoverers robots (no unsafe connections) and to contract the robot network when there are Recoverer robots (unsafe connections). At a close distance all the heterogeneous spring forces exert a repulsive force to avoid collisions.

The dispersion in the exploratory directions is achieved by setting the heterogeneous forces for the control network in such a way that a Maintainer robot is more at-

[5] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[6] The free spring length is the length for which the spring exerts a null force.

tracted to Explorer robots than to Maintainer robots and is repulsed by Pusher robots. Pusher robots are equally attracted to Maintainer and Explorer robots. The Pusher robots induce movement because they move towards the Maintainer robots while the Maintainer robots try to move away from them while moving closer to the Explorer robots. The Explorer robots move towards the unexplored areas dragging the robot network behind them.

Figure 4.1 presents an example that shows how the exploration is achieved trough the imposition of spring forces. It is observed that the Explorer robots R_0 and R_2 are attracted to the unexplored areas f_0 and f_1 respectively. The Maintainer robots are pulled by the Explorer robots towards the unexplored areas. The robots pull away from the Pusher robots. From the force diagrams it is observed that the Pusher robot R_4 exerts a large repulsive force on robot R_3 compared to the attractive force exerted by robot R_1. It is also observed that the Explorer robots R_0 and R_2 exert large attractive forces on robot R_1 while robot R_3 exerts a small attractive force.

The network is contracted when Recoverer robots exist. A Recoverer robot is subject only to the forces exerted by its unsafe connections. The Recoverer robots are then attracted to their unsafe connections. As the Recoverer robots move toward their unsafe connections their safe connections are pulled because the signal quality for these safe connections decreases. This pulling effect spreads through the network starting with the Recoverers. Figure 4.2 presents an example of the effect of contraction in the robot network. It is observed that the robots R_1 and R_3 transition from Maintainer to Recoverers. These robots start moving towards each other to improve the signal quality of their connection. This starts the contraction of the robot network, the robot R_4 then transitions to the Pusher behavioural role to avoid the risk of becoming disconnected. Once the connection between robots R_1 and R_3 is safe the exploration continues. Robot R_4 remains as a Pusher and aids the exploration by pushing the robot network towards the unexplored areas.

In BERODE the robots are allowed to modify or rebuild the control network to improve the signal quality of the MST connections. The updated control network contains the connections with the best signal quality. This facilitates the network maintenance task. The detection of new connections triggers a recalculation process that modifies the control network if the signal quality of the communication network is improved. The processes of building and modifying the control network are described in Section 4.4.

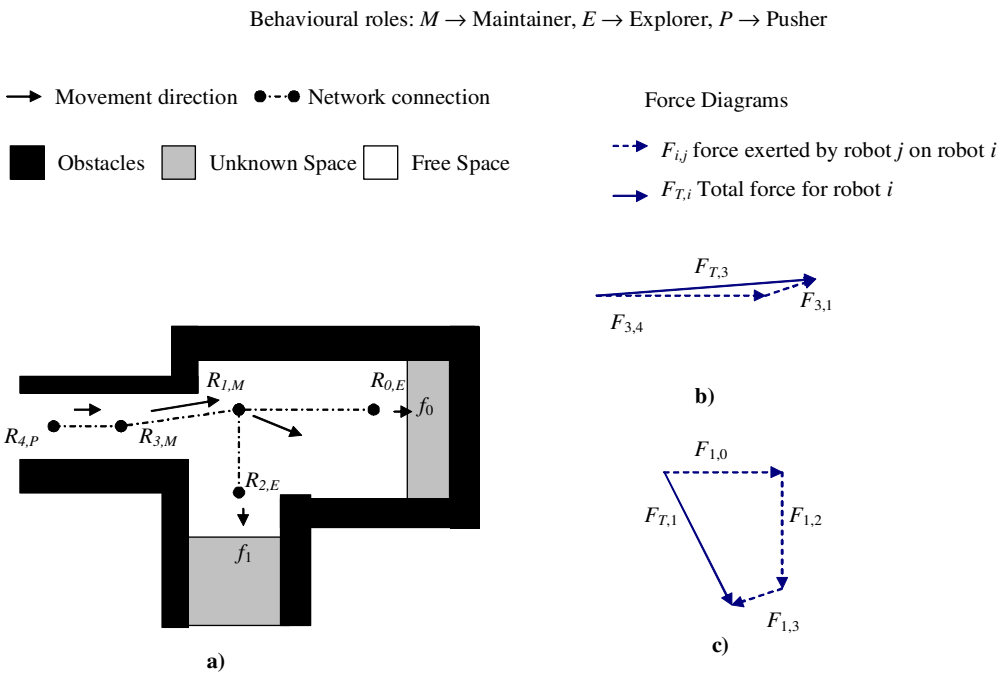


Figure 4.1: An example of the spring forces exerted by the robots in the network. a) The Explorer robots R_0 and R_2 are attracted to the unexplored areas f_0 and f_1 respectively. The Maintainer robots are pulled by the Explorer robots towards the unexplored areas. The robots pull away from the Pusher robot R_4. b) Robot R_4 exerts a repulsive force on robot R_3 while robot R_1 exerts an attractive force. c) Robots R_0, R_2 and R_3 exert attractive forces on robot R_1.

4.2.3 The Mapping Behaviour of the Robot Network

In BERODE each robot builds and updates its own map using an Extended Kalman Filter (*EKF*). The *EKF* produces an estimate (along with its uncertainty) of the location of the features and the robot. As the robots explore the environment they extract new features from their sensor measurements. Line and point features are extracted from the raw sensor data. A geometric representation of the features is obtained. The features are represented by a measurement vector and an uncertainty matrix. These features are used by the *EKF* to update the estimated locations.

The robots periodically distribute their observed features to the rest of the team. To improve scalability with respect to the numbers of robots the periodical distribution of features is done at two levels: local and global. The features are distributed more frequently at the local level than at the global level. The local level for a robot is formed by the robots that are within a *k-hop* distance in the control network. The global level is formed by all the robots in the network. The features distributed at local and global

levels are the features observed since the last distribution at the local and global level respectively. The features received at the global level are incorporated to the robots' maps using the same process as for locally extracted features. This is possible because the robots use the same global Cartesian frame of reference. The features received at the local level are used as navigational aids but not integrated to the robots' maps because their observations are contained in the features distributed at the global level.

The computational cost of planning in BERODE is small because the robots mostly generate short term plans to move towards close locations. With respect to planning, most of the time each robot maintain a local view of its map. Only when an Explorer robot does not find close unexplored areas it does acquire a global view of the map to search for unexplored areas in this map. The robots use their feature map to generate their plans.

The non-Explorer robots generate short term plans to move to the location where the energy from the virtual spring forces is minimized. The plan is generated by projecting the features that lie inside a local grid map. The limits of the local grid map are determined by the estimated positions of the robot and its control connections. Figure 5.9 (page 108) presents an example of the local grid map obtained from the projection of the feature map.

The Explorer robots generate plans to move towards unexplored areas. To generate the plan the Explorer robots also project a subset of features to identify close unexplored[7] areas safe to explore. An unexplored area is safe to explore if the communication coverage for this area is predicted to be safe. The robot selects the safe unexplored area with the largest utility. The utility of an unexplored area is the information gain value minus the cost of travelling to this area from the robot position assuming ideal movement. The information gain for an unexplored area is the number of unexplored cells that are within the circumference of the sensor range of the robot. If there are no close unexplored areas safe to explore the Explorer robot projects all the features into a global grid map. If there are no safe unexplored areas in the global map the robot considers the exploration of unsafe unexplored areas. If there are no unexplored areas the map is considered as complete. The exploration process stops once any one robot considers the map as complete.

The search for unexplored areas in the global grid map is computationally expensive, but only performed by explorers who can't find any local unexplored areas to

[7] An unexplored area is a portion of the environment in the projected grid map for which there is no evidence. The size of this portion is user defined.

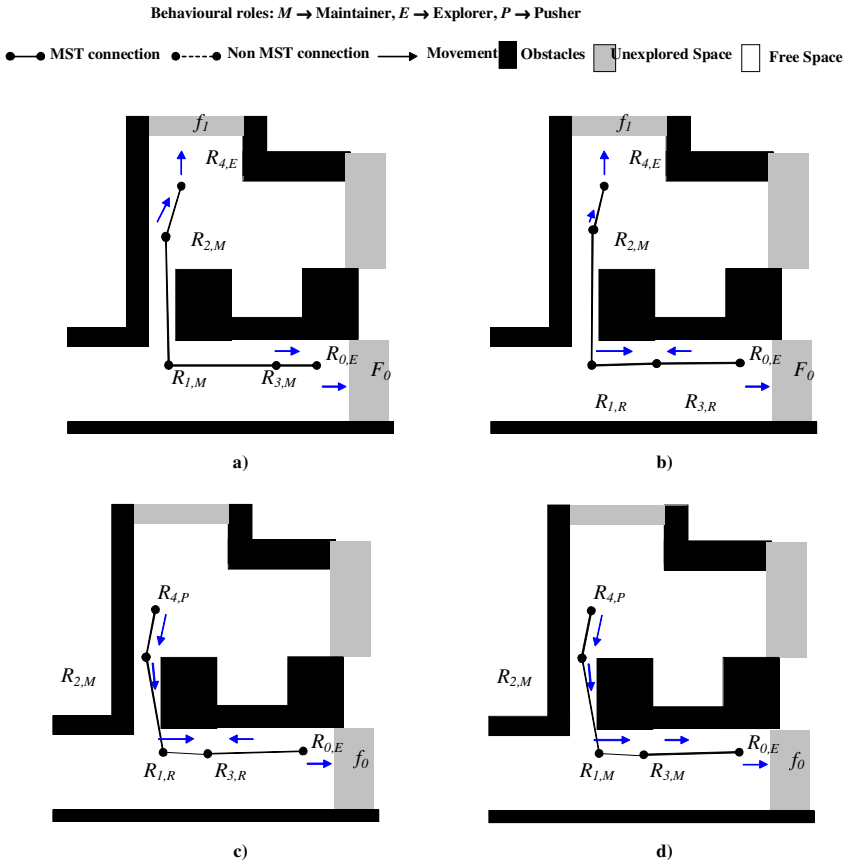


Figure 4.2: An example of robot network contraction. a) Robots R_0 and R_4 move towards the unexplored areas f_0 and f_1 respectively. b) The robots R_1 and R_3 transition from Maintainer to Recoverers and start moving towards each other to improve the signal quality of their connection. c) The network contracts and as a consequence robot R_4 transitions to the Pusher behavioural role to avoid the risk of becoming disconnected. d) The signal quality for the connection between robots R_1 and R_3 is safe and the exploration continues.

explore. This doesn't happen often and typically only towards the end of exploration when most has been mapped. Once an Explorer robot has made a plan to reach a distant unexplored area then the global search will be not be repeated until the robot reaches this unexplored area. To reduce the likelihood of task overlapping the Explorer robots transmit the location of the area that they selected to explore.

The following subsection formalises the description of the state of the robot network and defines how the communication constraints for the robots are imposed.

4.3 The State of the Robot Network

The state of the robot network for a robot is determined by its control connections. A robot has a direct connection with another robot if it is within communication range. A control connection is a connection that a robot has to maintain. Control connections are pair-wise constraints among pairs of robots.

The state of the robot network can be expressed as a graph G where the robots are the vertices and the direct connections represent the edges. The graph G is represented by a tuple $(\eta, \varepsilon, \kappa)$, where η is the set of robots, $\varepsilon \subseteq \eta x \eta$ is the set of edges representing the connection network and $\kappa \subseteq \varepsilon$ is the set of current control connections. κ_i is the set of control connections for robot i that contains $|\kappa_i|$ elements. $\varepsilon_{i,j}$ and $\kappa_{i,j}$ represents the connection status and control connection between the robots i and j respectively. The existence of a connection between R_i and R_j is expressed as $\varepsilon_{i,j} > 0$ and referred to as direct connection.

The value of $\varepsilon_{i,j}$ is referred to as the signal quality. Bigger values of $\varepsilon_{i,j}$ indicate a better signal quality. κ has to contain at least one connection for each robot. The metric to measure the signal quality depends on the implementation. Radio frequency (*RF*) technologies provide the *RSSL* (received signal strength value) which is a measure of the quality of the signal. In the implementation of the approach for *RF* (Chapter 8) this value is used as the signal quality. The signal quality is measured in decibels (dB).

Line of Sight (*LOS*) technologies (e.g. infrared communication) typically do not provide a measurement of the quality of the signal. In the implementation of the approach for *LOS* (Chapter 8) the signal quality value is predicted based on the current explored map. The signal quality in this case is defined as the distance between the pairs of robots and is measured in meters. Although *LOS* technologies may have very large communication ranges, because lines of sight are easily blocked, it is undesirable to permit long ranges in cluttered environments. Any small adjustment of position over a long sight line in structured or cluttered environments could easily block the connection. It is argued that this is less likely to happen when the robots are in close positions because the probability of having an obstacle between a pair of robots is smaller when they are closer. For this reason in the *LOS* implementation the signal quality is considered better when the robots are closer. The inequalities of Equations 4.1, 4.2 and 4.3 have been inverted to reflect our reasoning (Section 8.3, page 231).

For each control connection there are two inequalities $\kappa_{i,j} < \sigma_{safe}$ and $\kappa_{i,j} < \sigma_{prec}$ where $\kappa_{i,j} = \varepsilon_{i,j}$ and $\sigma_{safe} > \sigma_{prec}$. The parameters σ_{safe} and σ_{prec} are the commu-

nication thresholds that represent the desired levels of signal quality between pairs of robots. The values of the thresholds depend on the behavioural role of the robot and its direct connections.

The inequalities define three levels of safety with respect to the robots' configuration. A robot i with a control connection set $\kappa_i = \{\kappa_{i,1}, \kappa_{i,2}, ..., \kappa_{i,z-1}, \kappa_{i,z}\}$ is in a safety level L_i according to

$$L_i = \begin{cases} Safe & if \quad \forall j \left(\kappa_{i,j} > \sigma_{safe}\right) \\ Precautionary & if \quad \forall j \left(\kappa_{i,j} \geq \sigma_{prec}\right) \wedge \exists j \ \left(\kappa_{i,j} \leq \sigma_{safe}\right) \\ Unsafe & if \quad \exists j \left(\kappa_{i,j} < \sigma_{prec}\right) \end{cases} \tag{4.1}$$

The behavioural roles are designed with the objective of keeping the robots inside their safe level in the configuration space. The precautionary level is designed to maintain a certain signal quality. In the unsafe level communication is marginal with at least one control connection. It is observed from Figure 4.3 that the safety levels define regions in the communication space of the robot. The safety level for a position can be estimated based on these regions. In BERODE a robot estimates the safety level for nearby positions for two purposes: network maintenance and exploration. For the purpose of network maintenance the signal quality from the robot's communication constraints is predicted for nearby positions to the robot position. The robot moves to the position where the overall signal quality (OSQ) is the best. For the purpose of exploration the signal quality for nearby unexplored areas is predicted to determine if the area is safe to explore. The robot tends to plan to explore areas that are predicted to have a safe level.

The set of control connections for a robot is the set of connections in the *minimum spanning tree* (*MST)* of the connection network. The *MST* contains the minimum number of connections necessary to maintain the network of robots connected and will be referred as MST control network. The signal quality of the control connections determines the safety level. Based on the safety level a set of communication constraints is activated. This set is a subset of the robot control connections in the MST control network. The communication constraints exert virtual forces that impose constraints in the robot movement ensuring that the robot is maintained within communication range of its control connections.

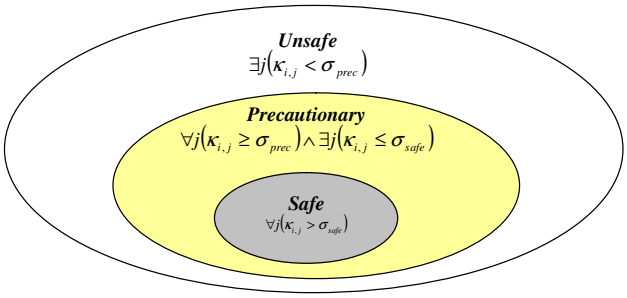


Figure 4.3: The three safety levels define regions in the robot communication space. The robot generates communication sensitive plans to try to remain in the safe region.

4.3.1 MST Control Connections

As defined previously, ε is the set of edges representing the network status and $\kappa \subseteq \varepsilon$ is the set of current control connections. The only condition necessary to maintain the *n* robots connected as a single network is that there are at least *n*-1 control connections and that κ has to contain at least one control connection for each robot. However, it is not necessary for a robot to maintain all of its connections. Moreover, the more control connections that a robot has to maintain the more constrained the motion of the robot is. Thus, it is desirable that the set κ of control connections contains the least number of constraints.

A spanning tree satisfies the required conditions. A spanning tree of a graph is a sub graph which is a tree and connects all the vertices together. The *minimum spanning tree* (*MST)* is the best spanning tree because it minimises constraints on robot movement while preserving communication safety. In BERODE several heuristics to calculate the *MST* are proposed and compared. These heuristics use different weight metrics and are based on the robots' estimated locations and the signal quality. The objective of the heuristics is to maximize the number of the robots in their safe levels.

4.3.2 Activation of Communication Constraints and the Network Status

The communication constraints are activated depending on the current safety level. For a robot *i* the set of communication constraints ϕ_i is defined as

$$\phi_i = \begin{cases} \{\} & Safe \\ \{\, j \,|\, \sigma_{prec} \leq \kappa_{i,j} \leq \sigma_{safe} \} & Precautionary \\ \{\, j \,|\, \kappa_{i,j} < \sigma_{prec} \} & Unsafe \end{cases} \tag{4.2}$$

The communication constraints and the internal state of the robots determine the behavioural role. In the unsafe level the set of communication constraints is a subset of the control connections $\phi_i \subseteq n_i(z)$. It is argued that a robot could move out of the unsafe level faster when is constrained only by the subset of unsafe connections instead of the complete set of control connections. σ_{safe} has to be much larger than σ_{prec} to minimise the risk of disconnecting the robot network because of the temporary exclusion of some control connections. The network status of a robot is defined as the set of communication constraints and the current safety level.

It is important to observe that, regardless of the number of robots and communication constraints, when the robot i is constrained by the robot j, robot j is also constrained by robot i. This is important for the decentralized control approach.

4.3.3 Configuration Space for Attractive/Repulsive Forces

To maintain the connectivity of the network, the robots have to satisfy their communication constraints. This thesis proposes a *virtual spring* model to maintain the communication constraints within communication range. In this approach robots exert attractive/repulsive vector forces on each other. These forces are based on a *virtual spring* model and are a function of the signal quality as well as the behavioural roles. The potential energy vector generated by the combined forces is calculated. The potential energy vector suggests a direction where the energy for the communication constraints is minimized; therefore the overall signal quality (OSQ) is maximized. This vector is used by the robot to try to improve the OSQ for its communication constraints. A robot in the safe level remains in the same position because no communication constraints apply at this level.

As mentioned in Chapter 2, in previous research pairs of robots exert homogeneous forces between each other (McLurkin and Smiths, 2004). A homogeneous force is a force that depends only on the signal quality. This research proposes the implementation of heterogeneous spring forces. The spring forces are described as heterogeneous because they are asymmetric and their free spring length is a range of values rather than a single value. The free spring length is a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection. It is argued that the use of heterogeneous spring forces aids the exploration process because the movement in the exploration directions can be accelerated by setting different thresholds (Section 4.2).

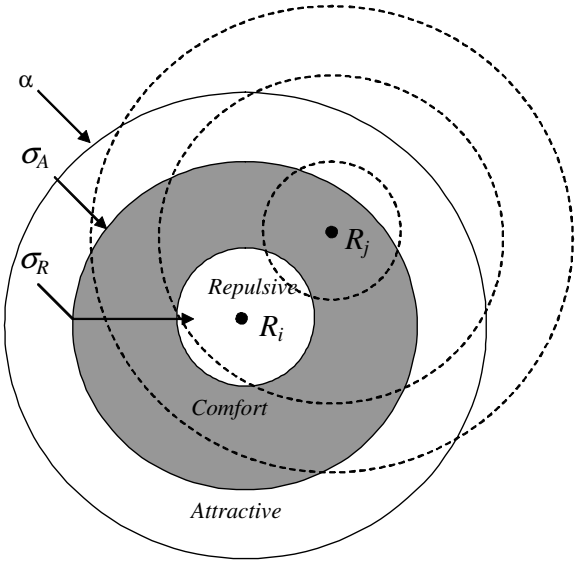


Figure 4.4: Robots R_i and R_j have a communication constraint $\kappa_{i,j}$. The constraint defines three force regions in the configuration space: attractive, comfort and repulsive. The robots R_i and R_j try to remain in their comfort region by exerting attractive/repulsive forces on each other when they are outside this region.

In general, as defined in the previous section $\kappa_{i,j}$ is the communication constraint between robots R_i and R_j. The constraint defines three force regions in the robot configuration space: attractive, comfort and repulsive (Figure 4.4). These regions are formed because the free spring length is a range. The regions depend on the signal quality $\varepsilon_{i,j}$ ($\kappa_{i,j} = \varepsilon_{i,j}$ for the communication constraint) and their boundaries σ_A and σ_R. σ_A and σ_R are thresholds that depend on the behaviours exhibited by the pair of robots. The constraint $\kappa_{i,j}$ exerts an attractive force when $\kappa_{i,j} < \sigma_A$ and a repulsive force when $\kappa_{i,j} > \sigma_R$. The repulsive force is designed to avoid collisions between the pair of robots. For the proposed *virtual spring* model the range of values $[\sigma_R, \sigma_A]$ is a range where the force generated by the *virtual spring* is null. This range defines the comfort region in which the signal quality is reliable for the communication constraint. When $\varepsilon_{i,j} < \alpha$ the communication is lost. α is the maximum communication range. The discomfort distance for robots R_i and R_j is defined as

$$\sigma_D(\kappa_{i,j}, \sigma_A, \sigma_R) = \begin{cases} \sigma_A - \kappa_{i,j} & \kappa_{i,j} < \sigma_A \\ 0 & \sigma_R \geq \kappa_{i,j} \geq \sigma_A \\ \sigma_R - \kappa_{i,j} & \sigma_R < \kappa_{i,j} \end{cases} \tag{4.3}$$

This distance is used to compute the *virtual spring* force generated by the constraint.

In this section we have formally defined the state of the robot network. Based on

the communication network an MST control network is calculated. The MST control network will be used by the robots to keep the communication network fully connected. The following section presents several heuristics to calculate the MST control network. Afterwards the task and goals for the proposed behavioural roles are described. The chapter finishes with a summary of the contributions of the proposed approach.

4.4 Formation and Maintenance of the MST Control Network

The multi robot system is comprised of *n* robots that start off in known locations. The robots have a local communication system and communicate periodically. As the robots move through the space they form a mobile *ad hoc* network in which the robots are nodes in the network and can act as routers to relay information through the network; this network is dynamic. The robots periodically transmit beacon signals to their direct connections. The beacon signals contain the estimated position of the robot and its uncertainty. The beacon signals are used to determine the network status of the robot. Each robot adopts a particular behavioural role according to its network status. To build the initial MST control network it is required that each robot have a direct connection to at least one other robot.

The MST control network is calculated at the start of the exploration. A designated[8] robot gathers the information required to build the MST control network. The robots remain in their initial positions until they receive the MST control network. The initial behavioural role for the robots is the Maintainer role. The MST control network is calculated using a heuristic and the connection status of the network. Several heuristics are proposed and compared in this thesis. The heuristics are based on the robots' estimated positions and the signal quality. The MST control network can be modified or rebuilt by the robots over time. In the first implementation of BERODE the robots never rebuilt the complete control network they only rebuilt their local network. The local network for a robot is the network that contains all the robots within a *k-hop* distance. Each robot in the network has a different local network. In the second implementation of BERODE the robots periodically rebuilt the complete control network because some special circumstances with respect to multiple Explorers turned out to require this (Section 8.7, page 246). Merging and validation mechanisms are necessary

[8] In the experiments the robot with the lowest ID number is the designated robot.

to ensure that the robots maintain the consistency of the MST control network (Section 4.4.3). The network is calculated only when there are no robots in the Recoverer behavioural role (i.e. no broken control connections).

The MST control network serves two purposes: Control and Information Distribution. For the purposes of control the network is used by the robots to keep them forming a single connected network. For the purposes of Information Distribution, the network is used to distribute relevant knowledge (e.g. extracted features from the robot sensors) among the robots in an intelligent fashion to avoid communication bandwidth problems. In BERODE each robot has to distribute its information. Information is used to produce coordinated behaviours. Coordinated behaviours are produced when robots share consistent maps of the environment. Consistent maps are maps that have integrated the same information about observed features. These maps usually have small metric differences, but the general topology of the map is the same.

One of the main goals in BERODE is scalability. A hierarchical approach to distribute information is proposed. Information is distributed to a point in which the robots share commoned maps of the environment. Commoned maps are maps that have integrated a common subset of information about observed features. The robots have to share consistent feature maps of the environment at the end of the exploration. However, we argue that it is not necessary that the robots share consistent feature maps during the whole exploration to achieve the coordinated exploration of the environment. Instead it is only necessary that the robots have common feature maps to achieve coordinated behaviours. Robots that are close (based on their *k-hop* distance) need to have more in common between their feature maps than robots that are distant because close robots need to achieve coordinated motion to avoid disconnections. This is not the case for distant robots.

Based on this argument, the robots initialize the exploration by building the MST control network to determine their network status. After the initial process is executed the robots retain knowledge of their local network. The local network is used by a robot to distribute its information to close robots (based on their *k-hop* distance) with a frequency (f_1). The MST control network is used to distribute the information to all the robots in the network with a frequency (f_2) where $f_2 < f_1$. The details of distribution of information are presented in Section 5.10 (page 130). Chapter 9 presents simulations that show the trade-off between the communication costs and other variables such as the size of the local network.

4.4.1 Calculation of the MST Control Network

As explained in the previous section the MST control network is calculated at the start of the exploration with an initialization process carried out by a designated robot. During the exploration process the robots may opt to modify or rebuild the MST control network. This section describes the general calculation procedure. The following section describes the procedure to recalculate partially the MST control network which derives from the procedure explained in this section.

For the purposes of calculation of the MST the robot network is seen as a graph where the robots are the nodes and the edges are the direct connections between the robots. Each edge has a weight, which is a value representing how unfavourable it is, and the algorithm uses this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. The MST is then a spanning tree with weight less than or equal to the weight of every other spanning tree. The MST is the safest spanning tree in terms of communication.

A modified version of Dijkstra's algorithm is used to build the MST (Cormen et al., 1990). Dijkstra's algorithm iteratively adds the edge with the smallest accumulative path that connects a *visited* node with a *non-visited* node. Every time an edge is added the *non-visited* node is labelled as *visited*.

The modification consists of adding the edges that are connected to an Explorer node to the initial MST. The algorithm starts by adding to the initial MST the minimum weight edge for every Explorer[9] node. The Explorer nodes are marked as visited as their minimum weight edge is added. Adding initially the edges of the Explorer nodes to the MST and labelling only the Explorer nodes as visited nodes guarantees that the graph is an MST and that Explorer nodes have only one connection in the MST. The purpose of this modification to Dijkstra's algorithm is to try to keep the robots that were in the Explorer role before the MST calculation in their behavioural role thus dimishing the disturbance to the exploration task. The Explorer robots thus maintain their progress of their exploration task. Our modification gives the MST assuming that the Explorer behavioural roles are held constant. If there are no Explorer nodes the standard version of Dijkstra's Algorithm is applied.

After the initial process is carried out the standard version of Dijkstra's Algorithm is applied. In our version of Dijkstra's algorithm we add the edge with the smallest weight rather than the edge with the smallest accumulative path because our concern

[9] An Explorer node is a robot in the Explorer behavioural role. An Explorer robot has only one connection in the MST.

is to optimise communication quality rather than path length. This does not affect the way that the algorithm operates.

Heuristic	**Information required for the heuristic**	**Weight for the edge between robots R_i and R_j**
Connectivity	Roles of the robots, number of direct connections for each robot	$w_{i,j} = \left(\lvert\varepsilon_i\rvert + \lvert\varepsilon_j\rvert\right)/2$
Position	Roles of the robots, estimated position of each robot	$w_{i,j} = \left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{1/2}$
Pos. predictive	Roles of the robots, estimated position of each robot	$w_{i,j} = \sigma_D(\kappa_{i,j}, \sigma_A, \sigma_R)$ (in m)
QS Predictive	Roles of the robots, signal quality for the connections	$w_{i,j} = \sigma_D(\kappa_{i,j}, \sigma_A, \sigma_R)$ (in dB)

Table 4.1: Calculation of the weight for the edge $w_{i,j}$ between the robots R_i and R_j for the four proposed heuristics using the terminology from Section 4.3. $|\varepsilon_i|$ is the number of direct connections for the robot R_i. The robot R_i has an estimated position $P_i(x_i, y_i)$. $\kappa_{i,j}$ is the signal quality for the connection which can be estimated using the positions of the robots (in m) or can be obtained from the hardware communication device (in dB). σ_D is the discomfort distance which is the distance between the signal quality for the connection and a desired signal quality. σ_A and σ_R are the attractive/repulsive thresholds for calculating the discomfort distance using Eq 4.3.

As previously explained the initial behavioural role for the robots is the Maintainer role. In the initialization process the designated[10] robot gathers the information required to build the initial MST control network. If during the exploration process a robot decides to update the MST control network the robot gathers the information required to build the MST control network. The information gathered to build the MST control network is used as the weights in the calculation of the MST control network. Four weighting heuristics have been proposed and compared experimentally in Chapter 9. Table 4.1 presents the formulas to calculate the weight for the connection $w_{i,j}$ between the robots R_i and R_j for the different heuristics.

In the Connectivity heuristic the weight of an edge is the average number of connections between the pair of nodes that form the edge. When two edges have the same value the priority is Pusher $\longrightarrow$ Maintainer $\longrightarrow$ Explorer. If the roles are the same the robot with the lowest ID number is used as the final criterion. This heuristic is designed to try to constrain every node (robot) to the least number of nodes, and is based

[10] In the experiments the robot with the lowest ID number is the designated robot.

on the assumption that the fewer constraints a node (robot) has to satisfy the better it can accomplish its purposes.

In the Position heuristic the weight of an edge is the distance between the positions of the robots. This heuristic is inspired by the Travelling Salesman Problem. It is expected that robots that are close to several robots will have them as constraints. Close robots are likely to be in their comfort regions. The robots will tend to move less in trying to improve the overall signal quality (OSQ).

The Pos. predictive and QS predictive heuristics are calculated in the same way. The difference is in the values used as weights. The Pos. predictive heuristic uses the robots' positions to estimate the signal quality (Section 4.3), while in the QS predictive heuristic the *RSSL* is used as the signal quality. The QS predictive heuristic is only applicable to *RF* technologies.

In the Pos. predictive heuristic the weight of an edge is the discomfort distance (Eq. 4.3) between predicted positions for the robots. Predicted positions are based on the assumption that by moving half the discomfort distance in the direction of each other pairs of robots will reach the boundary of their comfort zone. Figure 4.5 shows the predicted positions P_i and P_j for robots R_i and R_j respectively in the cases of attraction and repulsion. In the case of attraction the robots are predicted to move towards each other while in the case of repulsion the robots move away.

Once the edge is added to the MST the discomfort distance is stored and used to recalculate the weights of the edges for the graph. At each iteration the weights are updated based on the predicted positions for the robots. In general the predicted position P_i for a robot R_i with k connections in the current MST is calculated by successively updating the predicted position for the k connections. The connections $MST_i(k) = \{D_1, D_2..., D_{k-1}, D_k\}$ are ordered incrementally with respect to their discomfort distances ($D_1 < D_2 < ... < D_{k-1} < D_k$).

Figure 4.6 presents an example of the first three iterations using the Pos. predictive heuristic. At the start of the process the predicted positions for the robots are their original positions. $D_{i,j}$ is half the discomfort distance between the predicted positions for robots i and j. In (a) the smallest edge $E_{3,4}$ is added having R_3 as origin. In (b) after the predicted positions are calculated for R_3 and R_4 the weights for the graph are updated. It can be observed that for the original positions the smallest accumulative path to a visited node is obtained by adding the edge $E_{2,3}$. Based on the predicted positions the edge to add is $E_{1,3}$ instead of $E_{2,3}$. In (c) the predicted position for robot R_3 is calculated by iteratively updating its predicted position for its two MST

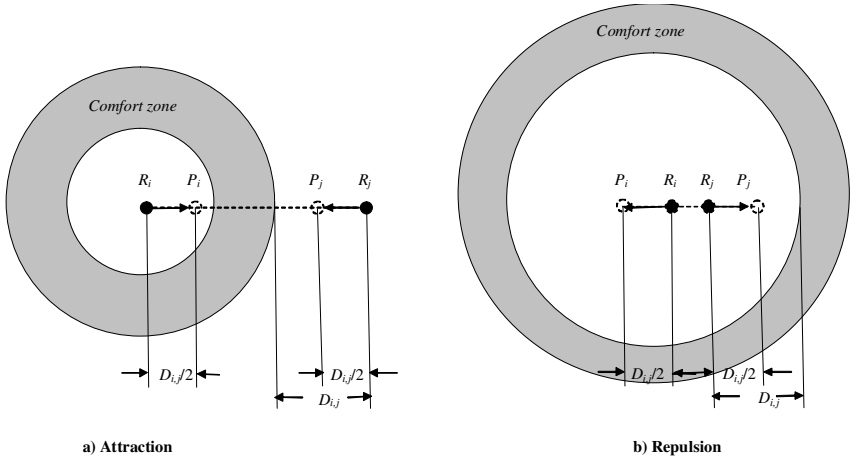


Figure 4.5: Robots R_i and R_j are predicted to move to positions P_i and P_j respectively. In the case of (a) attraction robots are predicted to move towards each other to decrease the discomfort distance by $D_{i,j}/2$, while in the case of (b) repulsion the robots are predict to move away.

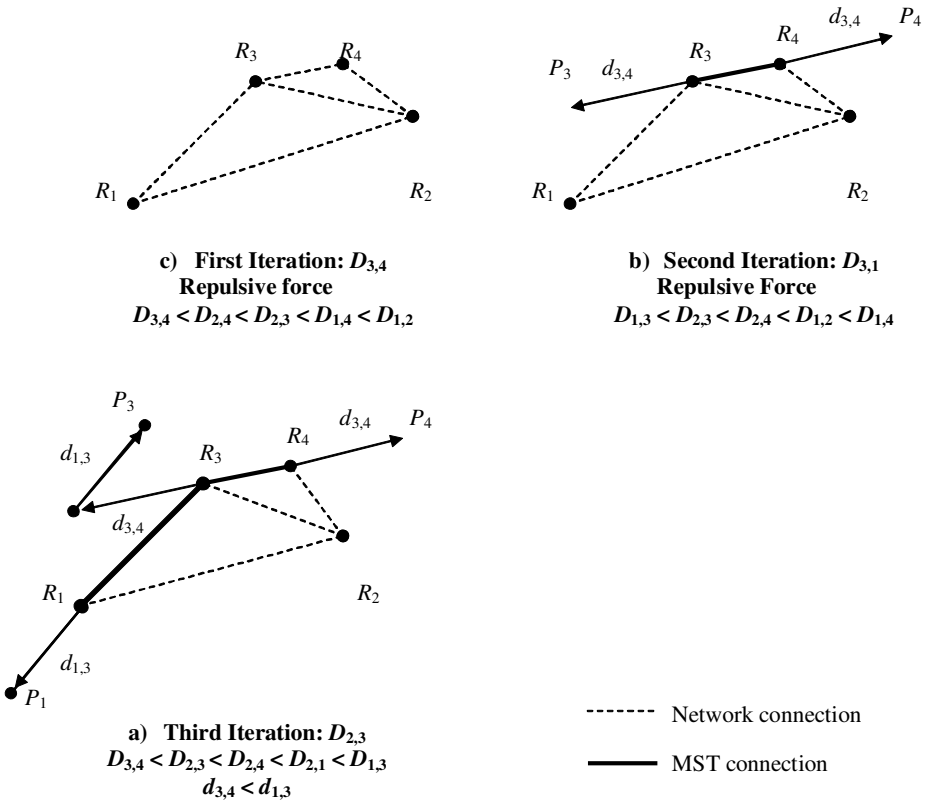


Figure 4.6: Robot positions for the predictive heuristic. (a) The smallest edge $E_{3,4}$ is added to the MST, (b) The smallest edge $E_{1,3}$ based on the predicted positions for robots R_3 and R_4 (P_3 and P_4 respectively) is added, (c) The smallest edge $E_{1,3}$ based on the predicted positions for robots R_3 and R_4 (P_3 and P_4 respectively) is added.

connections as explained previously. The edge to add based on the predicted positions is $E_{2,4}$.

4.4.2 Building a Local Control Network

After the initial MST is calculated the robots retain knowledge of their local network. The local network for a robot is the network that contains all the robots within a *k-hop* distance. Each robot in the network has a different local network. Robots can modify their local network to try to improve the network conditions. This process is referred to as a local network event. For instance, when a pair of robots becomes within range a new direct connection is formed and a local network event is generated.

Figure 4.7 presents a common scenario for *LOS* communication where the benefits of recalculating the MST control network are illustrated. Robots $R_{2,E}$ and $R_{4,E}$ are robots in the Explorer behavioural role that are attracted to the unexplored space. At time t_2 (Figure 4.7 (b)) $R_{4,P}$ has changed its role because it cannot explore the area without going out range of $R_{3,M}$ and has come within communication range of $R_{2,E}$. At time t_3 (Figure 4.7(c)) the network is recalculated. It is observed that the edge $E_{2,4}$ replaced the edge $E_{2,3}$ on the MST control network. The recalculated network is a better network because it will allow $R_{4,E}$ to keep exploring the unknown space without losing its control connection. This is not the case if the previous MST control network was maintained. In that case after a certain time has passed $R_{2,E}$ and $R_{4,E}$ would give up exploring the area because they cannot sense the unexplored space without losing their control connections. Although later in the exploration process the robots would return to finish the exploration of this area so the time to complete the exploration would increase. The recalculation of the MST control network improves the exploration process.

Robots in the network are identified with incremental ID names. When a new direct connection is formed the process to build the local network is carried out by the robot with the lowest ID number. This robot is referred as the MST builder. Figure 4.8 shows the algorithm used by a robot when it detects a new connection and tries to update the local network.

A robot starts the recalculation process by comparing its ID number with the ID number of the other robot. The robot with the smallest ID is the robot that recalculates the local network and is referred as MST builder. The robot with the highest ID number transmits its current local network and behavioural role to the robot with the

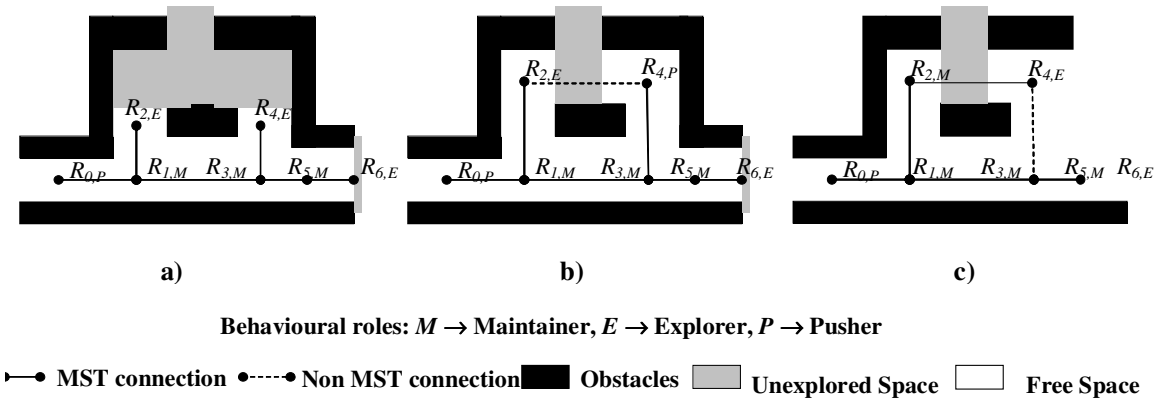


Figure 4.7: A common *LOS* communication scenario. a) At time t_1 robots R_2 and R_4 explore the space, b) at time t_2 connection R_2-R_4 is discovered, c) R_2 rebuilds its local network and its behaviour is now Maintainer.

lowest ID number. Upon reception of this information the MST builder checks if the received local network is consistent with its local network. The pair of local networks is consistent if they have at least one common edge and at least one of the robots from the new direct connection is not in the Explorer behavioural role. If the pair of local networks is not consistent the recalculated local network is inconsistent with the MST control network. If the inconsistent local networks are used to update the MST control network this network is no longer an MST. Appendix A demonstrates the consistency property. When the local networks are inconsistent the calculation process is aborted by the MST builder by sending an inconsistency message to the other robot that forms the new connection and the robots resume to their current roles.

Once the consistency test has been successful the MST builder transmits a request message (Section 5.10, page 130) to gather information relevant to the current MST heuristic. The joint local neighbourhood formed by the pair of local networks has (k+1) levels and is referred as the joint local network. The request message is then transmitted within the joint network. Robots that received the request message remain static to avoid disruption of the process. When the MST builder sends the request message it initializes a timer and waits for a certain time (user defined) to get the information necessary to recalculate the joint network. The timer is necessary to avoid the situations where two or more joint networks that overlap[11] are calculated simultaneously. A robot that receives multiple queries to build different joint networks will send its information only to the recalculation process that started first. This avoids inconsis-

[11] The joint networks overlap if they have at least one robot in common.

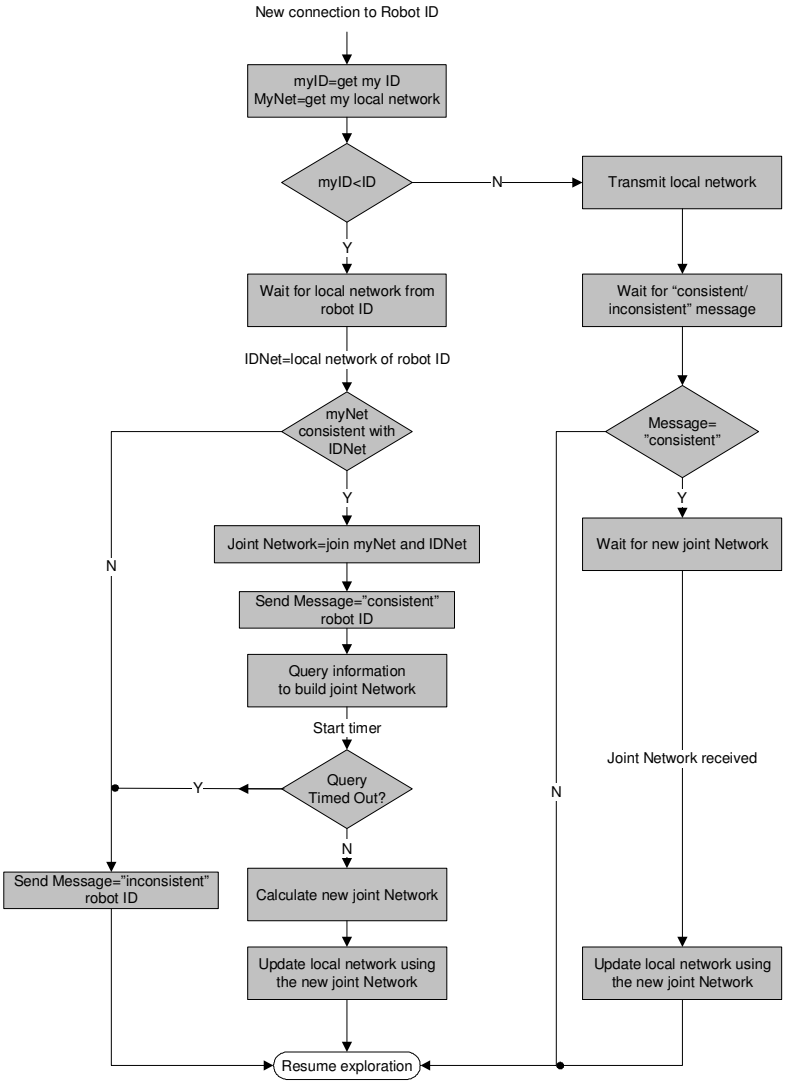


Figure 4.8: Flow diagram to update a local network for a robot when it detects a new connection with the robot named ID. The robot with the lowest ID number carries out the updating process. This robot checks that the local networks of the two robots are consistent. If the networks are consistent a new joint network is built. The robots update their local networks with the updated joint network and resume to the exploration process.

tency problems because if more than one joint network is calculated in parallel there is no guarantee that the MST control network will be consistent (a minimum spanning tree). If the process of querying information times out the recalculation process is aborted. The MST builder sends an inconsistency message to the other robot that forms the new connection and the robots resume to their current roles.

If the process of querying information obtains all the information before the operation times out the MST builder calculates the new joint network according to the current MST heuristic. At the distribution stage the MST builder transmits the new joint network to the robots inside this joint network. The robots inside the joint network receive the network and update their local network using the joint network. A robot obtains its updated local network from the joint network by building a tree of control connections with it as the root and removing the connections above the *k* level of the tree.

Figures 4.9 and 4.10 show the update process for a local network when a new connection is detected for the scenario from Figure 4.8. Figure 4.9 show the process when the local network size is *k*=2. It is observed that because the robots R_2 and R_4 share a common edge $R_1 - R_3$ (Figure 4.9b) the recalculation process can be carried out. The joint network size is *k*=3 for R_2 (Figure 4.9c, number of hops to R_4). R_2 recalculates the joint network; the recalculated network is shown in Figure 4.9d. R_2 updates its local network using the joint network (Figure 4.9e) by removing the connections above the *k* level ($R_{5,M}$) and updates its role to Maintainer role (Section 5.4, page 98). R_4 receives the joint network, updates its local network by removing the connections above the *k* level of the tree (R_0, R_3 and R_5) and reselects its behavioural role which is now Explorer (Section 5.4, page 98).

Figure 4.10 illustrates the process when the local network size is *k*=1. It is observed that in this case the update process is aborted because the robots R_2 and R_4 do not share a common edge.

4.5 Tasks and Goals for the Behavioural Roles

We designed four behavioural roles to achieve the exploratory behaviour of the network. These roles are Explorer, Maintainer, Recoverer and Pusher. The BERODE architecture is implemented in the robots through modules. The behavioural roles are a combination of planning and reactive modules. Reactive modules avoid collisions between the robots and the environment. Planning modules generate reactive movement

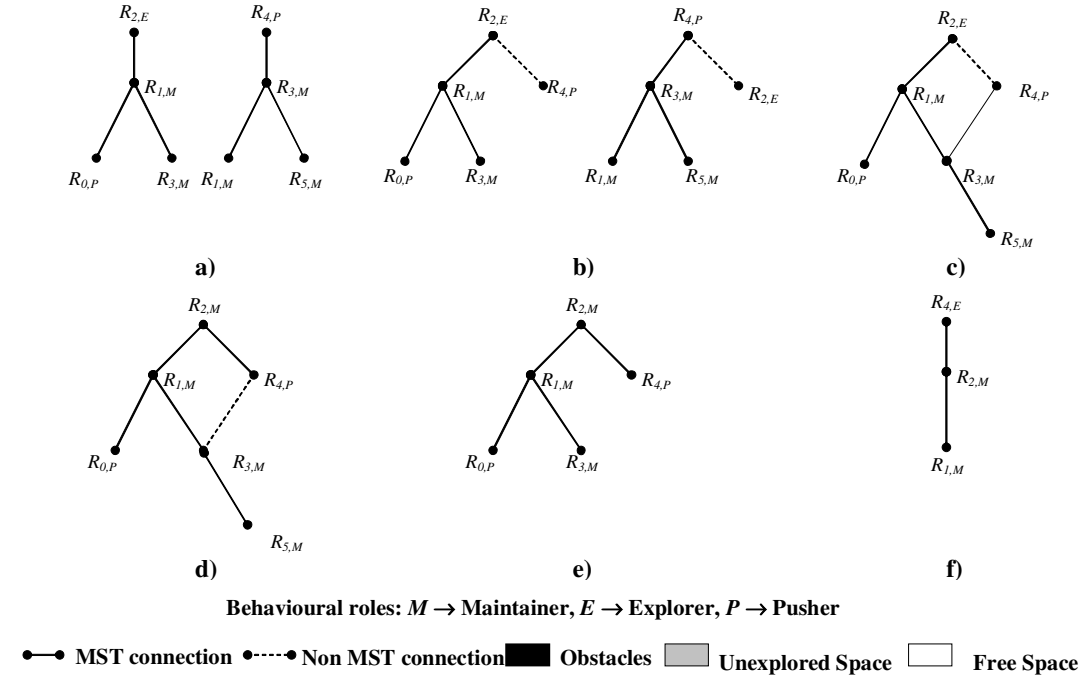


Figure 4.9: Building a local network for robots R_2 and R_4 with local network size k=2. a) The local network for robots R_2 and R_4 for the network from Figure 4.8a. b) A new connection between R_2 and R_4 is detected, R_4 sends its local network to R_2, R_2 verifies that the networks are consistent (they have at least one common edge R_1-R_3. c) R_2 requests information from the formed joint network. d) R_2 recalculates the joint network and transmits the updated joint network to the robots inside this joint network. e) R_2 updates its local network using the joint network. f) The local network for R_4 after the joint network was received and used to update the local network.

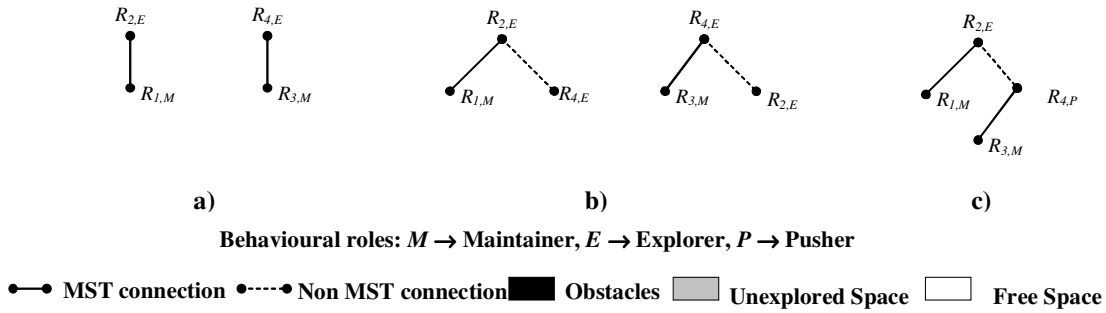


Figure 4.10: Building a local network for robots R_2 and R_4 with local network size k=1. a) The local network for robots R_2 and R_4 for the network from Figure 4.8a. b) A new connection between R_2 and R_4 is detected, R_4 sends its local network to R_2, R_2 verifies that the networks are consistent. c) R_2 aborts the recalculation process because there is no common edge between the pair of local networks.

plans based on the predicted communication safety. A robot reselects its behavioural role when the MST control network is modified by itself or another robot, or when the robot has reached its current goal. Each behavioural role has a different degree of compromise towards the task of network maintenance. The goals for each behavioural role are:

Behavioural Roles

1. **Explorer**: Move towards unexplored areas while remaining in the safe level.
2. **Maintainer**: Move towards the position that maximizes the overall signal quality (OSQ) for the communication constraints.
3. **Recoverer**: Move towards a position where signal quality is in the safe level for the communication constraints.
4. **Pusher**: Move towards the position that maximizes the OSQ for its communication constraint.

The goal for the Maintainer, Pusher and Recoverer behavioural roles is essentially the same. The difference is that the parameters of the *virtual spring* model are a function of the behavioural roles of the robots (Section 4.3.3). This model is used to determine the location that maximizes the OSQ (Section 4.3.3). This parameterization based on roles generates local interactions that induce *leader–follower* motions in the robot network where the Explorers direct the exploration and the Pushers accelerate the movement of the robot network in the exploration directions.

The first three roles are based on previous research on the areas of network recovery and maintenance (Section 2.3, page 24). In previous research in network maintenance (Thibodeau et al., 2004) there is only one robot that explores (Explorer) the space while the rest of the team (Maintainers) keeps the network connected.

In BERODE several robots may decide to explore the space at the same time. This speeds up the exploration process but under certain circumstances conflicts arise. The Pusher role is designed to solve these conflicts. Instead of having two heads (Explorers) pulling in different directions one becomes an active tail: a Pusher. An example of a typical conflict is depicted in Figure 4.11. Robots $R_{0,E}$ and $R_{2,E}$ move towards f_0 and f_1 respectively behaving as Explorers. R_1 behaves as a Maintainer. In Figure 4.11(a) the robots are at the maximum distance where the network remains connected. $R_{0,E}$ and $R_{2,E}$ wait for a certain time for improvement in the network conditions. After

a certain time (Figure 4.11(b)) $R_{0,P}$ decides to give up exploring and transitions to the Pusher behavioural role. $R_{0,P}$ induces motion in the network by closely following $R_{1,M}$ (Figure 4.11(c)). The motion effect is achieved by having different free spring lengths for the v*irtual spring* forces within the pairs of robots. The free spring length depends on the behavioural roles adopted by the pairs of robots as well as the signal quality.

A Pusher robot is the result of the lack of unexplored areas where the communication coverage is estimated to be in the safe level (Section 4.2.1). Due to the dynamic nature of the network and the exploration over time, unexplored areas in the safe level of communication are discovered. The Pusher robot can then transition to Explorer.

In real world scenarios signal propagation is unpredictable and time varying. This could lead to jittering transitions between the Explorer and Pusher behaviours. Additionally it was found in experimentation that robots with communication constraints to robots that transition from the Pusher to Explorer behavioural role generated large modifications in their reactive plans. These large modifications were caused because the forces depended on the behavioural roles. Large modifications generated local minima for the OSQ that deteriorated the performance[12] of the robot network.

This kind of jittering problem is typically addressed in one of two ways. The first one is setting different thresholds for transitioning into and out of a role. The second one is to set a minimum role time. Setting different thresholds does not always solve the jittering problem because under certain circumstances the communication conditions change very fast. For instance, when the robots use *LOS* communication if a robot that has *LOS* to other robot moves a small distance but goes behind a wall the communication is lost. As soon as the robot backtracks this small distance the communication conditions change drastically. Therefore the minimum role time solution is the most suitable for BERODE because the issues of jittering and local minima can be solved by setting a minimum role time that addresses these problems as follows:

- Jittering problem: Once the robot transitions to the Pusher behaviour the transition to the Explorer role is inhibited for a certain minimum time. This time is referred as expiration time. After this time has passed the robot may transition to the Explorer role. The Pusher robot transitions to Explorer only if the conditions for the Explorer role are satisfied (Section 4.5). A Pusher robot may also transition at any time[13] to either Maintainer or Recoverer behavioural roles if a

[12] The performance of the robot network is the time to build a complete map of the environment.
[13] Including the time within the expiration time.

network event occurs or the conditions for these roles are satisfied.

- Large modifications: Forces that involve robots in the Pusher behavioural roles are redefined in terms of the expiration time. The attractive/repulsive thresholds for the forces are then gradually modified from the Pusher to the Explorer thresholds. The reactive plans generate small modifications because of the gradual variation of the thresholds.

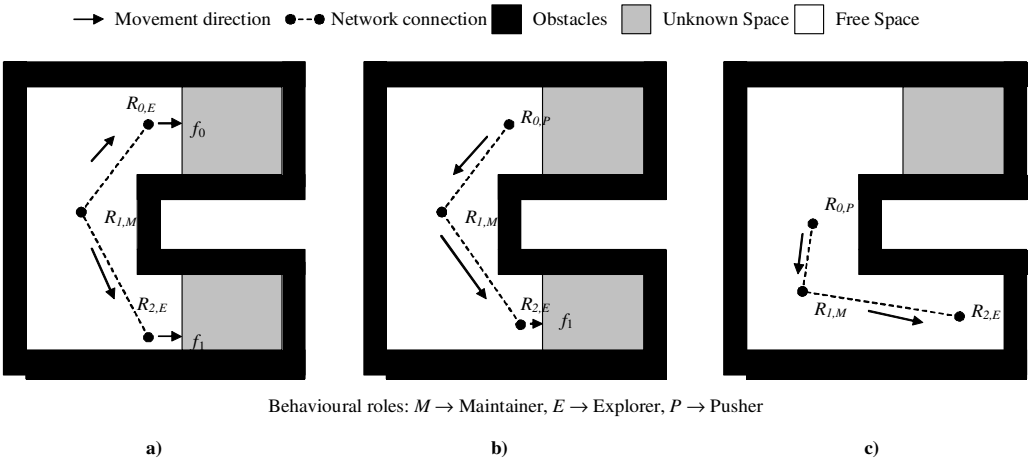


Figure 4.11: An example of decentralized conflict resolution. a) Robots $R_{0,E}$ and $R_{2,E}$ explore the space while $R_{1,M}$ maintains the network. b) Robot $R_{0,P}$ transitions to Pusher since the exploration area is not safe. c) Robot $R_{0,P}$ induces motion into the network by closely following to robot $R_{1,M}$.

4.6 Conclusions

This chapter has presented the purposes and design of the BERODE (BEhavioural ROle DEcentralized) architecture. The BERODE architecture is a decentralized architecture to explore environments using multiple robots with local communication systems. In BERODE the robots remain in communication range to coordinate and efficiently explore the environment. Robots coordinate by adopting behavioural roles that reactively adapt to the dynamic MANET that they form as they traverse the environment. According to its role a robot exhibits a varying degree of interest in the exploration task.

An MST (minimum spanning tree) control approach was presented. The robots build an MST control network of the communication network and keep the MST con-

trol network by adopting a certain behavioural role. The behavioural roles generate reactive plans that ensure the connectivity of the robot with its direct connections on the MST control network.

A novel heterogeneous *virtual spring* model was introduced. The virtual springs are described as heterogeneous because they generate asymmetric[14] forces and their free spring length is a range of values rather than a single value. The free spring length is a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection. This model is used in the generation of the reactive plans where the direct connections of the robot impose *virtual forces*.

A two level approach to communication was introduced. The approach is based on the argument that the degree of required coordination for a pair of robots is related to their *k-hop distance* on the MST control network. In the approach most of the communication is retransmitted within a small number of hops thus avoiding scalability problems as the number of robots in the architecture increases.

In BERODE each robot builds and updates its own feature based map of the environment using an Extended Kalman Filter (*EKF*). As the robots explore the environment they observe new features and use these observations to update their estimates about the robot and feature locations. The robots share the same global Cartesian frame of reference. The robots periodically distribute their observed features with the team of robots. Robots incorporate the received features to their maps using the same process as for locally extracted features. The exploration is considered to be completed once any one robot has projected its feature map into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a used defined threshold (Section 8.6, page 239).

The main features of the approach are:

- MST control network: Control based on the MST control network minimizes the number of communication connections that have to be maintained thus minimizing the constraints imposed on the robots.

- Heuristics for the MST control network: Several heuristics to calculate the MST control network have been proposed. These heuristics are based on the robot positions, the number of direct connections for the robots and the predicted positions and signal quality.

[14] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

- Hierarchical communication levels: The degree of coordination required for a pair of robots is related to their *k-hop* distance on the MST control network. To get a good trade-off between coordination and communication costs two levels of communication are proposed: local and global. Information is shared frequently at the local level, while at the global level information is shared less frequently.

- Adaptive MST control: The topology of the MST is adapted as the robot network traverses the environment. The topology is adapted by the robots either partially or globally. Adaptability allows the robots to remain in communication range while maintaining a high level performance in the exploration task.

- Role Based Distributed Control: Robots adopt behavioural roles adapt to the dynamic conditions of the *MANET*. The roles balance between the tasks of exploration and network maintenance.

- Heterogeneous *virtual force* model: The communication constraints of the robots exert heterogeneous virtual forces. Heterogeneous forces aid the exploration process because they induce a movement effect towards the unexplored areas of the environment.

- Reactive Predictive Planning: Non-Explorer robots generate reactive plans based on predictions for the signal quality. These short term plans are more adequate for dynamic robot networks in real world scenarios than purely reactive approaches because of the unpredictable nature of wireless communication in indoor environments.

Chapter 5

The Implementation of the BERODE Architecture in Individual Robots

5.1 Introduction

The BERODE (BEhavioural ROle based DEcentralized) architecture is a decentralized architecture to explore environments using a group of robots with short range communication. In BERODE the robots are kept as a single connected network. This improves coordination and minimizes the overlapping of tasks for the group of robots. The BERODE architecture is based on behavioural roles such as Explorer and communication Maintainer. These roles reactively adapt to the dynamic conditions of the communication network formed by the robots as they explore an environment.

The previous chapter described the purposes and design of the BERODE architecture. This chapter describes the implementation of the BERODE architecture in individual robots. The architecture has been implemented using modules that are sequentially executed. The implementation and parameterization of some of the modules depends on the behavioural role. This introductory section first summarizes the design of the BERODE architecture and then describes the modules used in the implementation of the control architecture.

The BERODE architecture was designed to obtain the exploratory behaviour of the robot network through the interaction of the individual roles. This interaction is achieved through the imposition of *virtual spring* forces for the connections in the MST control network. The MST control network is a minimum spanning tree of the communication network that contains only the necessary connections to keep the communication network connected. The *virtual spring* forces keep the robot network con-

nected while moving towards unexplored areas. This is achieved by using heterogeneous forces. The forces are described as heterogeneous because they are asymmetric[1] and their free spring length[2] is a range of values rather than a single value. The magnitude of these forces is a function of the quality of the connection and the behavioural roles of the robots that form the connection. The Explorer robots are attracted towards unexplored areas dragging the other robots in the network behind them.

The robots use the MST control network to distribute and gather relevant information to achieve coordination (e.g. extracted features from the robot sensors). The MST control network is calculated at the start of the exploration by a designated[3] robot. This robot distributes this network among all the robots. The robots retain knowledge of their local network. The local network for a robot is the network that contains all the robots within a *k-hop* distance on the MST control network. The robots distribute their information at two levels: frequently to the robots inside their local network, and less frequently to all the robots. Each robot generates information and retransmits information received from other robots.

The robots periodically transmit beacon signals. The robots are assumed to be able to measure the signal quality of the beacon signals. The signal quality measurements depend on the hardware implementation. For instance, in the implementation for *RF* technologies we use the *RSSL* (Received Signal Strength Level) which is an available value for this technology (measured in dB) as the signal quality value.

The beacon signals contain the estimated position of the robot and its uncertainty. In BERODE each robot builds and updates its own map using an Extended Kalman Filter (*EKF*). The positions of robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[4] number. The *EKF* produces an estimate (along with its uncertainty) of the location of the features and the robot. As the robots explore the environment they extract new features from their sensor measurements. Line and point features are extracted from the raw sensor data. A geometric representation of the features is obtained. The features are represented by a measurement vector and an uncertainty matrix. These features are used by the *EKF* to update the estimated locations. The extracted features are periodically distributed among the team of robots. Robots incorporate the received

[1] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[2] The free spring length is the length for which the spring exerts a null force.

[3] In the experiments the robot with the lowest ID number is the designated robot.

[4] The robots have an ID number that allows them to identify each other in the network.

features to their maps using the same process as for locally extracted features. This is possible because the robots use the same global Cartesian frame of reference. The exploration process stops once any one robot considers the map as complete[5].

The robots select their behavioural role based on their network status and their internal state. The network status of a robot comprises the safety level and the set of communication constraints. The safety level for a robot is determined by the signal quality of its connections on the MST control network. A robot is on a safety level if all of its connections have at least that safety level. BERODE implements three safety levels with the following decreasing order in safety: safe, precautionary and unsafe. Depending on the safety level a set of communication constraints is imposed. In the safe level the set of communication constraints is empty because the signal quality for the control connections is above the safe threshold (σ_{safe}). In the precautionary level the set of communication constraints is formed by the control connections. In the unsafe level the set of communication constraints is formed by the control connections for which the signal quality is below the precautionary threshold (σ_{prec}).

The behavioural roles balance between the tasks of exploration and network maintenance. A robot in the BERODE architecture exhibits one of the following behavioural roles: Explorer, Maintainer, Pusher and Recoverer. The Explorer and Maintainer behavioural roles are the basic behaviours for the robots in the network. The Explorer focuses on the exploration task while the Maintainer robots focus on keeping the robot network connected. The Pusher behavioural role helps the exploration task when it becomes problematic; for instance when two Explorer robots pull the robot network in opposing exploratory directions the exploratory behaviour of the robot network is halted. One of the Explorer robots gives up its exploration task and transitions to the Pusher behavioural role. The Pusher robot then induces movement in the robot network by pushing its direct connections away from it. The Recoverer robot helps the communications maintenance task when it becomes problematic. A robot exhibits this role when the level of safety is unsafe. The Recoverer role tries to improve the signal quality for its unsafe connections.

The non-Explorer robots keep the communication network connected by generating plans to move to locations where the energy from the *virtual spring* forces is minimized. The Explorer robots are not directly subject to *virtual spring* forces; in-

[5] A map is considered to be complete once it is projected into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a used defined threshold (Section 8.6, page 239).

stead these robots monitor the level of safety for their control connection and wait for improvement if necessary to minimize the risk of becoming disconnected. The Explorer robots guide the exploration process by generating a plan to move to the safest unexplored area in terms of communication quality and moving in that direction to the limits of their communication safety.

The robots reselect their behavioural role once an event has occurred. An event is generated when a robot has reached its current goal or modified its local network. A robot tries to modify its local network when it detects that a new connection has been formed. The local network is modified when its modification does not generate an inconsistent MST control network. When a robot modifies its local network this network is transmitted to the robots inside the local network. The robots inside the local network update their local network and reselect their behavioural roles. In the original implementation of BERODE the complete control network is never rebuilt. In the second implementation of BERODE the robots periodically rebuilt the complete control network because some special circumstances with respect to multiple Explorers turned out to require this (Section 8.7, page 246).

The following section presents the implementation of the control architecture for the robots in BERODE. The control architecture is formed by modules that address specific tasks (e.g. the Planning Module builds the path that the robot has to follow). The robots' control architecture is the same regardless of their behavioural roles. The modules are sequentially executed. Section 5.2 presents the sequence of execution for the modules and describes how the information is shared between these modules. The modules of the architecture are described in detail in subsequent section as follows:

- Section 5.3 presents the Communication Manager. This module handles the information received from the robot network and maintains a queue of events which is processed when the module is called.

- Section 5.4 presents the Behaviour Selection Module. This module selects the behavioural role for the robots according to the robot's internal state and its network status.

- Section 5.5 presents the Collision Avoidance Module. This module ensures the safety of the robot by detecting collisions with other robots and with static obstacles.

- Section 5.6 presents the Planning Module. This module creates a suitable plan

for the behavioural role of the robot. The plan is communication sensitive which keeps the robot connected to the network. The module generates a path for the plan and delivers the path to the Robot Motion Controller.

- Section 5.7 presents the Robot Motion Controller. The controller generates the motion commands for the robot actuators according to the path planned.

- Section 5.8 presents the Network Manager Module. This module monitors the signal quality of the direct connections of the robot. Depending on the quality of the signals a robot may decide to modify the control network to improve signal quality. The modification of the control network generates an event which is transmitted to the robot network along with the new control network.

- Section 5.9 presents the Map Interface Module. This module builds and updates the feature map of the environment. The feature map is updated using an *EKF* which estimates the location and uncertainty of the robot and the features.

- Section 5.10 presents the application level communication protocol used by the robots to distribute and gather information in the robot network. To get a good trade-off between coordination and communication costs two levels of communication are proposed.

- Section 5.11 presents a summary of the implementation of the BERODE architecture in individual robots.

5.2 The BERODE Architecture in the Individual Robots

The BERODE architecture in the individual robots is the same for all the behavioural roles. The architecture is formed by modules that are sequentially executed. In BERODE the robots assume a certain role depending on their network status and their internal state. The network status is comprised by the safety level and the set of communication constraints. The safety level for a robot is determined by the signal quality of its control connections (connections on the MST control network). A robot is on a safety level if all of its connections have at least that safety level. BERODE implements three safety levels with the following decreasing order in safety: safe, precautionary and unsafe. Depending on the safety level a set of communication constraints is imposed (Section 4.3.2, page 65).

The role selection process is triggered by events. An event occurs once a robot has achieved its current task, detected a change in its safety level or a change in the local or the complete MST control network. Events are either internal or external. Internal events are events that are generated by the robot. Internal events are transmitted if the event is relevant to the other robots in the network. External events are events which are generated by other robots in the network and are received by the robot through the communication device. The robot network is then kept coordinated by means of events.

The architecture was designed as a modular architecture where each module addresses a specific task (e.g. the planning module builds the path that the robot has to follow). Figure 5.1 presents the sequence of execution of the modules in the main control loop for a robot. As previously explained at the beginning of the exploration process a designated robot[6] builds the initial MST control network. The initial MST control network generates the initial event that triggers the exploration process. The modules keep executing forming a loop until the map is considered to be completed.

The robots have a short range communication device. The robots form a wireless mobile *ad hoc* network (*MANET*) as they explore the environment where each robot is in charge of distributing its information. Each robot generates information and retransmits information received from other robots. Most of the information is distributed within the local network improving the scalability with respect to the numbers of robots.

The Communication Manager executes two processes in parallel to the main control loop. These processes do not disturb the execution sequence of the main control loop (Figure 5.1) because these processes are only updating processes. These processes are: periodically sending beacon signals, and retransmission and storing information from the robot network obtained from the hardware communication device. The beacon signals contain the most recent estimated robot position and its uncertainty. These variables are calculated by the Map Interface Module. Information received from the robot network is stored and retransmitted immediately as necessary to minimise communication delays. Information is retransmitted according to the communication protocol from Section 5.10.

A module that uses information stored in the Communication Manager obtains this information at the start of its sequential execution and uses this frozen information throughout all its execution. Information which is updated after a module has started its

[6] The robot with the smallest ID number.

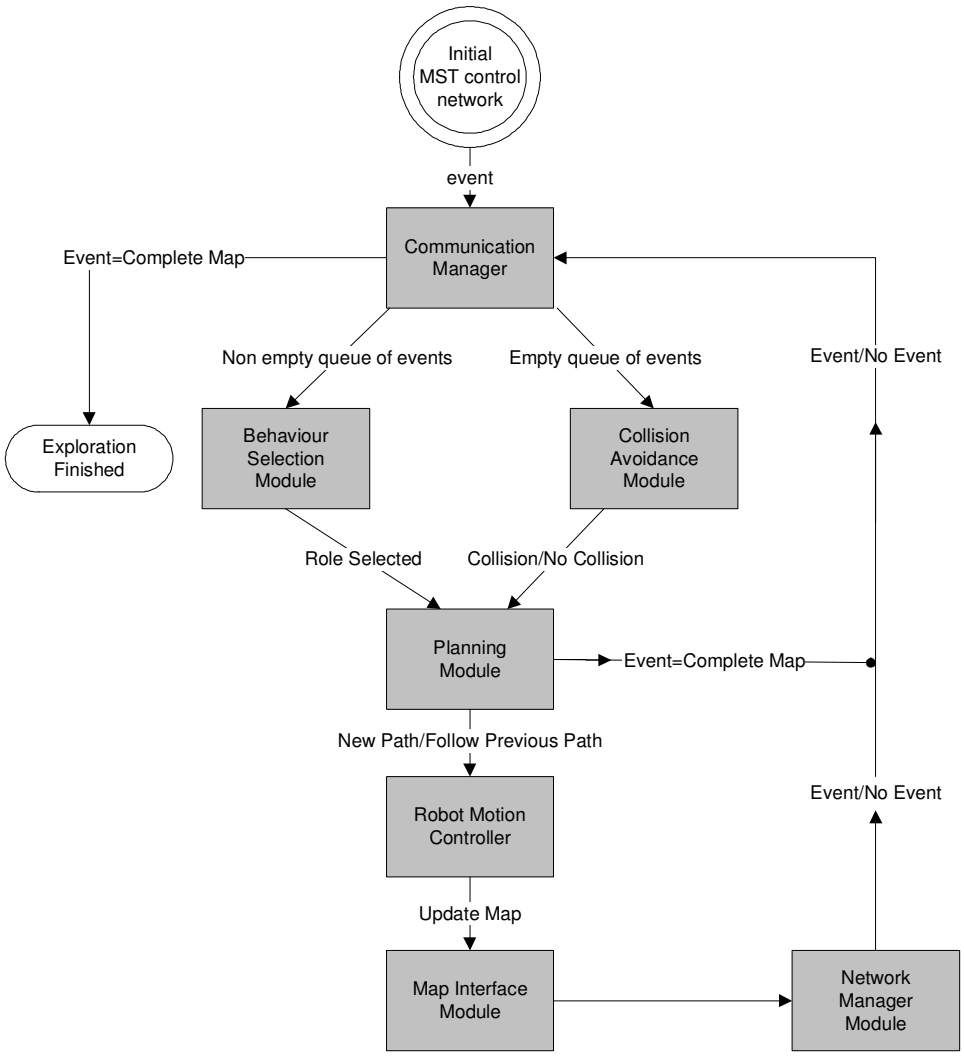


Figure 5.1: Execution sequence for the modules for the robots in the BERODE architecture. The calculated initial MST control network generates the initial event that triggers the exploration process. An event is generated when the suitability of the current behavioural role for the current robot network has to be reviewed. The robot then reselects its behavioural role and generates a plan according to which it moves and updates its map representation. The process stops once the map is considered to be complete by any one robot.

execution is not used by this module to avoid inconsistencies in the internal processing stages of the module.

Figure 5.2 shows the exchange of information between the modules in the BERODE architecture. A module receives the necessary information from other modules at the start of its execution and delivers its output information once it finishes executing.

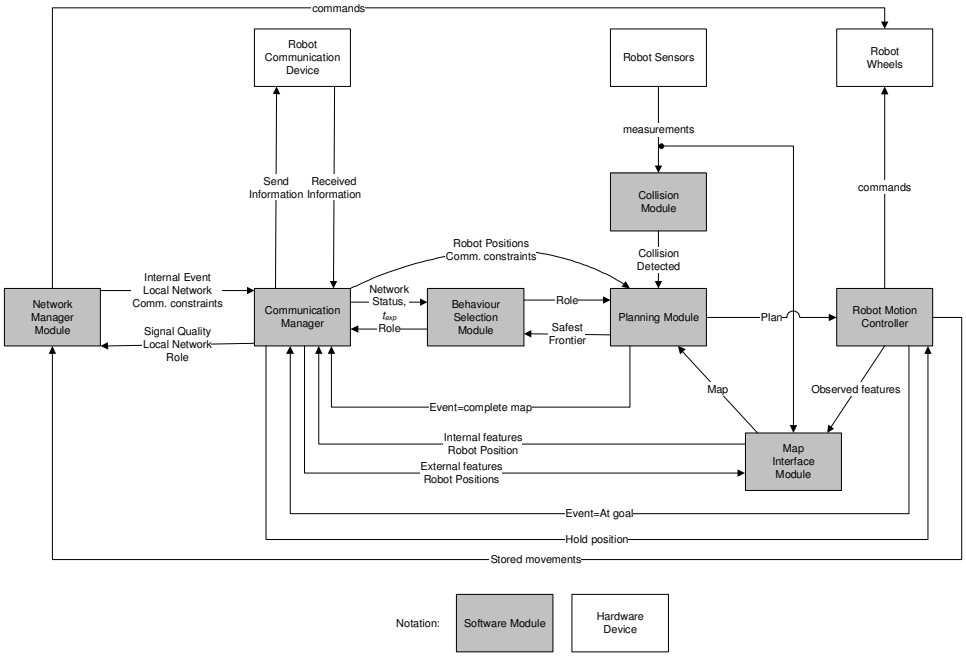


Figure 5.2: Information exchange between the modules in the BERODE architecture and their interaction with the robot sensors. Information between the modules is exchanged by means of messages. The modules execute in sequence as described in Figure 5.1. The modules start their execution by acquiring the relevant information from other modules. Once a module finishes its execution it produces an output which serves as input for the subsequent module. The modules obtain information from the robot hardware (e.g. sensors) and send commands to control the robot.

The Communication Manager is the module that handles the information from the robot network and maintains a queue of the events. Once the Communication Manager is called the events in the queue are processed. The Communication Manager calls the Behaviour Selection Module if the queue of events was not empty. The Behaviour Selection Module selects the appropriate behaviour and calls the Planning Module that creates a plan according to the behaviour selected. When the queue of events is empty the Communication Manager calls the Collision Avoidance Module. The Collision Avoidance Module ensures the safety of the robot by detecting static and dynamic

close obstacles.

The Planning Module is either called by the Behaviour Selection Module or by the Collision Avoidance Module. The Planning Module checks if the current plan is valid and creates a new plan if this is not the case. A new plan is created when: the role of the robot has changed, the robot has completed the current plan or a collision has been detected. The Planning Module is the module that determines when a map has been completed. When the map is considered as completed the Planning Module generates a "complete map" event message and calls the Communication Manager. The Communication Manager transmits this message to the other robots in the network and the exploration process is finished. Upon reception of the "complete map" event the robots stop the exploration process. In a practical implementation the robots might then be required to return to their starting positions. Generally speaking this is not a difficult problem and we have not addressed it in this work.

If the map is not considered as completed the Planning Module calls the Robot Motion Controller. The Robot Motion Controller generates commands to move the robot according to the current plan. The robot collects sensor measurements once the robot has moved a small distance (10cm in our implementation, page 213). The feature map is updated by the Map Interface Module using the collected sensor measurements. The Network Manager Module is the module that monitors the MST control network and modifies this network if it determines that the signal quality of the MST connections can be improved. The modules of the architecture are described in detail in the remaining sections following the sequence for the control flow diagram (Figure 5.1).

5.3 The Communication Manager

The Communication Manager has three tasks: send periodical beacons, handle the exchange of information with the network of robots and sequentially executing when called by the main sequential loop (Figure 5.1). As explained in the previous section the periodical sending of beacon signals, and the storing and retransmission of information are processes that occur in parallel to the main execution sequence of the modules in the architecture. The beacon signals contain the most recent estimated robot position and its uncertainty. In BERODE the robots determine the quality of the beacon signals by measuring the *RSSL* (Received Signal Strength Level) which is a value typically available in *RF* technologies. When the *RSSL* is not available the Cartesian distance between the robots is used as the signal quality measurement (Section 8.3, page 231).

The Communication Manager handles the information that is exchanged between the robots in the network. Information is exchanged by using messages. A message has the following properties: type, source and content. A message can have one of the two types: event or periodical. Messages of an event type are those whose content may modify the behaviour of the robot (e.g. an updated local network). Messages of a periodical type are those whose content does not modify the behaviour of the robot. The source for a message can be external or internal. External messages are the messages that a robot received from the robot network through its communication device. Internal messages are messages generated by another module in the architecture once it finishes its sequential execution. For instance the Interface Map Module generates a message whose contents are the most recent measured features. Table 5.1 presents the list of messages of an event type. Table 5.2 presents the list of messages of a periodical type. The tables present a description of the content of the message and show which module generates the message.

From Table 5.2 it can be seen that the robots send their extracted features to other robots in the network (Internal features). These features can be local or global. Local features are sent to the robots in the local network. Global features are sent to all the robots in the network. Local features are sent more frequently than global features to improve scalability.

The features that the robot receives from others are referred as External features. Like internal features these features can be either local or global. Section 5.10 describes the type of transmission used for each type of message.

Figure 5.3 shows the processes executed depending on the type and source properties of the message received. Messages of an event type are placed into an event queue.

The event messages with content “update role” and “t_{exp}” are messages generated by the Network Manager Module when the current behavioural role of the robot is not appropriate according to the level of safety. The event message with content “at goal” is generated by the Robot Motion Controller when the robot has reached its current goal. The purpose of these messages is to trigger the reselection of the behavioural role for the robot. For this reason these messages are not stored for transmission. The Behaviour Selection Module is called by the Communication Manager when the event queue is not empty (Figure 5.4). When the reselected role differs from the previous role the new role is transmitted using the “Role” message (Table 5.2).

The messages received from other robots in the network (source=“external”) are re-

transmitted as necessary according to the communication protocol from Section 5.10. The retransmission of messages occurs immediately to minimise communication delays. The information from these messages is stored by the Communication Manager. The information from these external messages is sent on a request basis to other modules. When the Communication Manager receives a message related to the recalculation of the local network or the global MST control network the Communication Manager sets a “hold position” flag (Figure 5.2). This keeps the robot stationary while the recalculation process is executed.

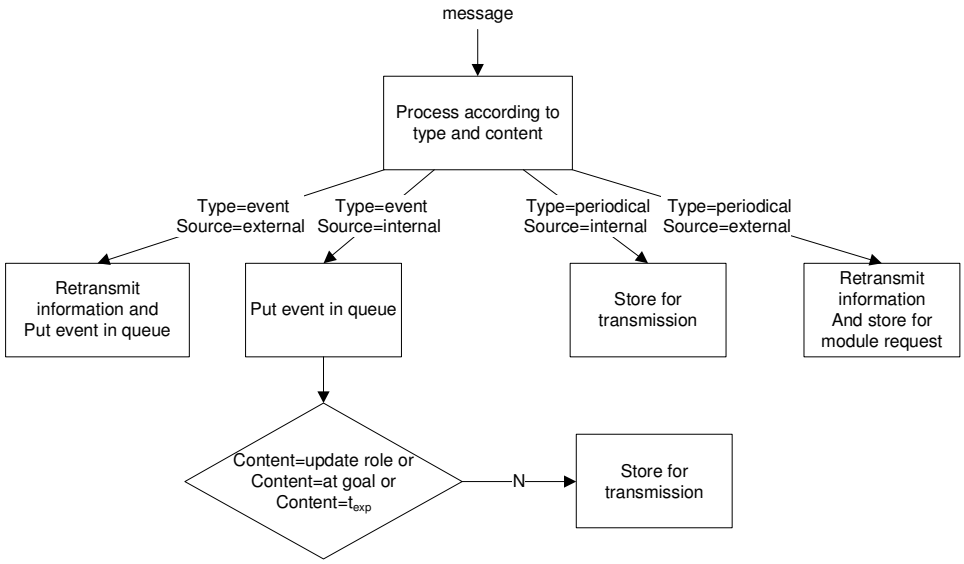


Figure 5.3: Process executed by the Communication Manager depending on the type and source properties of a message received. Messages of an event type are put in an event queue. Messages whose source is internal are stored for transmission. When the module is called for its sequential execution the event queue is processed and the internal messages are transmitted. External messages are retransmitted as necessary. The content of external periodical messages is stored and sent on a request basis to other modules.

The Communication Manager stores all the information related to the robot network. This information is:

1. The estimated positions of the direct connections of the robot.
2. The local network.
3. The behavioural roles of the robots inside the local network.
4. The goal location for the robots inside the local network.

Message Content	Source	Generated by	Description
At goal	Internal	Robot Motion Controller	The robot has reached the current desired position
Update role	Internal	Network Manager Module	The role is unsuitable for the current network conditions.
New Local network	Internal	Network Manager Module	Local network updated by the robot
New MST control network	Internal	Network Manager Module	Network updated by the robot
t_{exp}	Internal	Network Manager Module	An Explorer robot had an exploration failure and inhibits this role until the time t_{exp} has passed (Section 4.5, page 77)
Map Complete	Internal	Planning Module	The map is considered as completed
New local network	External	Robot Communication Device	Network updated by a robot within the local network
New MST control network	External	Robot Communication Device	Network updated by another robot
Map Complete	External	Robot Communication Device	Another robot considered the map as complete

Table 5.1: Messages of the event type. These messages are placed into a queue of events by the Communication Manager. The event message can have a local origin (internal) or can be an event message received from the robot network (external). Internal Event messages are generated by the modules when they finish their execution.

5. The local external features from the robot inside the local network.

6. The global external features from all the robots in the network.

As explained in Section 5.2 the initial call to the Communication Manager is done by an initialization process in which the initial MST control network is built, after which it is called at the start of the main control loop (Figure 5.1). Figure 5.4 shows the flow diagram for the Communication Manager when it is called for its execution. The Communication Manager starts its execution by setting an internal flag “new network” to false. Afterwards the manager starts processing the event messages stored in the event queue. Messages whose contents modify the MST control network are used to update the network. The network is updated by replacing the previously stored MST control network. This sets the internal flag “new network” to true. As explained in Section 4.4.1 (page 70) the process of recalculating a local network includes validation

Message Content	Source	Generated by	Description
Role	Internal	Behaviour Selection Module	The selected behavioural role
Robot Position	Internal	Map Interface Module	The estimated position of the robot
Local Internal Features	Internal	Map Interface Module	The features that are shared with the robots in the local network
Global Internal Features	Internal	Map Interface Module	The features that are shared with all the robots in the network
New Goal	Internal	Planning Module	The Planning Module generated a new path
Robot Position	External	Robot Communication Device	A robot has sent the beacon signal that contains its estimated position
New Goal	External	Robot Communication Device	A robot in the local network has set a new goal location
Role	External	Robot Communication Device	A robot in the local network has modified its role
Local External Feature	External	Robot Communication Device	Feature extracted by another robot in the local network
Global External Feature	External	Robot Communication Device	Feature extracted by another robot in the network

Table 5.2: Messages of the periodical type. The information from these messages is stored by the Communication Manager. The event message can have a local origin (internal) or can be a event message received from the robot network (external). Messages are generated by different modules.

mechanisms to ensure that the robots maintain the consistency of the MST control network. The Communication Manager stores the value for the variable t_{exp} which is the expiration time for the Pusher role (Section 4.5, page 77). Once this expiration time has passed the Network Manager Module (Section 5.9) generates an event message and the variable is set to the current time[7].

The Communication Manager calls the Behaviour Selection Module if there was at least one event in the queue of events, otherwise the Collision Module is called. If the map has been completed (message content=map complete) the exploration process finishes. After processing the queue of events the information stored for transmission is sent through the communication device to the network of robots.

[7] The robots have an internal clock which is started at the beginning of the group exploration process.

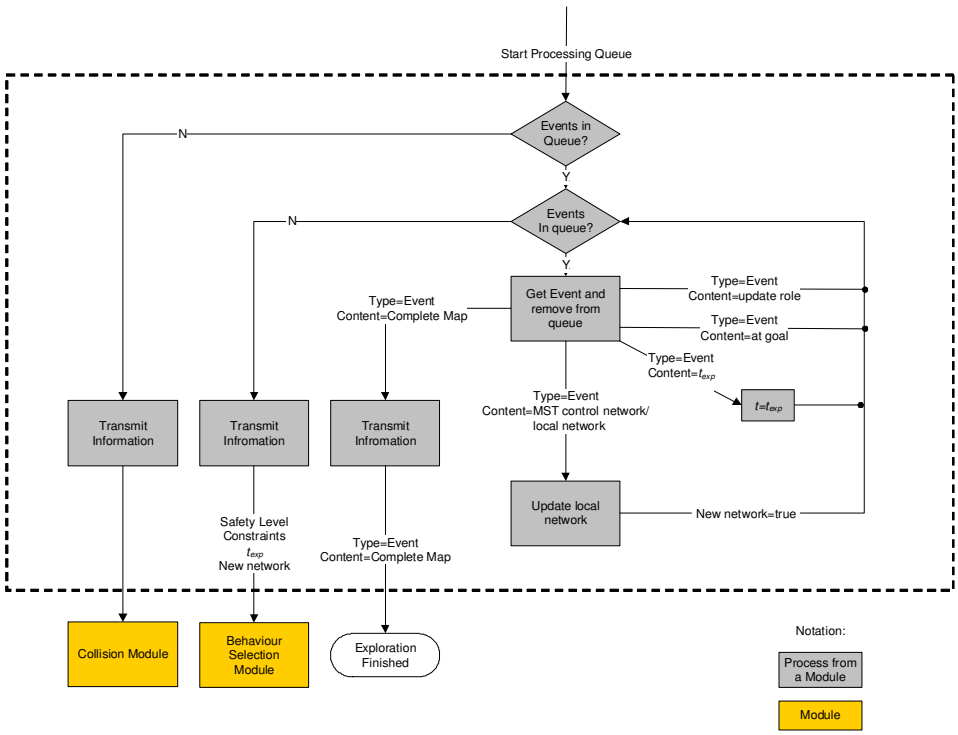


Figure 5.4: Control flow for the Communication Manager. The manager processes the events in the queue. The Collision Module is called if the queue of events was empty at the start of the execution; otherwise the Behaviour Selection Module is called. If the map has been completed the exploration process finishes. After processing the queue of events the information stored for transmission is sent through the communication device.

5.4 Behaviour Selection Module

The Behaviour Selection Module is called by the Communication Manager when an event has occurred. An event occurs once a robot has achieved its current task, detected a change in its safety level or a change in the local or the complete MST control network.

In the BERODE architecture a robot exhibits one of the following behavioural roles: Explorer, Maintainer, Pusher and Recoverer. The Explorer and Maintainer behavioural roles are the basic behaviours for the robot in the network. The Explorer robots are the robots with one connection in the control network while the Maintainer robots are the robots that have two or more connections in the control network. Because of this the Explorer robots are focused on exploration while the Maintainer robots are focused on keeping the robot network connected. The Pusher behavioural role helps the exploration task when it becomes problematic; for instance when an Explorer robot had an exploration failure. An exploration failure occurs when an Ex-

plorer robot determines that it is not possible to reach an unexplored area without losing communication (Section 4.5, page 77). The Recoverer robot helps the communications maintenance task when it becomes problematic. The Recoverer role tries to improve the level of safety for its unsafe control connections. To speed up the improvement on the safety level the set of communication constraints for a Recoverer robot is formed only by the unsafe control connections rather than the entire set of control connections (Section 4.3.2, page 65).

Figure 5.5 presents the algorithm executed by the Behaviour Selection Module to select the appropriate behaviour. The algorithm starts by obtaining the state of the robot $Z(t)$ at the current time. The state Z of the robot at time t is described by $Z(t)$={Safety level, Constraints, t_{exp}, New network, Safest frontier}. The Safest frontier variable is obtained from the Planning Module (Figure 5.2) by means of a query, while the other variables are provided as input by the Communication Manager. The possible values for the robot state variables and their meaning is:

1. Safety level: safe, precautionary, unsafe. The safety level forms part of the network status for a robot (Section 4.3, page 63) and describes the risk for a robot of going out of communication range for at least one of its communication constraints.

2. Constraints:1,..., λ. This variable represents the number of communication constraints for a robot and is part of the network status of the robot (Section 4.3.2, page 65).

3. t_{exp}: 0,...,$t + t_{pusher}$. This is the expiration time associated with the Pusher behaviour. t is the time at which an Explorer robot had an exploration failure. t_{pusher} is a user defined variable that determines the amount of time for which the transition to the Explorer role is inhibited.

4. New Network: Boolean state that is true if the MST control network has changed.

5. Safest frontier: safe, precautionary, unsafe. This variable returns the level of safety for the safest unexplored area according to the predicted communication quality for the robot and its direct connections (Section 5.6).

In the BERODE architecture when a new connection is detected the robots recalculate the local network to improve signal quality. If the local network is modified

the robots inside this network reselect their roles. When the local network is modified (New Network) the Pusher behavioural role is excluded from the selection process because the exploration failures that triggered the transition to the Pusher role are unlikely to be present due to the improved signal quality for the local network connections.

5.5 Collision Avoidance Module

The Collision Avoidance Module detects collisions with static and dynamic obstacles. The Planning Module is called after the Collision Avoidance Module. Usually the current plan remains the same, but if any collisions are detected the plan is adapted to avoid them using the most recent information about the positions of the obstacles and robots.

The Collision Avoidance Module estimates the distance and velocity for obstacles based on the information obtained from the robot sensors and the current robot velocity. Regardless of the sensors used the sensor bearing of the robot is partitioned into regions (Figure 5.6). The distance and velocity of the closest obstacle for the each region is estimated independently. The estimation process uses a temporary measurement window because of the noise in the sensor measurements. The measurement window stores the last m measurements for each region ($m = 5$ in our implementation).

The robot platform used in this work uses sonar sensors. Sonar sensors are suitable for obstacle avoidance purposes because of their wide beam, typically 25^o to 45^o. Each sonar sensor covers a region in the sensor bearing of the robot as observed in the Figure 5.6. For sensors such as laser and infrared with narrow beams measurements from neighbouring sensors that lie in the region can be grouped. The closest obstacle for each region is then the smallest measurement of the grouped measurements.

The distance $D_{obstacle}$ and velocity $V_{obstacle}$ of the closest obstacle for each region is calculated as follows

$$D_{obstacle} = \frac{\sum_{i=1}^{m} d_i}{m} \tag{5.1}$$

$$V_{obstacle} = \frac{\sum_{i=1}^{m-1} w_i(d_i, d_{i+1}) * v_{obstacle}(d_i, d_{i+1})}{\sum_{i=1}^{m-1} w_i(d_i, d_{i+1})} \tag{5.2}$$

where

$$w_i(d_i, d_{i+1}) = \begin{cases} K_{safe} * \left(D_{safe} - \left(\left(d_{i+1} + d_i\right)/2\right)\right) & D_{safe} \geq \left(\left(d_{i+1} + d_i\right)/2\right) \\ 0 & D_{safe} < \left(\left(d_{i+1} + d_i\right)/2\right) \end{cases} \tag{5.3}$$

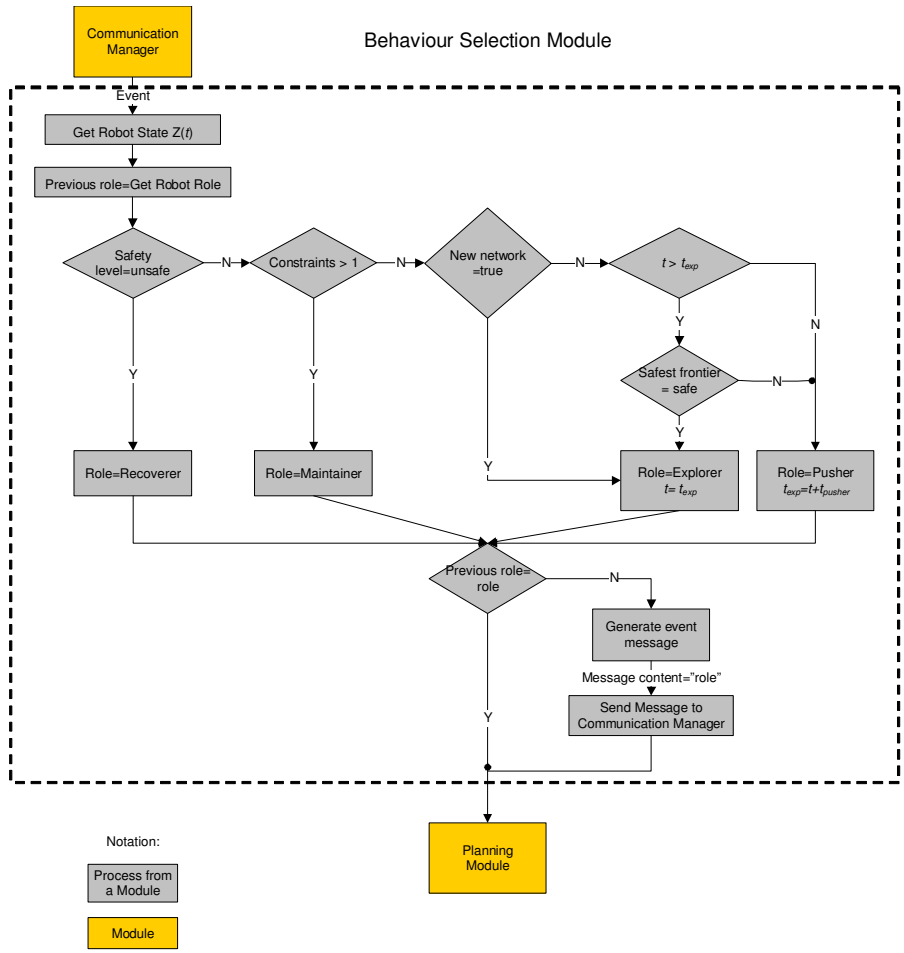


Figure 5.5: Algorithm to determine the behavioural role for a robot. The Behaviour Selection Module is called by the Communication Manager once an event has occurred. The behaviour is determined based on the robot state $Z(t)$. Once the behaviour has been selected the Behaviour Selection Module calls the Planning Module which generates an appropriate plan for the behavioural role.

and

$$v_{obstacle}(d_i, d_{i+1}) = \begin{cases} \frac{d_{i+1}-d_i}{t_s} & d_{i+1} < d_i \\ 0 & d_{i+1} \geq d_i \end{cases} \tag{5.4}$$

where d_i is the i^{th} measurement in the temporal window. The parameter $w_i(d_i, d_{i+1})$ is a weighting factor. This factor depends on the values of K_{safe} and d_{safe}. d_{safe} is a precautionary distance. When an obstacle is closer than this distance the velocity of the obstacle is estimated. The parameter K_{safe} is a scaling factor and is user determined. Large values of K_{safe} rely more recent measurements. The velocity of the obstacles $v_{obstacle}(d_i, d_{i+1})$ is assumed to be constant and is estimated between sampling intervals. t_s is the sampling time for the sensors. A collision is detected when

$D_{obstacle} < D_{collision}$ or $V_{obstacle} > V_{robot} + V_{threshold}$. The first type of collision typically is a collision with a static obstacle; whereas the second type of collision occurs for dynamic obstacles. For static obstacles with perfect sensors $V_{obstacle} = V_{robot}$. $V_{threshold}$ is a user defined threshold to avoid the detection of collisions with static obstacles as dynamic obstacles because of the noise in the sensors. Figure 5.7 presents an example for a static and a dynamic obstacle for one sensorial region. The robot travels along its x direction. It is observed that the distance for the dynamic obstacle diminishes faster than for the static obstacle. The apparent movement from the static obstacle (Eq. 5.4) is observable in Figure 5.7(a) where the distance diminishes from d_1 to d_2.

5.6 The Planning Module

This section describes the Planning Module used by the robots. The Planning Module obtains the estimated positions and the signal quality of the robot direct connections from the Communication Manager (Figure 5.2). The Planning Module uses this information to generate communication sensitive plans which keep the robot connected to the network. Depending on the behavioural role of the robot a different planner is selected. BERODE implements two planners: The predictive planner and the exploration planner. The non-Explorer behavioural roles use the predictive planner to generate reactive plans to keep the network connected. The Explorer role uses an exploration planner to generate a plan to move towards the most attractive unexplored area of the environment. The most attractive area is the safest area with respect to communication with the largest utility. The utility of an area is a function of the size of the area and the path length from it to the Explorer robot.

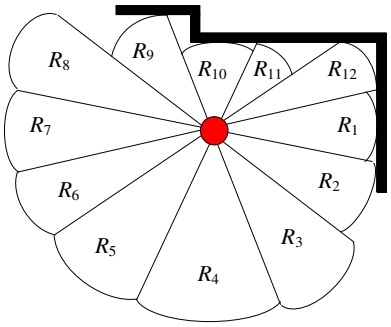


Figure 5.6: Regions for the Collision Avoidance Module for a robot with sonars with a wide beam of 30^o.

Figure 5.8 presents the processes carried out by the Planning Module. This module

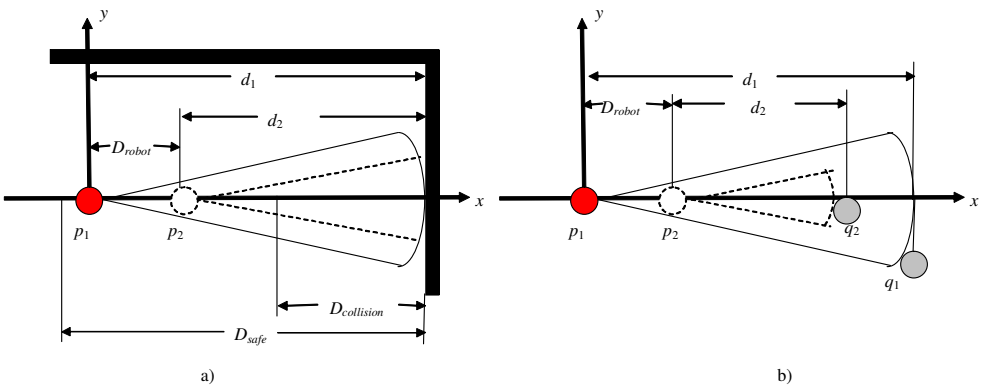


Figure 5.7: An example of a robot detecting a (a) static and a (b) dynamic obstacle. At time t_1 the robot is in position p_1 and the dynamic obstacle is in position q_1, at time t_2 the robot moves to position p_2 and the dynamic obstacle moves to position q_2.

can be called by the Collision Module or by the Behaviour Selection Module as shown in Figure 5.1. When the Planning Module is called by the Collision Module for a non-Explorer robot the validity of the plan is checked. A plan remains valid as long as the Cartesian distance between the last reported position for the constraining robots (communication constraints) and their positions at the last planning time is smaller than a user defined threshold (Section 4.3.2, page 65). Once it has been determined that the plan is not valid a new plan using the most recent information is generated. When the Planning Module is called by the Behaviour Selection Module the appropriate planner is selected and a plan is generated.

To generate the plan the Planning Module obtains the feature map from the Map Interface Module (Figure 5.2). The predictive and exploration planners project the feature map into a grid map. The predictive planner projects a subset of features close to the estimated robot position into a local grid map. The exploration planner also projects a subset of features to identify close unexplored[8] areas. If there are no close unexplored areas the exploration planner projects all the features into a global grid map. If there are no unexplored areas in this global map the exploration is considered as completed. The search for unexplored areas in the global grid map is computationally expensive, but only performed by Explorer robots who can't find any local unexplored areas to explore. This doesn't happen often and typically only towards the end of exploration when most has been mapped. Once an Explorer has made a plan to reach a distant unexplored area then the global search will be not be repeated until the

[8] An unexplored area is a portion of the environment in the projected grid map for which there is no evidence. The size of this portion is user defined.

robot reaches this unexplored area.

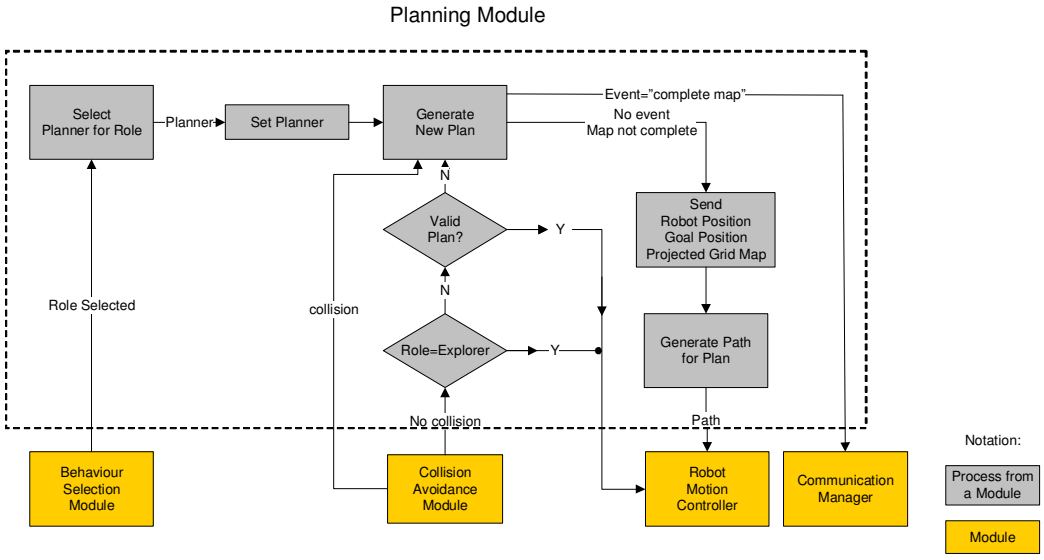


Figure 5.8: Control Flow for the Planning Module. The Planning Module can be called by the Collision Module or by the Behaviour Selection Module as shown in Figure 5.1. The Planning Module determines when the map has been completed and calls the Communication Manager. When the map is not considered as completed the Planning Module generates a plan according to the current behavioural role of the robot. The plan delivers a path that serves as input for the Robot Motion Controller.

When the map is considered to be complete the Planning Module generates a "complete map" event message and calls the Communication Manager. The Communication Manager will then process the "complete map" event message and the exploration process will stop for the robot. The message is transmitted by the Communication Manager to the robot network and upon reception of this message the other robots in the network will stop their exploration process.

When the Planning Module does not consider that the map has been completed this module generates and delivers a path that serves as input for the Robot Motion Controller. When the Planning Module has determined that it is not necessary to create a new plan the Robot Motion Controller keeps using the last path generated by the Planning Module.

When the Behaviour Selection Module is executed this module queries the Planning Module about the level of safety for the safest unexplored area according to the predicted communication quality for the robot and its direct connections. The Planning Module uses the same projection process used in the generation of the plan to

predict the communication quality. The feature map is projected into a local grid map. The size of the grid map is determined based on the boundaries formed by the robot and its direct connections (Figure 5.9). The communication quality is predicted using the projected feature map and the estimated positions of the robot and its direct connections.

The following sections describe the predictive planner, the exploration planner and the path generator. The planners generate as output a goal location and a projected grid map. These features are used by the path generator to create a path that the robot has to follow. The goal location is the location to which the robot has to move. The projected grid map is a projection into a grid of the feature map built by the Map Interface Module (Section 5.8).

5.6.1 Predictive Planner

The predictive planner is used by a non-Explorer robot to generate a reactive plan that keeps its communication constraints within communication range. This planner builds a plan based on the attraction/repulsion forces exerted by the robot communication constraints and the obstacles. The force of a communication constraint is modelled as a heterogeneous *virtual spring*. The springs are described as heterogeneous because they are asymmetric[9] and their free spring length[10] is a range of values rather than a single value. The magnitude of these forces is a function of the discomfort distance and the roles of the robots that form the connection. The discomfort distance for a pair of robots is the difference between their current signal quality and a desired signal quality (Section 4.3, page 63).

Obstacles generate repulsive potential fields. These repulsive potentials are a function of the distance. The planner generates a plan to move to the best OSQ (Overall Signal Quality) position. The best OSQ position is a nearby position where the energy generated by the forces and potentials is minimized. In this position the communication constraints have small values of discomfort distance. When a robot is in the safe level and there are no obstacles in the nearby area the energy is zero because no communication constraints apply in this safety level (Section 4.3.2, page 65). The robot remains stationary in this situation.

In previous research in control based on potential fields the robots move in the

[9] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

[10] The free spring length is the length for which the spring exerts a null force.

direction of the total potential energy vector (Powers and Balch, 2004). Once the robot has new information about the network status this vector is recalculated and the new direction of movement is determined. This approach is reactive and has been successfully implemented in *leader- follower* control approaches (Thibodeau et al., 2004). Wireless communication is unpredictable in indoor environments where temporary disturbances and local *hot spots* due to reflections are likely to be present. In this type of environment a reactive approach is likely to fail because of the local maxima. For such reasons, we proposed the predictive model based on short term plans. Based on the results of our simulations (Section 9.3, page 270) it is argued that this model is more suitable for real world scenarios because the network is kept fully connected for more time when short plans are used compared to long term plans.

The predictive model is based on the assumption that the robots are static. This assumption is reasonable because the robots test only close positions. In the general case, the difference between the predicted signal quality and the real signal quality value is small. Moreover if the robots are considered dynamic objects they require the exchange of information about their current planned path. The planner then has to predict the positions (including the planning robot) of all the robots for each tested position. The predicted positions for the robots are different because the path length for each tested position is different. The prediction of the positions depends then on the assumption that the robots move at the same speed and that their plans will remain the same. This increases the computational cost of the model and relies on another assumption that is less likely to hold in real world scenarios.

The predictive model can be either reactive or plan oriented. In the reactive model the positions tested are closer than in the predictive model. Thus, shorter plans are generated more frequently. Chapter 9 presents experiments to analyze the effect of length of plan in BERODE.

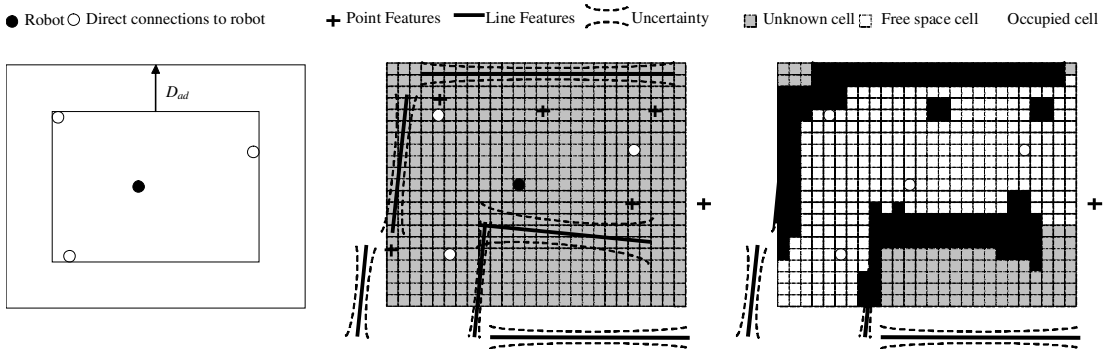
The predictive planner has two stages: projection and sampling. At the projection stage a subset of features from the feature map is projected into a local grid map. At the sampling stage the current robot location and several locations inside the local grid map are tested to obtain the best OSQ (goal position). Once the planner has calculated the goal location the path generator process (Figure 5.8) is called using as input the robot and goal positions, and the projected grid map to generate the path that the robot has to follow to reach the goal location. The projection and sampling stages are described in the following subsections.

5.6.1.1 Projection of Features into the Local Grid

The predictive plans are short term plans that test close positions to the robot current location. The Map Building Module maintains a feature map representation. The feature map is composed of point and line features. This map is used for localisation but is inconvenient for path planning purposes. In the sampling stage of the module random close positions are tested. The test calculates the predicted OSQ for the positions. This prediction is based on the projection of the features on a probabilistic grid map. Path planning algorithms for grid maps are simple and easy to implement. Moreover obstacle avoidance of static and dynamic obstacles can be easily achieved using potential fields. To the best of our knowledge all the path planning algorithms that handle dynamic obstacles use a local grid representation to avoid the collisions. As the environment size increases the difference between the computational cost of planning in grid maps and feature maps becomes larger. Planning in large grid maps is impractical because of the computational cost. This is not an issue for the Planning Module because only the features that are close to the robot location need to be projected to generate a short term plan. Line and point features are ordered incrementally according to their distance from the origin (initial robot position) in the Map Building Module (Section 6.3.9, page 156). The process to select the features is speeded up because of this sorting.

An example of the projection process is shown in Figure 5.9. The initial size of the grid map is determined based on the boundaries of the rectangular area formed by the estimated positions of the robot and its communication constraints (Figure 5.9(a)). Once the initial size of the map is determined the rectangular area is expanded a distance D_{add}. The value of D_{add} is a user defined variable. D_{add} has to be large enough to guarantee that the best OSQ position is found. Once the grid map is generated the point and line features that lie inside the grid map are selected (Figure 5.9(b)) and projected (Figure 5.9(c)) in the local map. The feature projection process incorporates the uncertainty and viewpoints of the feature. The uncertainty of the parameters is obtained from the *EKF* in the Map Building Module (Chapter 6). The viewpoints are relative positions to the feature at which the feature was observed. These viewpoints are grouped in clusters for storage and computational efficiency. The projection process of the features and their viewpoints is described in detail in Appendix B. Probabilistic maps usually classify cells as: occupied, unknown and free space cells. Before the features are projected all the cells in the projected grid map are unknown. After

the projection the features generate occupied and free space cells. The occupied space is formed by the cells where the features are projected, whereas the free space is the space generated by the viewpoints of the features.



a) Rectangular bounded area

b) Projected features on the local grid map

c) Probabilistic grid map

Figure 5.9: Projection stage in the Planning Module. a) Rectangular boundary area determined by the robot and its direct connections, b) line and point features projected in the local grid map, c) probabilistic grid map generated by the features.

5.6.1.2 Sampling Signal Quality in the Local Grid

The sampling process randomly tests positions over the projected grid area (Monte Carlo approach). The positions of the random samples are calculated as

$$P_r = \begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x + N(0, \eta * x_{length}/2) \\ y + N(0, \eta * y_{length}/2) \end{bmatrix} \tag{5.5}$$

where (x,y) is the estimated position of the robot, $N(\mu,\sigma)$ is a Gaussian distribution, (x_{length}, y_{length}) are the dimensions of the projected grid map and η is a scale factor that determines the closeness of the positions. The value of η is user defined.

The sampling algorithm tests the current estimated position of the robot and a number of valid positions. A valid position is a position whose projection in the grid map lies in a free space cell that has not been tested in previous trials of the current sampling process. The value of the valid position is the difference between the magnitudes of the total energy at the robot and the valid position. The best viewpoint (BVP) is the valid position with the minimum negative difference. If the BVP is positive the robot is located at the best position and it remains static. The energy E for a valid position

with coordinates (x,y) in general is defined as

$$E(x,y) = \left| \sum_{i=1}^{c} \overline{U_i(x,y)} + \overline{U_{obs}(x,y)} \right| \tag{5.6}$$

where c is the total number of communication constraints for the robot, $\overline{U_i(x,y)}$ is the attractive/repulsive energy vector for the i^{th} communication constraint and $\overline{U_{obs}(x,y)}$ is the repulsive potential vector from the closest obstacle.

Attractive/repulsive forces for communication constraints

All the previous implementations that use attractive/repulsive *virtual forces* to control groups of robots use homogeneous forces (Section 2.3, page 24). These forces are a function of a distance parameter. In most of these approaches the distance parameter is the Cartesian distance between pairs of robots. We propose the use of heterogeneous *virtual forces* that are a function of a distance parameter and the behavioural roles exhibited by the pairs of robots. It is argued that the use of heterogeneous forces aids the exploration process because these forces are asymmetric (Section 2.4, page 27). The movement in the exploratory directions can be accelerated by setting the forces in such a way that a Maintainer robot is more attracted to Explorer robots than to Maintainer robots and is repulsed by Pusher robots. The Pusher robots can be used to accelerate the movement by being strongly attracted to the Maintainer and Explorer robots. The Explorer robots move towards the unexplored areas dragging the robot network behind them.

The *virtual forces* are modelled as springs. The potential energy for these forces is a quadratic function of the distance. Quadratic functions are the most widely used in this kind of application (Baker et al., 1985; Rimon, 1990) because of two properties. First, a quadratic function provides a linear control law with constant gain. Second, for small displacements all the potential functions are quadratic. Thus the quadratic function is a good potential function because of its simple form and because other potentials reduce to it for small displacements. The proof of these properties can be found in (Volpe and Khosla, 1990).

The discomfort distance is used to compute the *virtual spring* force generated by the communication constraint. The discomfort distance is the difference between their current signal quality and a desired signal quality for the communication constraint (Section 3.3, page 38). The magnitude of the force exerted by the communication

constraint $R_i(x_i,y_i)$ for the sampled robot position $T(x_j,y_j)$ is defined as

$$|F_i(x,y)| = \begin{cases} k_{compresion} * \sigma_D(SQ_{predicted},\sigma_A,\sigma_R) & SQ_{predicted} > \sigma_R \\ 0 & \sigma_A \geq SQ_{predicted} \geq \sigma_R \\ k_{stretching} * \sigma_D(SQ_{predicted},\sigma_A,\sigma_R) & SQ_{predicted} \geq \sigma_A \end{cases} \quad (5.7)$$

The magnitude of the potential energy function for the spring force is defined as

$$|U_i(x,y)| = \begin{cases} k_{compresion} * \sigma_D(SQ_{predicted},\sigma_A,\sigma_R)^2 & SQ_{predicted} > \sigma_R \\ 0 & \sigma_A \geq SQ_{predicted} \geq \sigma_R \\ k_{stretching} * \sigma_D(SQ_{predicted},\sigma_A,\sigma_R)^2 & SQ_{predicted} \geq \sigma_A \end{cases} \quad (5.8)$$

The orientation of the potential energy is

$$\theta_{i,j} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) \quad (5.9)$$

And its vector is

$$\overline{U_{j,i}} = |U_i(x,y)|\cos\theta_{j,i}\hat{x} + |U_i(x,y)|\sin\theta_{j,i}\hat{y} \quad (5.10)$$

σ_A and σ_R are thresholds that depend on the behaviours exhibited by the pair of robots. For the proposed spring model the range of values $[\sigma_R,\sigma_A]$ is a range where the force generated by the spring is null. The *virtual spring* is modelled as a spring with a range of free spring lengths rather than a single free spring length. $k_{compresion}$ and $k_{stretching}$ are user defined variables. $QS_{predicted}$ is the predicted signal quality according to the estimated position of the communication constraint $R_i(x_i,y_i)$ and the sampled position $T(x_j,y_j)$. σ_D is the discomfort distance between $R_i(x_i,y_i)$ and $T(x_j,y_j)$ and is calculated with Eq. 4.3 (page 67). The prediction model depends on the type of communication used. Chapter 6 presents the implementation of the prediction model for the *LOS* and *RF* communication technologies. In the *LOS* model the discomfort distance is the distance between the Cartesian positions of $R_i(x_i,y_i)$ and $T(x_j,y_j)$. In the *RF* model the discomfort distance is a function of the *RSSL* value obtained from the *RF* device.

Table 5.3 presents the values of σ_A and σ_R according to the robot behavioural roles. The thresholds are user determined and have to maintain the following condition

$$\sigma_{\max} > \sigma_{rep\_pusher} > \sigma_{far} > \sigma_{close} > \sigma_{far\_explorer} > \sigma_{collision} \quad (5.11)$$

This condition reflects our argument about heterogeneous forces and their usefulness in accelerating the movement in the exploratory directions. This condition prioritizes the roles in the following decreasing order: Recoverer, Explorer, Maintainer and Pusher.

As a result of this prioritization the robot network disperses towards the unexplored areas when there are no Recoverer robots and contracts when there are Recoverer robots. At a close distance all the heterogeneous spring forces exert a repulsive force to avoid collisions.

As explained in Section 4.5 (page 77) in initial experiments it was found that robots with communication constraints to robots that transition from the Pusher to Explorer behavioural role generated large modifications in their reactive plans. These large modifications generated local minima for the OSQ that deteriorated the performance[11] of the robot network. We addressed this problem by redefining the forces that involve robots in the Pusher behavioural roles in terms of the expiration time. The attractive/repulsive thresholds for the forces are gradually modified from the Pusher to the Explorer thresholds. The reactive plans generate small modifications because of the gradual variation of the thresholds.

Robot Role	**Constraint Role**	**Attractive/Repulsive thresholds**
Maintainer	Maintainer	$\sigma_A = \sigma_{far}$ $\sigma_R = \sigma_{close}$
Maintainer	Explorer	$\sigma_A = \sigma_{far\_explorer}$ $\sigma_R = \sigma_{collision}$
Maintainer	Pusher	$\sigma_A = \sigma_{decay}(t, \sigma_{max}, \sigma_{far}, t_{pusher}, t_{exp})$ $\sigma_R = \sigma_{decay}(t, \sigma_{rep\_pusher}, \sigma_{close}, t_{pusher}, t_{exp})$
Pusher	Maintainer	$\sigma_A = \sigma_{increment}(t, \sigma_{far\_explorer}, \sigma_{far\_pusherr}, t_{pusher}, t_{exp})$ $\sigma_R = \sigma_{increment}(t, \sigma_{collision}, \sigma_{close}, t_{pusher}, t_{exp})$
Recoverer	Recoverer	$\sigma_A = \sigma_{far\_explorer}$ $\sigma_R = \sigma_{collision}$

Table 5.3: The values of the attractive σ_A and repulsive σ_R thresholds for the *virtual spring* force model are a function the behaviours of the pair of robots.

The functions $\sigma_{inc}(t, \sigma_{start}, \sigma_{end}, t_{pusher}, t_{exp})$ and $\sigma_{decay}(t, \sigma_{start}, \sigma_{end}, t_{pusher}, t_{exp})$ are an incremental and a decay function respectively that provide smooth transitions with respect to Pusher robots. The decay and increment functions implemented are linear interpolations. These functions were selected because they are the smoothest (their first derivative is zero in all the range of values). These functions are defined as

$$\sigma_{inc}(t, \sigma_{start}, \sigma_{end}, t_{pusher}, t_{exp}) = \begin{cases} \frac{(\sigma_{end} - \sigma_{start})(t - t_{exp} + t_{pusher})}{t_{pusher}} + d_{start} & t \leq t_{exp} \\ \sigma_{end} & t > t_{exp} \end{cases} \quad (5.12)$$

[11] The performance of the robot network is the time to build a complete map of the environment.

$$\sigma_{decay}(t,\sigma_{start},\sigma_{end},t_{pusher},t_{exp}) = \begin{cases} \frac{(\sigma_{end}-\sigma_{start})(t-t_{exp}+t_{pusher})}{t_{pusher}} + d_{start} & t \leq t_{exp} \\ \sigma_{end} & t > t_{exp} \end{cases} \tag{5.13}$$

where

$$t_{exp} = t_{pusher} + t_0 \tag{5.14}$$

where t is the current time, t_0 is the time at which the robot transitioned to the Pusher behavioural role and t_{pusher} is the time that the robot exhibits the Pusher behavioural role before trying to transition to the Explorer behavioural role (Section 4.5, page 77). t_{pusher} is a user defined variable. Figure 5.10 presents the attractive/repulsive forces for the pairs of robots as a function of the time. The effect of prioritization and the smooth transition for the Pusher robots can be observed in this figure.

Repulsive potentials from the Closest Obstacle

The repulsive potential is an energy function which surrounds an obstacle and prevents the robot from colliding with the obstacle. According to Volpe and Khosla (1990) a repulsive potential function that is useful for modelling obstacles should have as attributes:

1. The potential contours near the obstacle should follow the obstacle contour so that large portions of the workspace are not effectively eliminated.

2. The potential of an obstacle should have a limited range of influence.

3. The potential and the gradient of the potential must be continuous.

Natural potential functions in physics (e.g. electrostatic, gravitational, etc.) exhibit an inverse dependence on distance. The repulsive potential function must have a K^{-1} dependence for short distance repulsion, but drop faster than K^{-1} for large distances. A function with these attributes is the Yukawa potential (Cohen et al., 1977):

$$Y(K) = A\frac{e^{-\alpha K}}{K} \tag{5.15}$$

The parameter α determines how rapidly the potential rises near the obstacle and falls away from it. The parameter A is an overall scale factor for the potential. Volpe proved that Yukawa's potential follows much better the shape of polygonal obstacles compared to other functions (e.g. square and Gaussian functions) tending to eliminate only the necessary area to ensure safety (Volpe and Khosla, 1990).

This thesis implements the Yukawa potential to calculate the repulsive potentials from the obstacles. The distance to the closest obstacle is calculated using a bounded

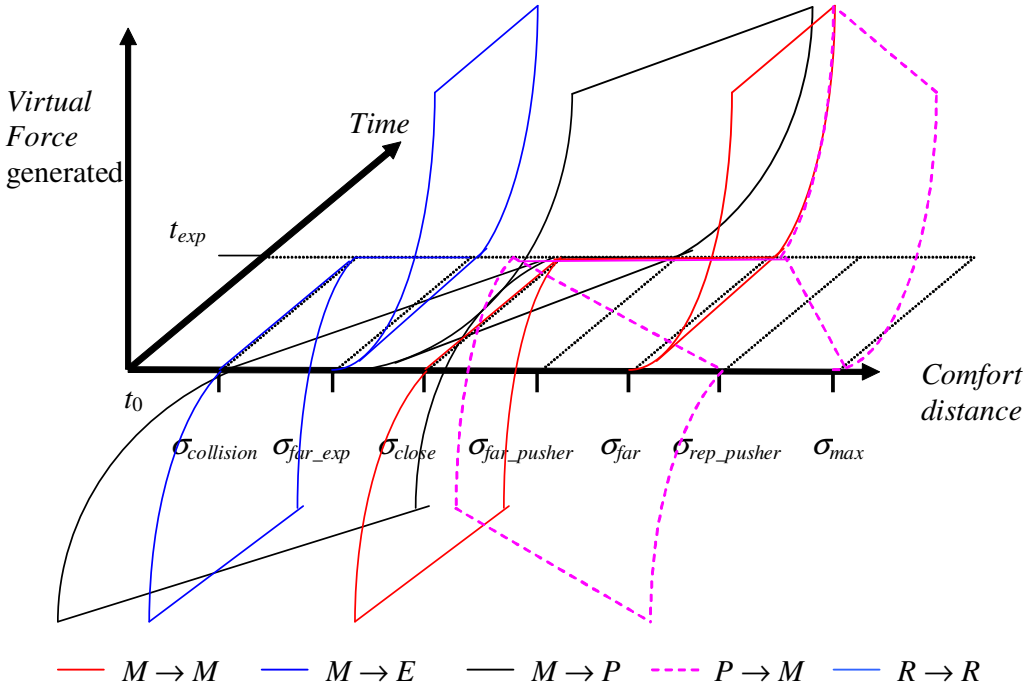


Figure 5.10: Attractive/Repulsive forces for pairs of behavioural roles as a function of the time. The forces that involve the Pusher role are gradually modified to avoid the generation of local minima for a robot with control connections to Pusher robots.

distance recursive algorithm in the projected grid map. The algorithm starts by marking as visited the locations contained in a list. The initial list contains the starting position. Neighbouring[12] unmarked cells to the cells in the list are checked to determine if they are occupied. Free space cells are marked as visited and used in the next iteration to test their unmarked neighbours. The algorithm stops after the bounded distance is reached or once an obstacle cell has been found. The bounding distance is implemented for computational efficiency. If the algorithm reaches the bounding distance the potential function of the obstacle defaults to zero.

5.6.2 Exploration Planner

The Exploration Planner is used by an Explorer robot to generate a plan to move towards the most attractive area. This area is the unexplored area with the largest utility in the safest hierarchic level. The utility of an area is a function of the size of the area and the path length from it to the Explorer robot. The hierarchy of an unexplored area depends on the predicted communication quality at the unexplored area location.

[12] The neighbourhood of a cell is an eight-cell neighbourhood.

The Planning Module has two stages: projection and selection. At the projection stage only the features within the range of the local grid map are projected into it. The features are projected as in Section 5.6.1.1 using the positions of the robot direct connections and the positions of the robots in the local network to generate the rectangular bounded area (Figure 5.9). When there are no unexplored areas inside the projected space the rest of the space is projected to search for unexplored areas in the global grid map generated by the complete set of lines and point features. This search is expensive but it is only performed when there are no unexplored areas inside the local grid map and a plan to move towards this distant unexplored area does not yet exist. When there are no unexplored areas in this map the planner generates a "complete map" message which is sent to the Communication Manager (Figure 5.2) and the exploration process stops (Figure 5.1). This "complete map" message is sent to all the robots in the network through the communication device. Upon reception of this message the robots in the network will stop the exploration. The exploration process is then stopped when the first robot considers having a complete map. In the selection stage of the algorithm the unexplored areas are evaluated as a function of their size, path length and predicted communication safety. This stage is described in Section 5.6.2.1. Once the planner has calculated the goal location the path generator process (Figure 5.8) is called using as input the robot and goal positions, and the projected grid map to generate the path that the robot has to follow to reach the goal location. The projection and sampling stages are described in the following subsections.

5.6.2.1 Hierarchical Selection of Unexplored Areas

The evaluation of an unexplored area as a function of size and path length is referred as utility and was introduced by Simmons et al. (2000). He defined the concept of information gain for *frontiers*. A *frontier* is a portion of free space that is adjacent to unknown space in the probabilistic grid representation of the environment. The information gain for a *frontier* is the nearby unexplored area. This area is calculated by counting the number of unexplored cells that are within the circumference of the sensor range of the robot. The information gain expected by a robot considering moving to a particular *frontier* is lessened if there are any robots near the location. The utility of a *frontier* is the information gain value minus the cost of travelling to the *frontier* from the robot position assuming ideal movement. Large frontiers closer to the robots will more likely be attractive for the robots. In Simmons' approach robots submit tenders for their evaluated *frontier* to a central agent that assigns a *frontier* for each robot. The

central agent tries to maximize the total expected utility for the robots. Figure 5.11 presents an example of the information gain for several *frontiers*.

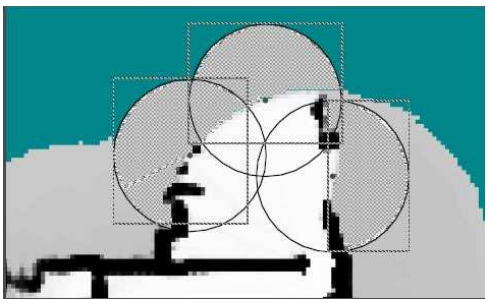


Figure 5.11: An example of information gain for several *frontiers* from (Simmons, 2000). Circles indicate sensor range. Cross-hatched areas are information gain regions.

In Simmons' research the robots have a global communication system while in this work the communication is local. We propose a hierarchic version of Simmons' approach that accounts for the communication quality at the *frontiers* positions. The communication quality of a frontier is the safety level for the position. The safety level is determined from the predicted signal quality (PSQ) for a group of robots $SQ = \{SQ_1, ..., SQ_i\}$ as follows

$$PSQ = \begin{cases} Safe & if \quad \exists j \left(SQ_i > \sigma_{safe}\right) \\ Precautionary & if \quad \exists j \left(SQ_i > \sigma_{prec}\right) \wedge \forall j \ \left(SQ_i \leq \sigma_{safe}\right) \\ Unsafe & if \quad \forall j \left(SQ_i < \sigma_{prec}\right) \end{cases} \tag{5.16}$$

where SQ_i is the predicted signal quality for the i^{th} robot. The group of robots depends on the hierarchical level of the *frontier*. The parameters σ_{safe} and σ_{prec} are the communication thresholds that represent the desired level of signal quality for the *frontier*. The signal quality prediction depends on the implementation. In *RF* technologies the communication signal is partially blocked by obstacles while for *LOS* technologies the signal is totally blocked by obstacles. Chapter 6 presents the implementation of the prediction models for these technologies. The prediction models use the grid map. The *RF* technology model assumes the free space[13] decay model and an average loss value for obstacles. The loss value is the amount of power a signal loses (in decibels) by passing through that obstacle and depends on the material and density of the obstacle. The *LOS* technology model assumes infinite line of sight.

[13] In the free space model the strength of a signal decays with the square of the distance between the transmitter and the receiver.

The hierarchy levels are designed to try to minimize the number of exploration failures. An exploration failure occurs when a robot generates a plan to a *frontier* which aborts because of possible loss of communication. These failures are likely to be present because the evaluation of the *frontiers* assumes that the robots will remain static. Even when all the unexplored areas are in the unsafe level, there will still be an Explorer planning to reach the best of these. The hierarchy level for a *frontier* is determined from Table 5.4.

Only the robots that are in the Maintainer and Pusher behavioral roles are considered in the determination of the hierarchy level because their main task is to keep the network connected. These robots tend to remain in nearby areas while trying to improve the OSQ. The Explorer robots are focused on the task of exploration and the Recoverer robots are focused in the fast recovery of communication constraints with low levels of signal quality.

Hierarchy	**Predicted *safety level***	**Group of robots**
h_0	safe level	The Robot
h_1	safe level	The Robot direct connections that are in the Maintainer or Pusher behavioural roles
h_2	safe level	The robots in the local network that are in the Maintainer or Pusher behavioural roles
h_3	precautionary/unsafe level	The robots in the local network that are in the Maintainer or Pusher behavioural roles
h_4	Any level	None

Table 5.4: Hierarchy level determination for a frontier. The safest hierarchy level is h_0 and the less safe is h_4. The robots select the frontier in the safest level with the largest utility. The utility is a function of the size of the area and the path length from it to the Explorer robot.

Figure 5.12 presents an example of the hierarchies for several *frontiers* for the *LOS* communication model. Frontiers f_1 and f_4 are in the safe level of robot $R_{4,E}$ and their hierarchy is h_0, frontiers f_3 and f_7 are in the safe level of robot $R_{0,P}$ and their hierarchy is h_1, frontiers f_6 and f_5 are in the safe level of robot $R_{5,M}$ and their hierarchy is h_2, although the frontier f_2 is in the safe level of robot $R_{2,E}$ its hierarchy is h_3 because of its behavioural role.

The planning algorithm selects the *frontier* with the largest utility in the smallest hierarchy level. The *frontiers* in the first three hierarchic levels are determined in the initial projected space. This initial space is the projected local grid map for the robot

direct connections and the robots in the local network[14]. When there are no unexplored areas inside the projected space the rest of the space is projected to search for unexplored areas in the grid map generated by the complete set of lines and point features. These additional frontiers have an h_3 hierarchy. Once there are no *frontiers* left to explore the map is considered as complete. The planner generates a "complete map" event message which is sent to Communication Manager (Figure 5.2). The Communication Manager sends this message to the robot network and the exploration process is finished.

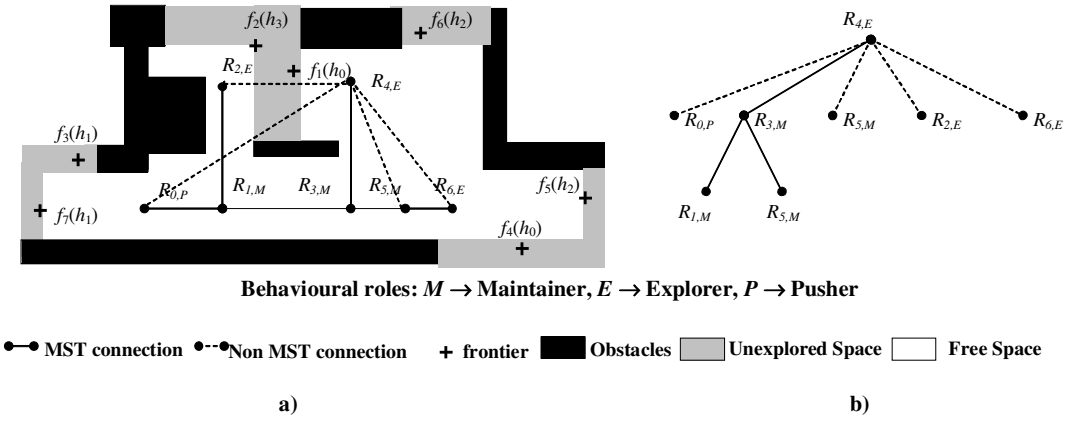


Figure 5.12: An example of the hierarchic *frontier* for the Explorer robot $R_{4,E}$ and its communication tree. a) the probabilistic map for the robot and the hierarchy of the evaluated *frontiers*. b) Tree of the direct connections and the local network for $R_{4,E}$.

When the Behaviour Selection Module is executed this module queries the Planning Module about the level of safety for the safest unexplored area. As explained in Section 5.5 the Planning Module predicts the communication quality for the robot and its direct connections. The prediction process is the same as the process used to identify close unexplored areas. These areas are the areas in the first two hierarchic safety levels and to predict their safety it is only necessary to make a local grid map projection.

5.6.3 Path Generator

The path generator is called by the currently selected planner (Figure 5.8). The path generator receives as input the position of the robot and the goal locations, and the projected grid map built by the planner.

[14] A direct connection for a robot is not necessarily part of the local network.

The path generator produces a sequence of points from the robot position to the BVP (Best View Point). The path generator uses a distance transform which is a grid-based wave propagation technique (*NF*1 function). *NF*1 constructs a navigation function which is an approximate potential field with a globally unique minimum at the goal (Latombe, 1991). Once this navigation function is calculated, the robot can reach the goal by descending along the gradient of this function. The idea underlying the *NF*1 is rather simple: Divide the environment into equally sized grid cells, mark all cells that lie within one robot radius of an obstacle ($C-$Space generation), then construct a monotonically increasing potential starting at the cells that are in the goal region. The robots are treated as static objects. Repulsive values are added to the cells that lie within a certain distance of the obstacles and robots in the propagation stage of the algorithm. These repulsive values are a function of the distance from the cell to the closest obstacle and robots. The Yukawa potential function is used to generate the repulsive values. The addition of this function generates paths where robots try to stay apart.

Because the robots are treated as static objects they are liable to collide into each other. The Collision Avoidance Module ensures the robot safety. This module is based on a *dynamic window* approach to detect collisions with dynamic and static obstacles. When the module detects a collision (Figure 5.8) the path for the current goal is recalculated using the same *NF*1 function with the most recent positions of the obstacles and the robots.

5.7 The Robot Motion Controller

The Robot Motion Controller generates the motion commands for the current path. The controller is called by the Planning Module (Figure 5.1). The Planning Module sometimes generates a new path when it finishes its execution (Figure 5.8). In most occasions the plan remains the same. If the Planning Module does not generate a new path the Robot Motion Controller maintains the last generated path as the path to be followed. The path is formed by a sequence of control points that the robot sequentially follows until it reaches the goal position. After the robot reaches a control point it takes some measurements that are used to update the feature map. In our implementation the distance between the control points is 10cm. In our experiments with a real robot we found that for this distance the trade-off between feature extraction and computational cost was the best for our sonar sensor measurements.

When the Robot Motion Controller is called and the hold position flag is set (Figure 5.2) the Robot Motion Controller keeps the robot in its current position until the hold position flag is removed. The Robot Motion Controller stores the sequence of the most recent movements executed by the robot. In our implementation the robot stores the last 3 movements (30cm). These movements are used by the robots in the Maintainer, Pusher and Explorer behavioural roles to backtrack their movements before transitioning to the Recoverer role (Figures 5.15 and 5.18).

5.8 The Map Interface Module

In BERODE each robot builds it own feature based map. The positions of the robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID number. The robots periodically distribute the features that they observe with the rest of the team. A robot incorporates the received features from others to its map using the same process as for locally observed features. This is possible because the robots use the same global Cartesian frame of reference.

The Map Interface Module is called by the Robot Motion Controller once the robot has executed a movement to follow the current plan (Figure 5.1). After the robot has executed a robot movement the Map Interface Module is called. The Map Interface Module obtains sensor measurements after the robot has executed a movement and extracts features from these sensor measurements. The extracted features are used to update the estimates of the position and uncertainty of the robot and previously mapped features.

The position of the robot and the mapped features are estimated using an Extended Kalman Filter (*EKF*). The *EKF* has two stages: prediction and update. In the prediction stage, odometry information is used to predict the locations of the robot and the mapped features. In the update stage, measurements of features are used to update the estimates of the locations of the robot and the mapped features. Feature measurements are obtained from a feature extraction and data association process. The features are extracted from raw sensor data from sonar and infrared sensors.

The processes of estimation and updating of the *EKF* are carried out by a Map Building Module which is contained in the Map Interface Module. This module is described in detail in Chapter 6. Figure 5.13 shows the algorithm executed by the Map Interface Module. The details about the feature extraction process and the prediction

and update stages of the *EKF* are presented in Chapter 6.

In BERODE each robot builds it own feature based map. Line and point features are extracted from the raw sensor data. A geometric representation of the features is obtained. The features are represented by a measurement vector and an uncertainty matrix. The robots periodically distribute their observed features to the rest of the team. The periodical distribution of features is done at the local and global level. These features are referred to as internal features and can be local or global. Internal local features are sent to the robots in the local network (local level). Internal global features are sent to all the robots in the network (global level). Internal local features are sent more frequently than internal global features to improve scalability with respect to numbers of robots. The Map Interface Module uses a local and a global feature store for storing and periodically distributing the internal local and global features.

The features that the robot receives from others are referred to as external features. Like internal features these features can be either local or global. Local and global features integrate the same observations. The global external features are integrated to the robots' maps using the same process as for locally extracted features. This is possible because the robots use the same global Cartesian frame of reference. The local external features are useful navigational information for the local robots (robots inside the same local network). Local external features become part of the map via the global feature updating process. These features are not used to update the map because that would cause a double update.

After the module has updated its local map (EKF update) using the new features extracted locally the module obtains the external global features received since the previous execution of the Map Interface Module. The external global features are stored by the Communication Manager and are requested by the Map Interface Module at the beginning of its execution (Figure 5.2). These features are integrated to the map by associating them with the mapped features and updating the *EKF*.

As seen in Figure 5.13 once the robot has extracted new features from sensor measurements the Map Interface Module creates a copy of these features into the new local and new global variables. After the Map Interface Module finishes updating the *EKF* with the external global features the module uses a data association process to obtain the pairings between the new local features and the features that were stored in previous iterations in the local feature store. The data association process is the same as that used to obtain the pairings to update the *EKF* filter (Section 6.3.8, page 155). Following Smith et al.'s (1988) formulation once a pairing between a new local feature $\hat{f}_j$

with a measurement uncertainty matrix C_j and a previously stored local feature $\hat{f}_i$ with an uncertainty C_i has been found the stored feature and its uncertainty are recursively updated as follows

$$\hat{f}_i = \hat{f}_i + \left[\frac{C_i}{C_i + C_j}\right](\hat{f}_j - \hat{f}_i) \tag{5.17}$$

$$C_i = \left(1 - \frac{C_i}{C_i - C_j}\right) C_i \tag{5.18}$$

The new local features that did not have a pairing with the stored features are added as new features to the local feature store. The features in the local feature store are transmitted to the local robots once a certain number of iterations of the Map Interface Module have passed. A message that contains the parameters of these local features is generated and sent to the Communication Manager. The Communication Manager will send this message to the robots in the local network when it is called by main control loop (Figure 5.1). The Map Interface Module cleans the local feature store after generating the message. Afterwards the same processes are carried out for the global feature store. The number of iterations for which the features remain in the local and global feature stores is an important parameter to determine the scalability of BERODE with respect to communication. The more frequent sharing of the contents of these stores can increase the demand on communication bandwidth to a point where it can become a bottleneck for the system. Chapter 9 presents simulations to analyse this aspect.

Before finishing its execution the Map Interface Module checks if it is time to send the features stored in the local and global feature stores (Figure 5.13).

Figure 5.14 presents the processes executed by the Map Interface Module when the Planning Module requests the Feature Map (Figure 5.2). The Map Interface Module requests the external local features from the Communication Manager. The internal features are obtained from the feature map build by the robot. The internal and external features are sent to the Planning Module. The Planning Module uses the feature map to generate communication sensitive plans.

5.9 The Network Manager Module

The Network Manager Module is the module that monitors the direct connections of the robot and tries to update the local network when a new connection is detected. The Network Manager Module starts its execution by obtaining the signal quality for

the robot direct connections and the behavioural role of the robot from the Communication Manager (Figure 5.2). The module monitors the connections to ensure that the appropriate behaviour is executed according to the current level of safety. If the current behaviour is not appropriate the module generates an event to reselect the appropriate behaviour.

The module detects when a new connection has been formed. A new connection is detected when the robot has received a beacon signal from a new robot during the last t_{stable} seconds. This minimizes the detection of unstable connections (t_{stable}=2s in our implementation). When a robot detects a new connection it tries to recalculate the local network.

The details of the recalculation process for a local network are presented in Section 4.4.2 (page 74). A robot recalculates the local network when the following conditions are accomplished:

- The ID number is smaller than the ID number of the other robot forming the connection. This ensures that the recalculation process is executed only by one robot.

- The local networks from the pair of robots that form the new connection are consistent. This ensures that the recalculated local network will be consistent with the MST control network.

- There are no Recoverer robots in the pair of local networks. This ensures that the recalculation process can be executed without the risk of having a disconnected network. When a Recoverer robot detects a new connection it does not try to recalculate the local network.

If the manager detects more than one new connection it will try to rebuild the local network for the connection with the best signal quality. Nevertheless, this happens very rarely in practice. While the local network is being calculated the robots inside the local network keep their positions stationary by switching off motor power to the wheels of the robot hardware platform (Figure 5.2).

The Network Manager Module is called by the Map Interface Module. When the Network Manager Module finishes its execution it calls the Communication Manager (Figure 5.1). The implementation of this module is different for each role.

The implemented module for non-Explorer robots generates an “update role” message when the current behavioural role is no longer appropriate. For the Pusher and

Maintainer roles this occurs when the manager detects that the level of safety is unsafe. For the Recoverer role this occurs when the manager detects that the level of safety is not unsafe.

The implemented Network Manager Module for the Explorer role monitors and modifies the control connection of the robot if it determines that there is a direct connection with a better signal quality. The signal quality for the control connection is used to determine the level of safety for the robot (Eq. 4.2, page 65). If the robot is not in the safe level, the robot waits for improvement on the signal quality and if there is no improvement the robot gives up its current exploration. The module then generates a "t_{exp}" message that will trigger the behaviour selection process (Figure 5.18). The following sections describe the implementation of the Network Manager Module for each behavioural role.

5.9.1 The Network Manager Module for the Maintainer and Pusher Behavioural Roles

The implementation of the Network Manager Module is the same for the Maintainer and Pusher behavioural roles. The module verifies that the level of safety for the robot is either safe or precautionary. Figure 5.15 presents the flow diagram for the module. The module is called by the Map Interface Module. The first process is the request of the signal quality for the control connections from the Communication Manager (Figure 5.2). The signal quality is obtained by the Communication Manager from the latest beacon signals (Section 3.7, page 48). Once the robot has the signal quality for the control connections it can determine its safety level (Eq 4.1, page 64). Afterwards the set of communication constraints is determined (Eq 4.2, page 65). If the robot is in the unsafe level and there are any movements stored by the Robot Motion Controller (Section 5.7) the robot backtracks these movements. The purpose of the backtracking the movements is to return to a previous location where the safety level was not unsafe. After the robot has backtracked its movements it obtains the updated signal quality for the control connections. If the robot remains in the unsafe level then an event message with content="update role" is sent to the Communication Manager. This message is added to the queue of events handled by the Communication Manager. This ensures that the Role Selection Module is called in the following iteration (Figure 5.1). The level of safety may increase because beacon updates may be received between the generation of the "update role" event and the execution of the Role Selection Module

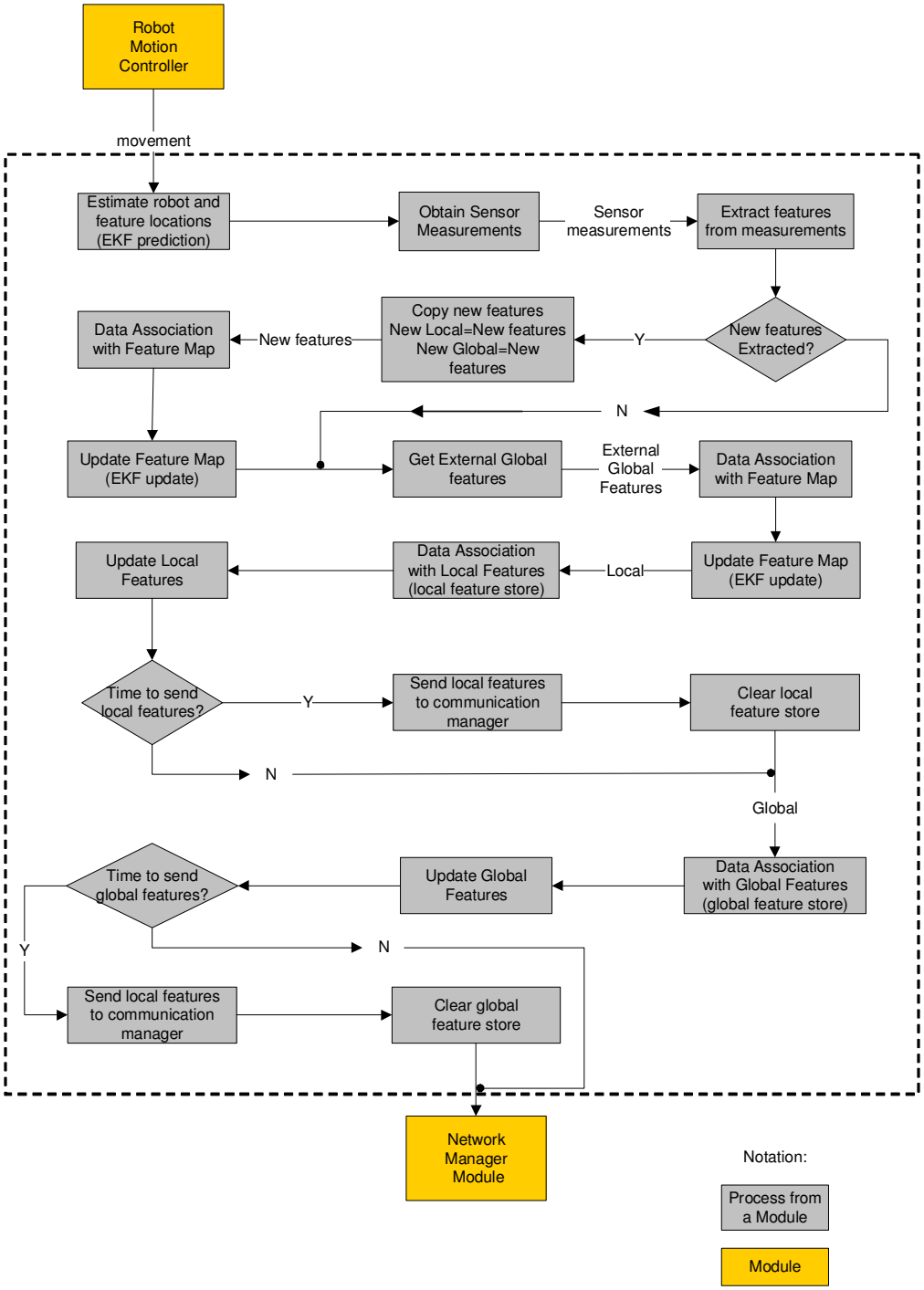


Figure 5.13: Flow diagram for the Map Interface Module. The module receives as input the movement of the robot. The module estimates the location of the robot and the features. Afterwards the module takes sensor measurements and extracts new features from these measurements. The feature map is updated using the new features. Copies of the new features are created and associated with the features previously stored in the local and global feature stores. The features from the stores are periodically transmitted to the robot network through the Communication Manager.

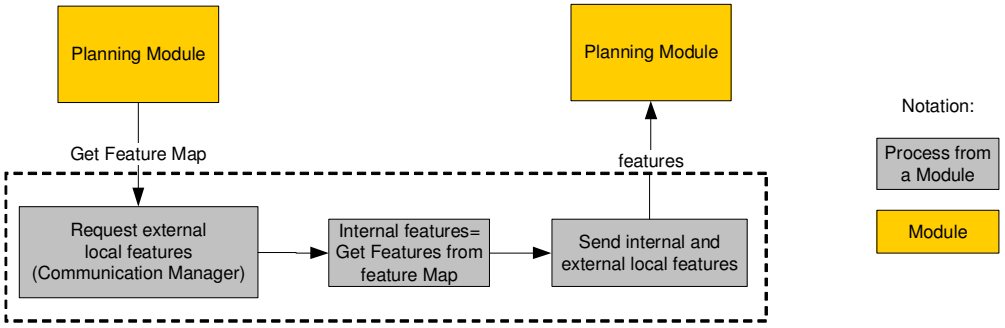


Figure 5.14: Processes executed by the Map Interface Module when the Planning Module requests the Feature Map. The Map Interface Module requests the external local features from the Communication Manager. The internal features are obtained from the feature map build by the robot. The internal and external features are sent to the Planning Module.

(Section 5.4) however this rarely occurs in practice. If the level of safety remains as unsafe the role that will be selected is the Recoverer role. This role will focus on generating plans to improve the control connections that are unsafe.

When the safety level is not unsafe the manager checks if there are new connections. If this is the case the manager tries to recalculate the local network. The previous section described the conditions that are necessary to recalculate the local network. When a new local network is created the manager creates an event message whose content is the new local network. This message is sent to the Communication Manager. The Communication Manager adds the message to its queue of events and sends the message to the robots in the local network. The queue is processed and the appropriate behaviour is selected. Upon reception of the message the robots in the local network will reselect their roles according to the updated local network.

5.9.2 The Network Manager Module for the Recoverer Behavioural Role

The Network Manager Module for the Recoverer behavioural role verifies that the robot is in the unsafe level. Figure 5.16 presents the flow diagram for this module. This module is called by the Map Interface Module. The first process is the request of the signal quality for the control connections from the Communication Manager (Figure 5.2). The signal quality is obtained by the Communication Manager from the latest beacon signals. Once the robot has the signal quality for the control connections it can determine its safety level (Eq 4.1, page 64). Afterwards the set of communication

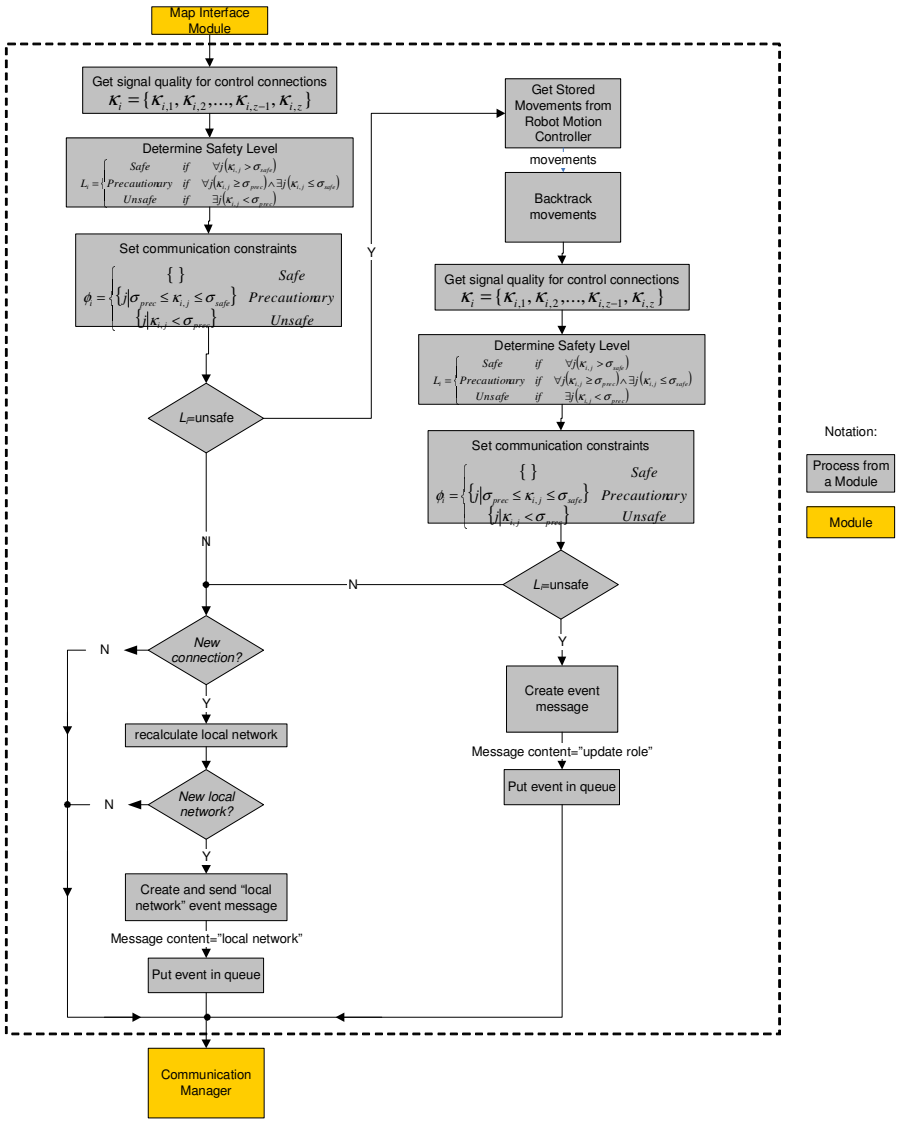


Figure 5.15: Flow Diagram of the Network Manager Module for the Maintainer and Pusher behavioural roles. The Network Manager determines the safety level L_i. When the safety level L_i is unsafe the robot backtracks its recent movements to try to improve the safety level. When a new connection is detected the manager tries to recalculate the local network. The event messages are sent to the Communication Manager. The Communication Manager adds the messages to its queue of events. The queue is processed and the appropriate behaviour is selected.

constraints is determined (Eq 4.2, page 65). If the robot is in the unsafe level the Network Manager finishes its execution. If the robot is not in the unsafe level then an event message with content="update role" is sent to the Communication Manager. This message is added to the queue of events handled by the Communication Manager. This ensures that the Role Selection Module is called in the following iteration (Figure 5.1). The Role Selection Module will then select the appropriate behaviour.

5.9.3 Network Manager Module for the Explorer Behavioural Role

The implementation of the Network Manager Module for the Explorer behavioural role monitors the local network and modifies this network if it determines that it can be improved. This module guarantees that the robot remains connected to the network because robots in the Explorer behavioural role are not subject to *virtual forces*. As explained in Section 4.5 (page 77) conflicts may arise when the Explorer robots pull the robot network in opposing exploratory directions halting the exploratory behaviour of the robot network. In these situations the Network Manager Module is used to resolve the conflict in a decentralized fashion. The module keeps the robot stationary waiting for improvement of the safety level for a small time. If the safety level does not improve after a certain time (user defined) the robot gives up the exploration task, backtracks its previous movements and reselects its behavioural role.

The Network Manager verifies that the Explorer robot is maintained in the safe level as it moves towards its current goal (Section 4.3, page 63). The manager modifies the robot control connection to improve the safety level. Based on the safety level the following actions are executed:

1. Safe level: The robot keeps moving toward its current goal.

2. Precautionary level: The robot maintains the current position for a certain time waiting for improvement in the safety level to the safe level after which the robot gives up the exploration of its current goal and reselects its behavioural role.

3. Unsafe level: The robot gives up the exploration, backtracks its previous movements and reselects its behavioural role.

Additionally when the manager detects a new connection, the manager tries to recalculate the local network. Section 5.9 described the conditions that trigger the recalculation of the local network.

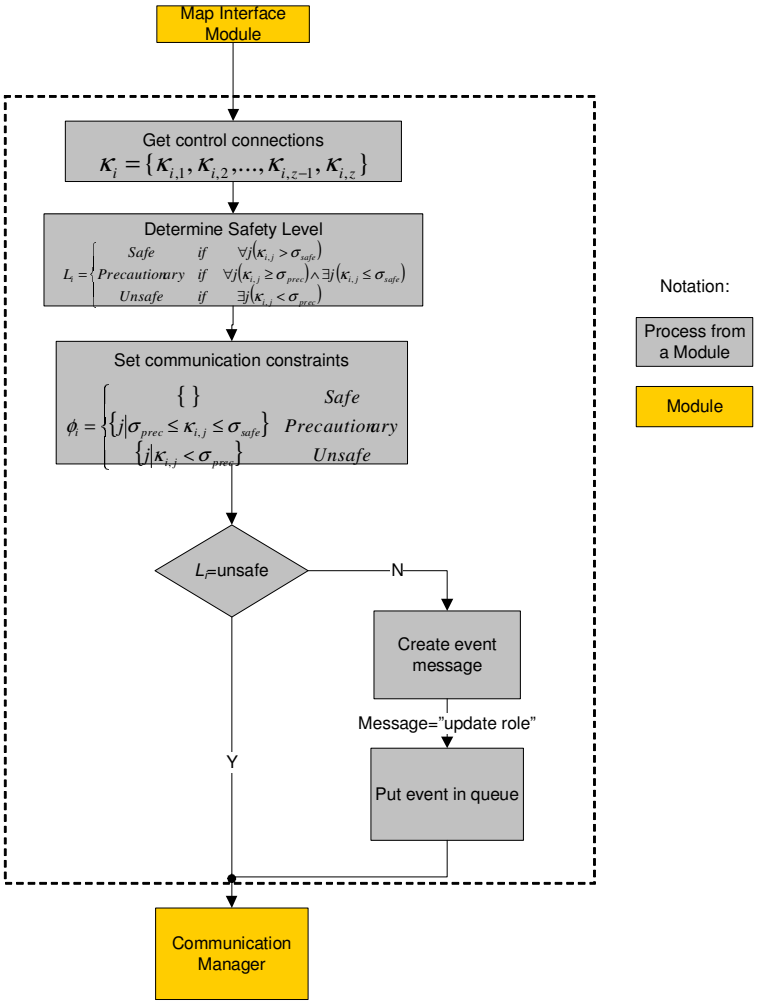


Figure 5.16: Flow Diagram of the Network Manager Module for the Recoverer behavioural role. The Network Manager determines the safety level L_i. When the safety level L_i is not unsafe an "update role" message is sent to the Communication Manager. This message is added to the queue of events handled by the Communication Manager ensuring that the appropriate behaviour is selected in the next iteration.

An Explorer robot has one control connection. Depending on the environment and the positions of the robots an Explorer robot may have at any time more than one direct connection. For instance in open spaces such as a large hall the robots may have several direct connections while in cluttered environments the number of direct connections tends to be smaller. This situation is illustrated in Figure 5.17 for *LOS* communication where in the open space environment the average number of direct connections for the robots is much larger than in the cluttered environment.

Figure 5.18 presents the flow diagram of the implementation of the Network Manager Module for an Explorer robot R_i. Recalling the notation defined in Section 4.3

(page 63) where for a robot R_i: κ_i is the current set of communication constraints, ε_i is the set of direct connections, L_i is the safety level, $\sigma_{safe\_explorer}$ and σ_{prec} are communication thresholds for the safety levels, $t_{network}$ is the time that the robot waits for improvement on the safety level to the safe level. t_{hold} is the time that the robot remains in its location before checking if the safety level has improved. P_c is an auxiliary counter used to determine the time that the robot has remained in its location. $\kappa_{i,current}$ is the signal quality for the current communication constraint. $ID_{current}$ is the ID of the current communication constraint.

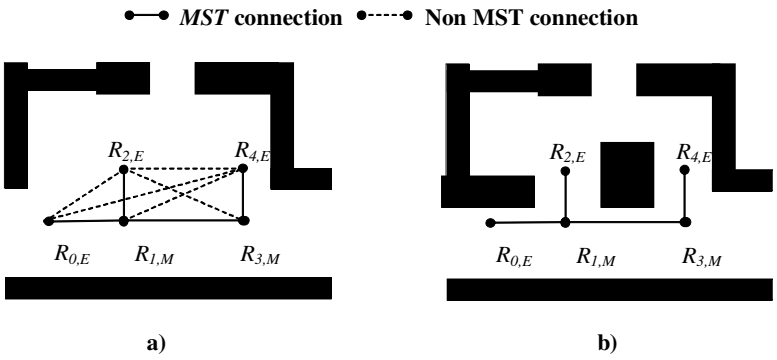


Figure 5.17: An example of the communication network for an open space (a) and a cluttered environment (b). In open spaces the average number of direct connections for the robots is much larger than in cluttered environments.

The first process executed by the Network Manager Module is the request of the signal quality for the control connection from the Communication Manager (Figure 5.2). The signal quality for a connection is obtained from the last beacon signal received from the robot that forms this connection. Afterwards the Network Manager obtains the signal quality for the direct connections and stores the value for the best connection (largest signal quality value) in the *best* variable. An Explorer robot is in the safe level when its best connection has a value above the safe threshold $\sigma_{safe\_explorer}$. The robot is in the precautionary level when the value for the best connection is between the safe and precautionary thresholds $\sigma_{safe\_explorer} > best > \sigma_{prec}$.The robot is in the unsafe level when its best connection has a value below the precautionary threshold σ_{prec}.

When the manager detects that the robot is in the precautionary level the manager waits a time t_{hold} keeping the robot stationary before checking if the safety level has improved. The manager keeps the robot stationary by sending commands to the wheels of the robot hardware platform (Figure 5.2). The manager waits for a maximum time

$t_{network}$ for improvement in the safety level to the safe level. If the safety level does not improve after the time $t_{network}$ has passed the manager assumes that the current goal cannot be achieved because the robot as to become disconnected from the robot network to reach the goal. An event message "t_{exp}" is then generated. This message contains the time during which the Exploration role is inhibited for the robot. Section 4.5 (page 77) presents a discussion about this issue. The "t_{exp}" message is sent to the Communication Manager. The Communication Manager adds the message to its queue of events. The queue is processed and the appropriate behaviour is selected which most of the time is the Pusher behavioural role.

If the robot is in the unsafe level and there are any movements stored by the Robot Motion Controller (Section 5.7) the robot backtracks these movements. The purpose of the backtracking the movements is to return to a previous location where the safety level was not unsafe. After this the manager sends the "t_{exp}" event message that triggers the selection process for the appropriate behavioural role.

When the robot is in the safe level or the level improves to this level after some waiting time the Network Manager checks if there are new connections. If this is the case the manager tries to recalculate the local network. Section 5.9 described the conditions that are necessary to recalculate the local network. When a new local network is created the manager creates an event message whose content is the new local network. This message is sent to the Communication Manager. The Communication Manager adds the message to its queue of events and sends the message to the robots in the local network. The queue is processed and the appropriate behaviour is selected. Upon reception of the message the robots in the local network will reselect their roles according to the updated local network.

If there are no new connections and the best connection is not the current control connection ($ID_{max} \neq ID_{current}$) the manager sets the best connection as the control connection for the robot and updates the local network. The manager creates an event message whose content is the updated local network. The process that occurs with this message was explained in the previous paragraph.

5.10 Communication Protocol and Message Types

In BERODE the robots are assumed to have a communication device with a limited communication range. The robots form a wireless mobile *ad hoc* network (*MANET*) as they explore the environment. This section presents the communication protocol used

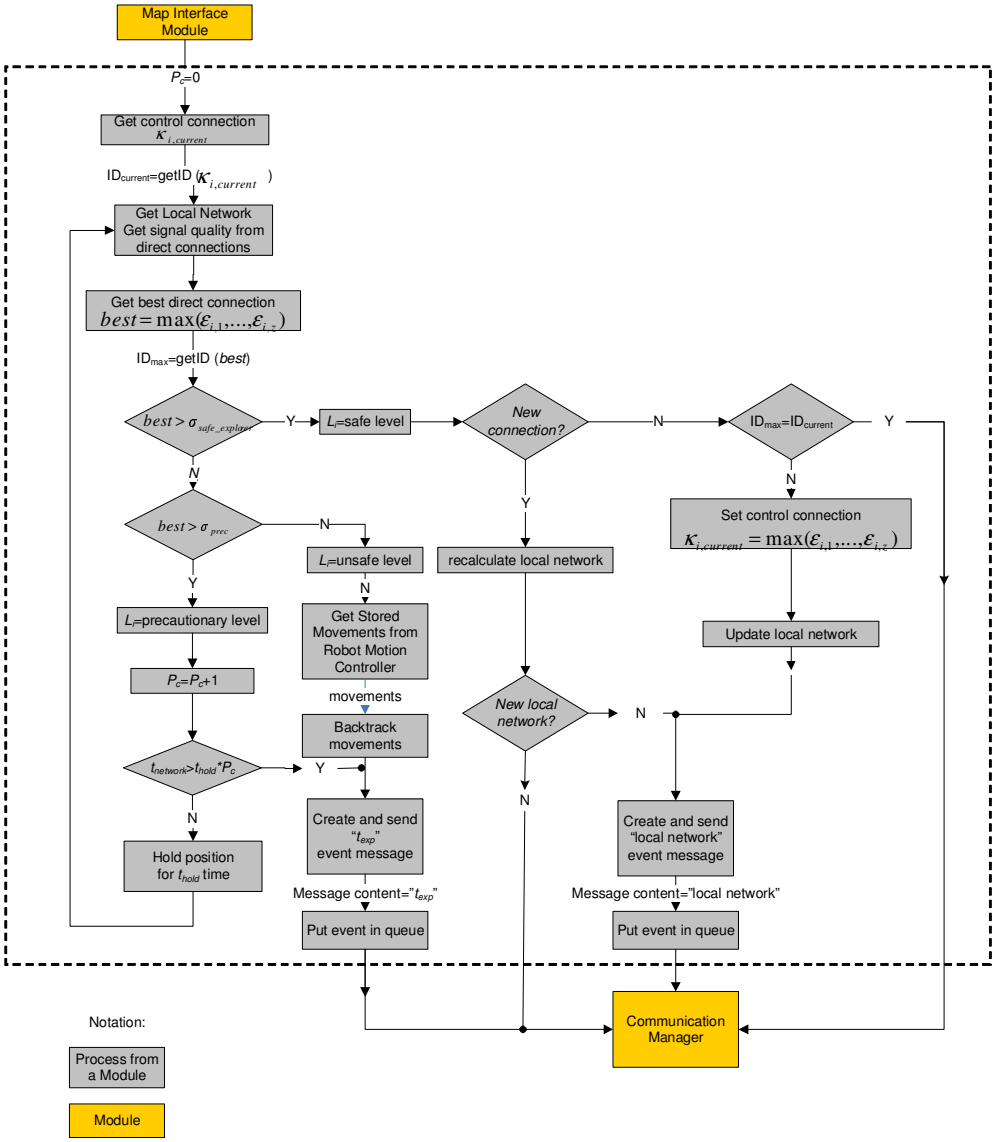


Figure 5.18: Flow diagram for the Network Manager Module for the Explorer behavioural role. The module obtains information about the network from the Communication Manager. The module verifies that the robot is in the safe level. The robot remains in its position for a $t_{network}$ time waiting for improvement if the level of safety is not safe. The robot backtracks its previous movements if the robot is in the unsafe level. When a new connection is detected the Manager tries to recalculate the local network.

by the Communication Manager of a robot to distribute its information and retransmit information received from other robots as necessary.

As the number of robots in a *MANET* increases, communication becomes a bottleneck for the system. Large communication delays that slow down the exploration process are likely to occur if information is shared using simple flooding protocols[15]. For this reason, both the overhead of handling messages and the quantity of information communicated must be minimized. This work argues that the degree of coordination required for a pair of robots is related to their *k-hop distance* on the MST control network (Section 4.4, page 68). To maximize the coordination and minimize communication costs, two levels of communication are proposed: local and global. The local level is composed of the robots inside the local network while the global level is composed of all the robots. Information is shared frequently at the local level, while at the global level information is shared less frequently. Most of the communication is retransmitted within a small number of *hops* thus avoiding scalability problems as the number of robots in the architecture increases. This two level approach ought to be easily extendable to more than two levels to improve scalability results. The two level approach was sufficient for the number of robots used in the simulations (up to 40 robots).

We developed an application level protocol to enable the local and global sharing of information. The protocol implements three types of communication: broadcast, acknowledged, and request. In the broadcast communication a robot transmits information where the confirmation of reception of the message by the receiver robots is not required. In the acknowledged and request communication the confirmation of reception by the receiver robots is required. The acknowledged communication is used by the robots to communicate local information to a group of robots. The request communication is used to gather information from a group of robots. The details of the protocol and the types of communication are presented in Appendix C. The following section presents the types of messages used by the robots to achieve coordination.

5.10.1 Message Types

As previously explained in BERODE the robots distribute their information and retransmit information received from other robots as necessary. The information is used

[15] A simple flooding protocol is based on the broadcast and retransmission of a message. A robot that receives a message retransmits the message. The protocol is inefficient because the robots receive the same message many times over.

for the purposes of control and map building. For control purposes robots share positional and control network information. Positional information is used to coordinate movements and keep the *MANET* as a single connected network. Control network information is used to update the MST control network. For map building purposes robots share information about features extracted from the environment.

This section presents a description and classification of the type of the messages used by the Communication Manager to distribute local information. These messages are generated by the other modules in the architecture. The Communication Manager stores the content of these messages as shown in the Figure 5.3. The Communication Manager transmits these messages when it is called for its sequential execution in the main control loop (Figure 5.4). Table 5.5 presents the messages and their classification where M_T is the maximum retransmission level for the message. Depending on their type, messages are transmitted periodically (periodical type) or on an event basis (event type). Information received from the robot network is stored and retransmitted immediately as necessary to minimise communication delays. The messages have a retransmission level tag which is increased when the message is retransmitted. A message is retransmitted when the retransmission level tag is smaller than the maximum retransmission level for the message. The messages are used for control and map building purposes.

Local features are used as temporary navigational aids by the robots. They are broadcast to minimize communication requirements; their loss does not affect the map representation because the observations from these features become part of the map via the global feature update process (Section 5.8). For this reason they are classified as control purpose messages (i.e. not map building).

A message of the event type is generated when the MST control network is modified. A robot may decide to modify the MST control network (partially or completely) to try to improve the signal quality.

The robot position message is used as the beacon signal by the robots. The new goal message is used by the Explorer robots to notify the robots about the current area that the robot is trying to explore. The Switch control message is used in the Network Manager Module of the Explorer robots to improve the safety level (Section 5.9.3). The message is sent within the *k-hops* to ensure that all the robots maintain the same MST control network. The role message is sent once a robot makes a role transition (Section 5.4). The local internal feature message is used to periodically distribute the recently extracted features within the local network. The global internal feature message is used

to periodically distribute the recently extracted features to the entire robot network. The Local and Global MST messages are used once a robot has recalculated either the local network or the MST control network respectively (Section 4.4, page 68). Two types of messages are used during the recalculation of these networks. The first one is a request message to gather the necessary information from all the robots inside the network being recalculated. The second one is an acknowledged message used to transmit the recalculated network to the robots inside the recalculated network.

Message Content	M_T	**Type of Transmission**	**Message Purpose**	**Message Type**	**Description**
Robot position	1	Broadcast	Control	Periodical	The estimated position of the robot (x,y,θ)
new goal	n	Acknowledged	Control	Event	The location that the robot is currently trying to reach (x,y)
Switch control	k	Acknowledged	Control	Event	recalculated local network
Local MST	k	Request, Acknowledged	Control	Event	recalculated local network
Role	k	Acknowledged	Control	Event	A robot modified its current behavioural role
Local internal feature	k	Broadcast	Control	Periodical	Parameters of the features extracted since the last local feature transmission
Global internal feature	n	Acknowledged	Map Building	Periodical	Parameters of the features extracted since the last global feature transmission
Global MST	n	Request, Acknowledged	Control	Event	Recalculated MST control network

Table 5.5: Types of messages transmitted by the Communication Manager. The messages have a retransmission level tag which is increased when the message is retransmitted. A message is retransmitted when the retransmission level tag is smaller than the maximum retransmission level M_T for the message.

5.11 Summary

This chapter has presented the implementation of the BERODE (BEhavioural ROle DEcentralized) architecture in individual robots. The BERODE architecture is a decentralized architecture to explore environments using a group of robots with short range communication. To improve coordination and minimize task overlapping the robots are kept as a single communication network. The BERODE architecture is based on behavioural roles such as Explorer and communication Maintainer. The robots select their behavioural role based on their status in the control network and their internal state.

The exploratory behaviour of the robot network emerges from the interaction of the individual roles. This interaction is achieved through the imposition of *virtual spring* forces for the connections in the MST control network. The *virtual spring* forces keep the robot network connected while moving towards unexplored areas. The MST control network is a minimum spanning tree of the communication network that contains only the necessary connections to keep the communication network connected. This network is calculated at the beginning of the exploration process by a designated robot[16] and modified when the signal quality of the MST connections can be improved. After the initial process is executed the robots retain knowledge of their local network. The local network for a robot is the network that contains all the robots within a *k-hop* distance.

In BERODE each robot builds its own feature based map. The positions of the robot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID number. The robots periodically distribute the features that they observe with the rest of the team. A robot incorporates the received features from others to its map using the same process as for locally observed features. This is possible because the robots use the same global Cartesian frame of reference.

The architecture has been implemented using modules that are sequentially executed (Figure 5.1). The modules acquire information from other modules by means of queries (Figure 5.2). Each module addresses a specific task (e.g. the Collision Avoidance Module prevents the robot from colliding with obstacles).The implementation and parameterization of some of the modules depends on the behavioural role. The

[16] In the experiments the robot with the lowest ID number is the designated robot. The robots have an ID number that allows them to identify each other in the network.

following paragraphs present a summary of the tasks for modules according to their execution order in the main control loop.

The Communication Manager of a robot handles the information related to the robot network. This module executes three processes: transmitting periodical beacons, handling the exchange of information with the network of robots, and maintaining a queue of events which is processed when the module is called. The first two processes are executed in parallel to the main control loop. These processes are updating processes that do not disturb the execution sequence of the main control loop. A module that uses information stored in the Communication Manager as input for its execution obtains this information at the start of its execution and uses this frozen information throughout all its execution.

The beacon signals contain the most recent estimated robot position and its uncertainty, which are calculated by the Map Interface Module. Information received from the robot network is stored and retransmitted immediately as necessary to minimise communication delays.

The Communication Manager maintains a queue of events. An event occurs when the internal state of the robot has changed or when the MST control network has been modified. The MST control network can be modified by any robot inside the modified network. The robots select their role according to their internal state and their connection state in the MST control network. The Communication Manager calls the Behaviour Selection Module when the queue of events is not empty otherwise it calls the Collision Avoidance Module. The Behaviour Selection Module selects the appropriate behavioural role for the robot. The Collision Avoidance Module is the module that ensures the safety of the robot by detecting collisions with other robots and with static obstacles.

The Planning Module is called afterwards either by the Collision Avoidance Module or the Behaviour Selection Module. The Planning Module creates a suitable plan for the current behavioural role. The plan is communication sensitive which keeps the robot connected to the network. The non-Explorer robots generate reactive plans to keep the network connected. These plans are based on the imposition of *virtual forces* by the robot's direct connections on the MST control network. An Explorer robot generates a plan to move towards the most attractive unexplored area of the environment. The most attractive area is the safest area with respect to communication with the largest utility. The utility of an area is a function of the size of the area and the path length from it to the Explorer robot. The Planning Module is the module that deter-

mines when the map is complete[17]. The exploration stop once any one robot considers the map as complete.

The Planning Module generates a path for the plan that serves as input for the Robot Motion Controller. The Robot Motion Controller generates the motion commands for the current path.

The Map Interface Module is called by the Robot Motion Controller once the robot has executed a movement. The Map Interface Module obtains sensor measurements and extracts features from these sensor measurements. These measurements are used to build and update the feature map of the robot. The feature map is updated using an *EKF* which estimates the location and uncertainty of the robot and the features. In BERODE the robots periodically distribute their locally extracted features. The Map Interface Module is the module that stores and distributes these features. Once the feature map has been updated the Network Manager Module is called.

The Network Manager Module monitors the direct connections of the robot and tries to update the local network when a new connection is detected. This module ensures that the appropriate behaviour is executed according to the current level of safety. When the current behaviour is not appropriate the module generates an event that triggers the reselection process. The Network Manager Module is the last module in the main control loop. After this module finishes its execution a new iteration of the main control loop is started. The process stops once the Planning Module of any one robot has determined that the map has been completed.

This chapter has described the implementation of the BERODE architecture in individual robots. Chapter 6 presents the Map Building Module used by the robots to build their feature maps. Chapter 7 presents experiments that show that the sensors and communication models used in our simulations are reasonable and conservative approximations to the experimental data. Chapter 8 presents the implementation of the approach for *LOS* and *RF* technologies. Chapter 9 presents simulations that show the robustness, the scalability, and the efficiency of BERODE. Our simulations show that the approach is robust to infrequent communication, scalable with respect to communication and number of robots, and more efficient than approaches with fixed control topologies. We can't conclude that these properties would still transfer to a real world implementation, but our simulations incorporate measured aspects of the robot sensors and communication devices to improve the realism of the simulations. Chapter 10

[17] A map is considered to be complete once it is projected into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a user defined threshold (Section 8.6, page 239).

presents experiments that analyze the consistency of the maps built by the simulated robots. Chapter 11 presents the conclusions of this thesis.

Chapter 6

Map Building Module

6.1 Introduction

This chapter presents the Map Building Module for the multi-robot architecture proposed in this thesis. The module has to perform two concurrent tasks, build a map of the environment as the robot moves and obtain estimates of its location in this map. This problem is known as Simultaneous Localization and Mapping (*SLAM*) and the different approaches to solving it have been discussed in Chapter 3. It was concluded that an Extended Kalman Filter (*EKF*) approach is the most suitable approach to build a representation of the environment using *low cost* sensors (e.g. sonar, infrared). The *EKF* is a feature based representation of the environment. Features are extracted from raw sensor data. A *feature management* process extracts, segments and associates the features. The extracted features are then used to update the *EKF* and improve the estimates of the robots and feature locations.

The chapter begins with the general description of the *EKF* in Section 6.2. Section 6.3 presents the components of the Map Building Module and its interaction with the *EKF*. A flow diagram describes the execution sequence of the *feature management* and *EKF* components. Section 6.3.1 introduces the feature map representation of the state of the robot and its environment. The representation is an augmented version of the common version that allows the extraction of information from multiple locations. Section 6.3.2 presents the vehicle model used to describe the robot movement. Section 6.3.3 describes the feature extraction process for multiple locations. Line and point features are extracted using a *RANSAC* (Random Sample Consensus) approach. Attention is then turned to description of the prediction and update stages of the *EKF* (Sections 6.3.4 and 6.3.5). Section 6.3.6 describes the addition of new features to the

map under uncertainty. Section 6.3.7 describes the prediction models for the features. These models are used in the data association process (Section 6.3.8) to determine the pairings between the extracted features and the features in the map representation. Section 6.3.9 discusses the efficiency of the data association process. Section 6.3.10 describes the integration of structural information (parallelism, perpendicularity, colinearity) in the feature map. Structural information is integrated by means of *virtual observations.* The chapter ends with a summary of the qualities of the Map Building Module.

6.2 The Extended Kalman Filter

The Extended Kalman Filter is a recursive least squares estimator. It produces at time t a minimum mean-squared error estimate $\hat{x} = (t\,|t\,)$ of a state vector $\hat{x}(t)$. This estimate is obtained by fusing a state estimate prediction $\hat{x}(t\,|t-1)$ with an observation $z(t)$ of the state vector $\hat{x}(t)$. The estimate $\hat{x}(t\,|t\,)$ is the conditional mean of $\hat{x}(t)$ given all observations $Z^t = \{z(t)...(z(t)\}$ up until time t.

$$\hat{x}(t\,|t\,) = E[x(t)|\,Z^t] \tag{6.1}$$

where $E[\hat{x}(t)|\,Z^t]$ is the expectation of the state vector $\hat{x}(t\,|t\,)$ given Z^t. The state at time t conditioned on the information up to time t-1 is referred as prior estimate $\hat{x}(t\,|t-1)$, while the state at time t given the information up to time t is referred as posterior estimate $\hat{x}(t\,|t\,)$. The prior estimate is calculated in the prediction stage of the *EKF*. This stage is described in Section 6.2.5. The posterior estimate is calculated in the update stage of the *EKF*. This stage is described in Section 6.2.6. The following section describes the role of the *EKF* in the *SLAM* problem.

6.3 Feature Based Localization and Mapping

Feature based Localization was first introduced by Smith et al. (1988), where an *EKF* was implemented. The position of the robot and the environmental features are collected in a state vector. When features are re-observed the filter is updated. When the robot enters unexplored areas, the state vector is augmented with the new features. The EKF has two stages: prediction and update. In the prediction stage, odometry information is used to predict the state at the next time step $\hat{x}(t\,|t-1)$. In the update stage, measurements of features are used to update the robot position and the mapped

features $\hat{x}(t\,|t\,)$. Measurements used to update the filter are extracted using an external feature extraction and data association process.

Figure 6.1 shows the diagram for the localization and mapping module. The map representation is an extended version developed by Leonard et al. (2002) to allow the mapping of partially observable features using multiple viewpoints. A new viewpoint is generated and a new set of measurements is collected once the robot has moved a certain minimal distance from the previous viewpoint. When the robot has not moved a distance large enough to add a new viewpoint or there are no new features observed the predicted state becomes the estimated state $\hat{x}(t\,|t\,) = \hat{x}(t\,|t-1\,)$ for the following iteration of the state estimation.

The data association process is carried out every time new measurements from the environment are available. This process has three stages: feature extraction, prediction and matching. In the feature extraction stage the information is segmented and a geometric representation of the features is obtained. In the prediction stage the mapped features are projected with respect to the current position of the robot. In the matching stage the correspondences between the projected mapped features and the new features are resolved and the filter is updated. Features that did not match mapped features are tracked before they are added as new states. Once a feature has been tracked and not discarded as noise it is integrated into the state vector and initialized as a new feature (Figure 6.1). The state vector is then augmented by adding the feature to the state vector.

The following sections present the robot motion model used in the prediction stage of the filter, the feature model, the feature based map representation, and the prediction and update stage stages of the *EKF*.

6.3.1 The Map Representation

The feature map introduced by Smith (Smith et al., 1988) consists of a state vector $\hat{x}(t\,|t\,) = \begin{bmatrix} x_r(t\,|t\,) \\ x_f(t\,|t\,) \end{bmatrix}$ where $\hat{x}_r(t\,|t\,)$ and $\hat{x}_f(t\,|t\,)$ are the robot and feature state estimates at time t. When a new feature is detected the state vector is augmented and the feature is initialized as described in Section 6.3.6. In Smith's (Smith et al., 1988) approach it is assumed that the state of the new feature can be determined using the information extracted at the last robot position. While this is typically the case for precise and usually expensive sensors (e.g. laser, vision) it is not generally the case for cheap sensors (e.g. sonar). For instance sonar measurements have one *DOF*, their distance

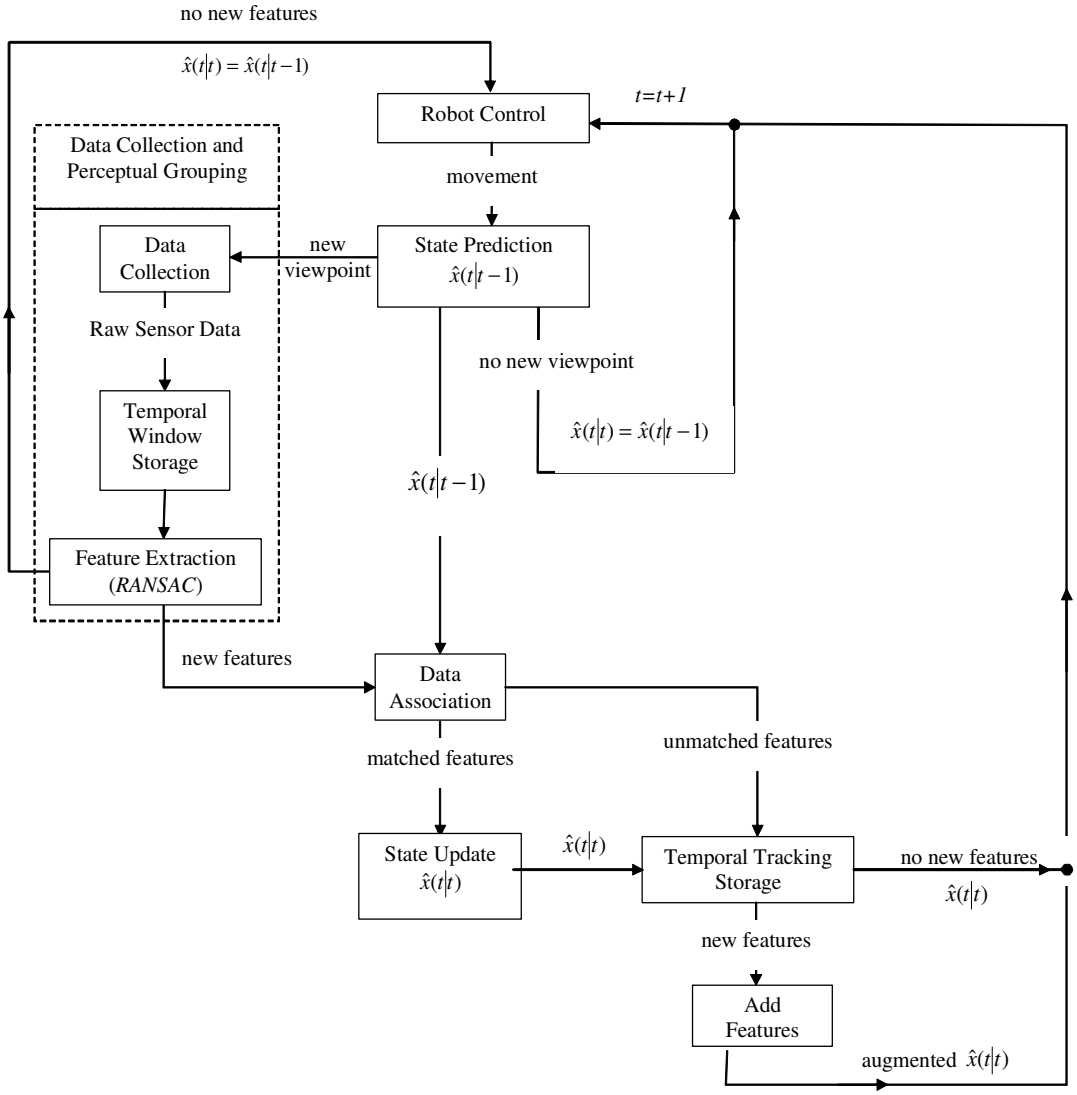


Figure 6.1: Flow diagram for the feature based localization and mapping module. The estimated state vector $\hat{x}(t\,|\,t\,)$ contains the positions of the robot and the features. The localization process has two stages: prediction and update. In the prediction stage the state at the next time step is estimated $\hat{x}(t\,|\,t-1)$. In the update the stage the state is updated $\hat{x}(t\,|\,t\,)$ using the information from new features observed.

to the closest obstacle. Their angular precision is limited typically in the range of 25 ° to 45 ° which is too large to be a useful constraint on the *DOF*. Line and point features have two *DOF* thus these features cannot be initialized from a single position.

The maintenance of sensing positions and the use of a temporal window for storage of the sensor readings allows the grouping of perceptions from multiple sensing positions. A sensing position is a position where the robot acquires sensor readings. The proposed Map Building Module integrates a Data Collection and Perceptual Grouping process (Figure 6.1). In this process the sensing positions and their readings are stored in a temporal window. An extraction process is then used to extract features from the temporal window. This process is described in Section 6.3.3.

The feature map is then redefined to incorporate the current position of the robot, the last m robot sensing positions $\hat{x}_m(t\,|t\,) = \left[\hat{x}_r^{t-m}(t\,|t\,)^T, ..., \hat{x}_r^{t-1}(t\,|t\,)^T\right]$, and all of the N features identified in the environment and is defined as

$$\hat{x}(t\,|t\,) = \left[\begin{array}{cccccc} \hat{x}_r(t\,|t\,)^T & \hat{x}_r^{t-m}(t\,|t\,)^T & \cdots & \hat{x}_r^{t-1}(t\,|t\,)^T & \hat{x}_{f_1}(t\,|t\,)^T & \cdots \quad \hat{x}_{f_N}\ (t\,|t\,)^T \end{array}\right] \quad (6.2)$$

Every time a new sensing position is added to the state, the oldest sensing position is removed from the state vector. The number of positions necessary to guarantee the extraction of some features from multiple sensing positions depends on the range of the sensors and the distance between their measurements.

The associated error for the augmented state vector $\hat{x}(t\,|t\,)$ is a covariance matrix, $C(t\,|t\,)$ which represents the error in the robot positions, and feature locations, and their cross-correlations:

$$C(t\,|t\,) = \left[\begin{array}{ccc} C_{rr}(t\,|t\,) & C_{rm}(t\,|t\,) & C_{rf}(t\,|t\,) \\ C_{mr}(t\,|t\,) & C_{mm}(t\,|t\,) & C_{mf}(t\,|t\,) \\ C_{fr}(t\,|t\,) & C_{fm}(t\,|t\,) & C_{ff}(t\,|t\,) \end{array}\right] \quad (6.3)$$

6.3.2 Robot Motion Model

Between time t and t-1 the robot receives a control input $\hat{u}(t)$ that defines a small increment in the position and orientation with respect to the current robot position $\hat{x}_r(t-1\,|t-1\,)$, the position of the robot at time t is then defined as

$$\hat{x}_r(t\,|t-1) = f(\hat{x}_r(t-1\,|t-1), \hat{u}(t)) \quad (6.4)$$

The Koala robot used in the investigation from Section 7.9.2 is a holonomic robot consisting of two wheels separated by a wheelbase b. Each wheel measures its differential motion based on the odometry measurements $\hat{u}(t) = \left[\begin{array}{cc} E_R & E_L \end{array}\right]$. Our simulated

robots are based on this robot. The motion model is frequently updated using a small time step. In general the robot position $\hat{x}_r(t\,|t-1)$ in Cartesian coordinates for this model at time t is defined as

$$\hat{x}_r(t\,|t-1) = \begin{bmatrix} x(t\,|t-1) \\ y(t\,|t-1) \\ \theta(t\,|t-1) \end{bmatrix} = \begin{bmatrix} x(t-1\,|t-1) + l\cos(\theta(t-1\,|t-1) + \phi/2) \\ y(t-1\,|t-1) + l\sin(\theta(t-1\,|t-1) + \phi/2) \\ \theta(t-1\,|t-1) + \phi \end{bmatrix} \tag{6.5}$$

where

$$l = \frac{E_R + E_L}{2} \tag{6.6}$$

$$\phi = \frac{E_R - E_L}{2b} \tag{6.7}$$

The error associated with the measurements is assumed to be independent and Gaussian and is defined as

$$Q_r = \begin{bmatrix} \sigma_R & 0 \\ 0 & \sigma_L \end{bmatrix} \tag{6.8}$$

where σ_R and σ_L are the right and left Gaussian odometric errors. These values are obtained from experiments with the Koala robot (Section 7.5, page 178).

6.3.3 Feature Model Extraction from Multiple Locations

As discussed in Section 3.1 (page 35) indoor environments have as their main stable features walls, corners, and columns. These features provide sufficient information for robots to build and maintain a map representation. Walls can be modelled as line features, while corners and columns can be modelled as point features. Line features are represented in polar coordinates while point features are represented in Cartesian coordinates. Both types of features have 2 *DOF* (Degrees of Freedom).

Most exploration systems are based on the assumption that there is enough measurement data available from a single robot position to extract new features. The sensing platform designed for the proposed architecture is built from *low cost* sensing devices (sonar and infrared). The extraction of line and point features from sonar devices cannot be achieved from a single position since each sonar measurement has only one *DOF*, the distance to the nearest surface. The bearing is only known within some range defined by the beam-width, typically 25° to 45°. However features can be extracted by grouping measurements from multiple close positions. To achieve this, a temporal window of measurements is maintained (Figure 6.2). A new set of readings from

p sonar sensors is added every time the robot moves a certain distance. The temporal window retains the measurements from the last m positions; the oldest position is replaced by the most recent one to maintain a fixed maximum size.

As discussed in Section 3.7 (page 48) two main methods have been used to group perceptual information from sonar readings, Hough transform and *RANSAC*. We implement a *RANSAC* based approach because it's more computationally efficient and robust to environment noise than a Hough transform approach. The basic idea in *RANSAC* is to allow probable precision to be traded against computational time, by randomly selecting a pairs of readings, generating a hypothesis, and then counting how many of the remaining measurements agree with the hypothesis. If the number of measurements that agree with the hypothesis is above a user defined threshold the hypothesis is accepted. This threshold was obtained in preliminary experiments as explained in Section 7.9.1 (page 213).

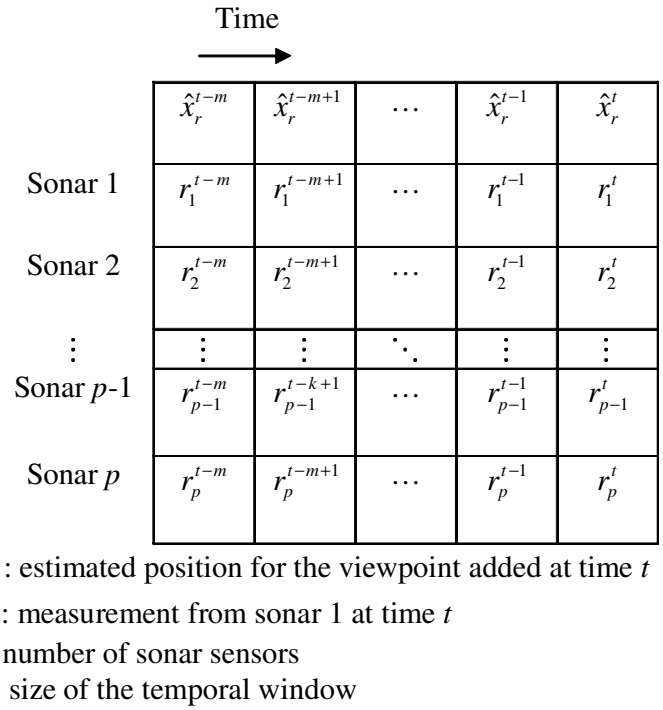


Figure 6.2: Temporal window for sonar readings. New sets of readings are added once the robot has moved a certain minimum distance. $\hat{x}_r^t$ is the position for the viewpoint added at time t to the temporal window for which the sonar readings $r_1^t, \ldots, r_p^t$ were taken.

The remainder of this section presents the formulation of the point and line models from pairs of positions and details of the *RANSAC* method implemented. The feature

models presented are based on Leonard's models (Leonard et al., 2002).

6.3.3.1 Point Model

Point features result from sonar echoes whose arcs intersect at the same point. The intersection point of two sonar echoes is determined by intersecting the circles of the two arcs and then determining which of the two solutions is inside the beam-width of the echoes. According to Leonard (Leonard et al., 2002) the intersection point $P(x,y)$ from the view points $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$ with range measurements r_1 and r_2 is calculated by obtaining the determinant δ:

$$\delta = \sqrt{((r_2+r_1)^2-d^2)\,(d^2-(r_2-r_1)^2)} \tag{6.9}$$

where

$$d^2 = (x_2-x_1)^2+(y_2-y_1)^2 \tag{6.10}$$

The intersection points are determined using the two possible signs of δ in the formulae:

$$x = \frac{1}{2}\left(\frac{\mp(y_2-y_1)\delta-(r_2^2-r_1^2)(x_2-x_1)^2}{d^2}+(x_1+x_2)\right) \tag{6.11}$$

$$y = \frac{1}{2}\left(\frac{\pm(x_1-x_2)\delta-(r_2^2-r_1^2)(y_2-y_1)^2}{d^2}+(y_1+y_2)\right) \tag{6.12}$$

where depending on the value of δ two, one or no solutions are found (Figure 6.3).

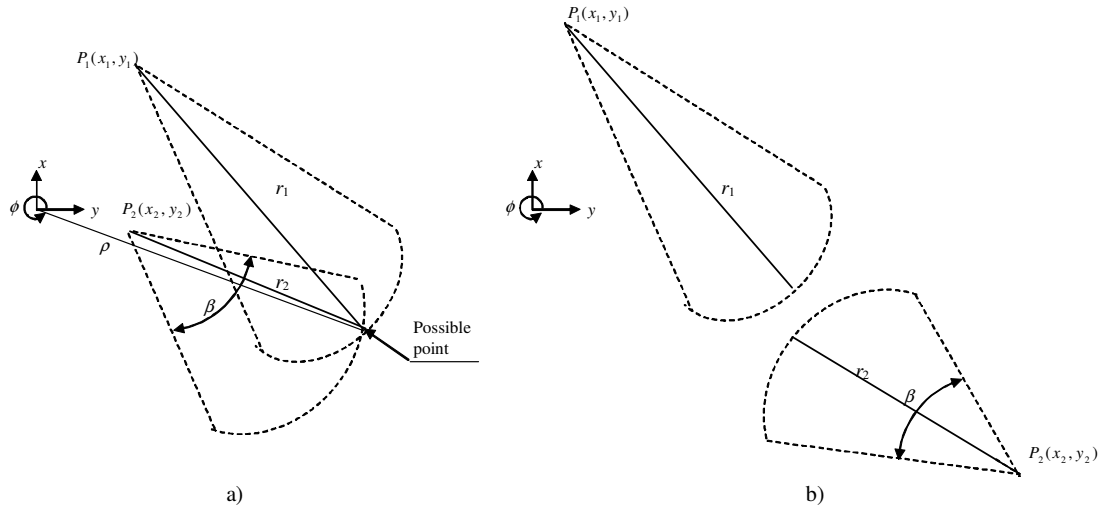


Figure 6.3: Model for a point obtained from the intersection of two view points $P_1(x_1,y_1)$ and $P_2(x_2,y_2)$ with range measurements r_1 and r_2. In (a) there is a possible solution whereas in (b) there is no solution.

6.3.3.2 Line Model

Line features are the result of multiple cotangent echoes. There are four possible solutions, but in the case of sonar processing the cotangent lines are required to be on the same side with respect to the circles (Figure 6.4). Following a similar formulation to the point model δ is defined as:

$$\delta = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 - (r_2 - r_1)^2} \tag{6.13}$$

where the viewpoints $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ have the range measurements r_1 and r_2 respectively. When δ is imaginary, the circles are concentric and do not have a cotangent. The line parameters $L(\rho, \phi)$ are:

$$\rho = \frac{\pm\delta(x_1y_2 - x_2y_1) - (y_2 - y_1)(r_1y_2 - r_2y_1) - (x_2 - x_1)(r_1x_2 - r_2x_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{6.14}$$

$$\phi = \arctan\left(\frac{\mp\delta(x_2 - x_1) - (y_2 - y_1)(r_2 - r_1)}{\pm\delta(y_2 - y_1) - (x_2 - x_1)(r_2 - r_1)}\right) \tag{6.15}$$

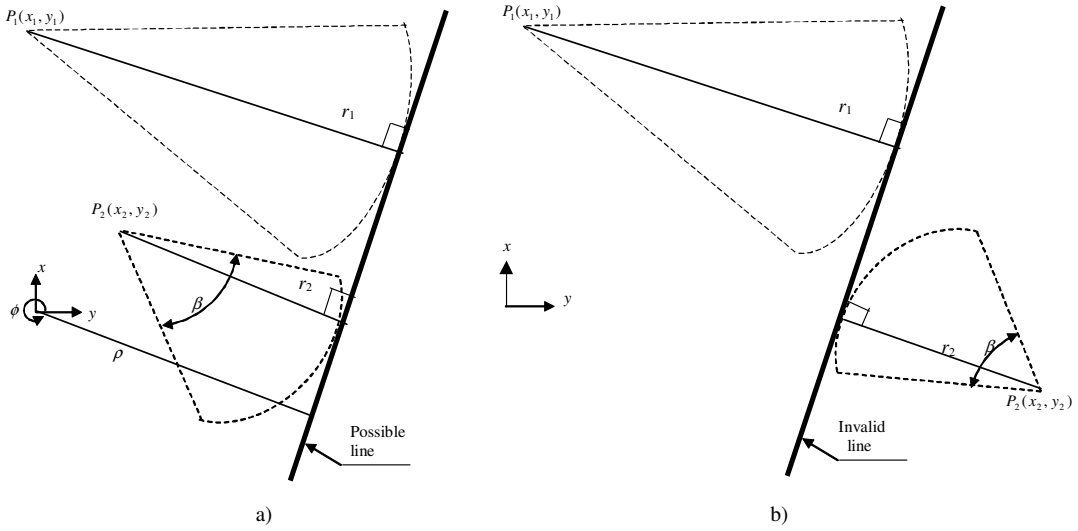


Figure 6.4: Model of a line obtained from two cotangent view points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ with range measurements r_1 and r_2. In (a) the line hypothesis is valid because the line is on the same side with respect to the viewpoints while in (b) this is not the case.

6.3.3.3 Efficient Data Association for RANSAC

RANSAC is a process that extracts one feature per trial. In every trial *RANSAC* selects randomly N_{pairs} pairs of readings and generates a hypothesis for each pair of readings

(if any). N_{pairs} is a value determined according to the success rate desired. The details can be found in Fischler and Bolles (1981).

Every hypothesis for the trial is tested against the remaining readings in the temporal window and the one with the largest consensus (most votes) is extracted. The readings associated with the extracted feature are marked as used to avoid the association in subsequent trials and a new trial is initialized. Trials stop once the consensus for the best hypothesis is below a certain threshold.

This process is not efficient when using a temporal measurement window (Figure 6.2) because in most occasions the recently acquired measurements are not related to the oldest ones. The consequence is that many trials are wasted in the formulation of invalid hypotheses. Reducing the size of the temporal window does not solve this problem because there is not enough information to extract features. Additionally, feature uncertainty is larger when few readings are associated with the feature. The parameter estimation then becomes unreliable.

Bosse (2003) proposed an efficient approach to address this issue. This work implements Bosse's approach. The approach uses preprocessing tests to determine which measurements may match. Pairs of measurements that may match are labelled as *compatible*. *RANSAC* is then applied selecting random *compatible* pairs to generate the hypothesis. The number of *compatible* pairs is considerably smaller than the total of possible pairs for the temporal window. The selection process is then performed in a smaller search space that contain more valid hypothesis. As a consequence fewer trials are required because fewer trials produce invalid hypotheses.

Every time a new set of readings is incorporated to the temporal window (Section 6.3.3), the new readings are tested against every previous reading in the window (Figure 6.2). Two tests are realized, one for point and one for line features. If a test is passed the readings are labelled as *compatible*. The result of this process can be visualized as two graphs (one for point features and one for line features). In these graphs the edges represent the *compatibility* between the pairs of readings.

The test for matchable point features first determines the distance between the centers of the two arcs from the sonar readings. If this distance is greater than the average width of the two beams at the range point the pair cannot match. Additionally if δ from Eq. 6.9 is imaginary the pair cannot match.

The test for matchable line features checks whether the pair comes from the same general direction. First, to generate a possible match the angle between the center of the two sonar readings must be smaller than the sonar beam-width angle. Then the

difference between the projected centers of the sonar readings to the approximated normal of the line is calculated. The approximated normal is calculated as the average direction of the sonar readings. If this difference is larger than a threshold the pair of readings cannot match. The threshold is user determined depending on the expected sensor noise and the beam-width angle. Additionally, if δ from Eq. 6.13 is imaginary the pair cannot match.

The following section presents the formulas used to estimate the covariance of the extracted features.

6.3.3.4 Measurement Covariance Matrix

Following Bosse's (2003) formulation the measurement covariance matrix for a feature extracted associated to i measurements is estimated as

$$R = \tau \left(\sum_{m=1}^{i} H_m^T H_m \right)^{-1} \tag{6.16}$$

where τ is the measurement uncertainty. In our implementation this value was obtained from our experiments with the hardware sensor devices (Section 7.6, page 185). H_m is the residual function of the m^{th} associated measurement.

For an extracted line feature $L(\rho,\phi)$ the function is defined as

$$H_m = \begin{bmatrix} -1 & y_m \cos\phi - x_m \sin\phi \end{bmatrix} \tag{6.17}$$

where (x_m, y_m) are the Cartesian coordinates of the m^{th} associated measurement.

For an extracted point feature $P(x,y)$ the function is defined as

$$H_m = \begin{bmatrix} x - x_m & y - y_m \end{bmatrix} \tag{6.18}$$

where (x_m, y_m) are the Cartesian coordinates of the m^{th} associated measurement. The details of the formulation are presented in (Bosse, 2003).

6.3.4 EKF Prediction

The robot positions and the feature locations are referred with respect to the initial position of the robot. The initial position of the robot is used then as the global coordinate system[1]. As the robot moves through the environment the estimation of its position

[1] In the simulations with multiple robots the simulated robots share a common global coordination system based on the initial positions of the simulated robot with the lowest ID number. The relative positions of the other simulated robots are known a priori with a small uncertainty.

changes. The estimated position $\hat{x}_r(t\,|t-1)$ at time t is a function of its position at a previous time $\hat{x}_r(t-1\,|t-1)$ and a control input $\hat{u}(t)$ and is defined as

$$\hat{x}_r(t\,|t-1) = f(\hat{x}_r(t-1\,|t-1), \hat{u}(t)) \tag{6.19}$$

where $f(\hat{x}_r(t-1\,|t-1), \hat{u}(t))$ has been defined in Eq. 6.4 for the robot used in our investigations and simulations. The covariance matrix for the state vector is defined as:

$$C(t\,|t-1) = F_{x^r} C(t-1\,|t-1) F_{x^r}^T + F_u Q_r F_u^T \tag{6.20}$$

The covariance matrix for the robot evolves as

$$C_{rr}(t\,|t-1) = F_{x^r} C_{rr}(t-1\,|t-1) F_{x^r}^T + F_u Q_r F_u^T \tag{6.21}$$

where Q_r is the covariance matrix of the process noise and F_{x^r} and F_u are the Jacobians of $f(\hat{x}_r(t-1\,|t-1), \hat{u}(t))$ evaluated at $(\hat{x}(t-1\,|t-1), \hat{u}(t))$. For the proposed odometry model the Jacobians are:

$$F_u = \frac{\partial f(\hat{x}_r, \hat{u})}{\partial \hat{x}_r} = \begin{bmatrix} \cos(\theta(t-1\,|t-1)+\phi) & -d\sin(\theta(t-1\,|t-1)+\phi) \\ \sin(\theta(t-1\,|t-1)+\phi) & d\cos(\theta(t-1\,|t-1)+\phi) \\ 0 & 1 \end{bmatrix} \tag{6.22}$$

$$F_{x^r} = \frac{\partial f(\hat{x}_r, \hat{u})}{\partial \hat{u}} = \begin{bmatrix} 1 & 0 & -d\sin(\theta(t-1\,|t-1)+\phi) \\ 0 & 1 & d\cos(\theta(t-1\,|t-1)+\phi) \\ 0 & 0 & 1 \end{bmatrix} \tag{6.23}$$

The covariance for the features is not altered by the movement of the robot; however the correlations between the positions of the robot and the features change. Following Chatila and Moutarlier (1989) the correlation $C_{ri}(t-1\,|t-1)$ between the robot and the i^{th} feature in the map is updated as

$$C_{ri}(t\,|t-1) = F_{x^r} C_{ri}(t-1\,|t-1) \tag{6.24}$$

where i=1, ..., N and N is the number of features in the map at time t-1.

6.3.5 EKF Update

At the update stage features are extracted from the sensors and a matching process determines the pairings between the extracted features and the features in the current stochastic map. In typical scenarios more than one feature correspondence is found, and the update process can be done processing one feature at a time or in a single

simultaneous update. In the simultaneous case a measurement vector is composed. Newman's (Newman, 1999) research shows that although simultaneous processing of features saves some processing time the computational complexity of the algorithm is the same; moreover for numerical reasons it is better to process features one at a time. For these reasons in this implementation the features are processed one at a time. In general, when at time t an observation $z_i(t)$ of the i^{th} feature becomes available the map is updated applying the *EKF* equations as follows

$$\hat{x}(t\,|t\,) = \hat{x}(t\,|t-1) + K\,(z_i(t) - h_i\,(t\,|t-1)) \tag{6.25}$$

$$C(t\,|t\,) = (I - KH_i)\,C(t\,|t-1) \tag{6.26}$$

where K is the Kalman gain, $h_i\,(t\,|t-1)$ is the predicted position of the feature, and H_i is the Jacobian of the observation model h_i with respect to $\hat{x}$ evaluated at $\hat{x}(t\,|t-1)$. The equations of the observation model $h_i\,(t\,|t-1)$ and the Jacobian H_i for the line and point features are presented in Section 6.3.7. Section 6.3.3 presented the feature extraction model for the observation $z_i(t)$ for line and point features. The Kalman gain is given by

$$K = C(t\,|t-1)H_i^T\left(H_i C(t\,|t-1)H_i^T + C_z^i\right)^{-1} \tag{6.27}$$

where C_z^i is the measurement covariance matrix (defined in Section 6.3.3.4).

As noted by Chatila and Moutarlier (1989) the matrix H_i is a sparse matrix, and as the mapped features increase the sparseness grows. By taking this into account the computational complexity is reduced from $\mathrm{O}(N^3)$ to $\mathrm{O}(MN^2)$, where M is the number of observed features, and N is the number of features in the current map.

6.3.6 Addition of New Features

New features are included in the state vector once it has been determined that they do not match any feature and have not been discarded as noise features. All this stage occurs at time t after the update stage of the *EKF* (Figure 6.1) that incorporates all the information up until time t. For this reason we omit the time labelling in this section to have a clearer notation.

The features are added to the estimated feature map $\hat{x}$ which contains N features. The new feature state $\hat{x}_{N+1}$ is determined using a transformation function $g(\hat{x}_r, \hat{z}_{N+1})$ that transforms the relative measurement $\hat{z}_{N+1}$ to the global reference frame according to the robot position $\hat{x}_r$. The transformation function for the line and point features and

their Jacobians are presented in the following subsections. The augmented state vector contains N+1 features and is defined as

$$\hat{x} = \begin{pmatrix} \hat{x} \\ \hat{x}_{N+1} \end{pmatrix} \tag{6.28}$$

The uncertainty of the new state is a function of the current robot's uncertainty C_{rr} and the uncertainty in the measurement C_z and is defined as

$$C_{N+1N+1} = G_{x^r} C_{rr} (G_{x^r})^T + G_z C_z (G_z)^T \tag{6.29}$$

where G_{x^r} and G_z are the Jacobians of $g(\hat{x}_r, \hat{z}_{N+1})$. The correlations between the new feature, and the positions of the robot and mapped features are given by

$$C_{N+1i} = G_{x^r} C_{ri} \tag{6.30}$$

for $i = 0, 1, ..., N$ where C_{N+1i} is the correlation between the i^{th} feature and the new feature and C_{ri} is the correlation between the i^{th} feature and the robot.

6.3.6.1 Transformation function for a line feature

The position of the new feature $\hat{x}_{N+1}$ in the absolute frame of reference for a line feature is obtained by applying the transformation function $g(\hat{x}_r, \hat{z})$ for the observed line feature $\hat{z}$ relative to the robot position $\hat{x}_r$ and is defined as

$$\hat{x}_{N+1} = g(\hat{x}_r, \hat{z}) = \begin{pmatrix} \rho_z + x\cos(\theta + \phi_z) + y\sin(\theta + \phi_z) \\ \theta + \phi_z \end{pmatrix} \tag{6.31}$$

where $\hat{x}_r = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the robot position and $\hat{z} = \begin{bmatrix} \rho_z & \phi_z \end{bmatrix}^T$ is the line measurement relative to the robot position. The Jacobians for the line feature are defined as

$$G_{x^r} = \frac{\partial g(\hat{x}_r, \hat{z})}{\partial \hat{x}_r} = \begin{bmatrix} \cos(\theta + \phi) & \sin(\theta + \phi) & -x\sin(\theta + \phi) + y\cos(\theta + \phi) \\ 0 & 0 & 1 \end{bmatrix} \tag{6.32}$$

$$G_z = \frac{\partial g(\hat{x}_r, \hat{z})}{\partial \hat{z}} = \begin{bmatrix} 1 & -x\sin(\theta + \phi) + y\cos(\theta + \phi) \\ 0 & 1 \end{bmatrix} \tag{6.33}$$

6.3.6.2 Transformation function for a point feature

The position of the new feature x_{N+1} in the absolute frame of reference for a point feature is obtained by applying the transformation function $g(\hat{x}_r, \hat{z})$ for the observed line feature $\hat{z}$ relative to the robot position $\hat{x}_r$ and is defined as

$$\hat{x}_{N+1} = g(\hat{x}_r, \hat{z}) = \begin{pmatrix} x + x_p \cos\theta - y_p \sin\theta \\ y + x_p \sin\theta + y_p \cos\theta \end{pmatrix} \tag{6.34}$$

where $\hat{x}_r = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the robot position and $\hat{z} = \begin{bmatrix} x_p & y_p \end{bmatrix}^T$ is the point measurement relative to the robot position. The Jacobians for the point feature are defined as

$$G_{x^r} = \frac{\partial g(\hat{x}_r, \hat{z})}{\partial \hat{x}_r} = \begin{bmatrix} 1 & 0 & -x_p \sin\theta - y_p \cos\theta \\ 0 & 1 & x_p \cos\theta - y_p \sin\theta \end{bmatrix} \tag{6.35}$$

$$G_z = \frac{\partial g(\hat{x}_r, \hat{z})}{\partial \hat{z}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{6.36}$$

6.3.7 Feature Prediction for Matching Purposes

After the robot moves a distance and new features are extracted, the current features have to be referenced to the current robot position for matching purposes. In general, the feature $\hat{x}_{f_i}$ at position i in the state vector $\hat{x}$ is transformed with respect to the current robot position $\hat{x}_r$ using the function $h_{f_i}(\hat{x}_r, \hat{x}_{f_i})$. The covariance C_p of the predicted feature can be composed as

$$C_p = J_{f_i} C(t|t-1) J_{f_i}^T \tag{6.37}$$

$$J_{f_i} = \frac{\partial h_{f_i}(\hat{x}_r, \hat{x}_{f_i})}{\partial \hat{x}} = \begin{bmatrix} \frac{\partial h_{f_i}(\hat{x}_r, \hat{x}_{f_i})}{\partial \hat{x}_r} & \frac{\partial h_{f_i}(\hat{x}_r, \hat{x}_{f_i})}{\partial \hat{x}_{f_1}} & \cdots & \frac{\partial h_{f_i}(\hat{x}_r, \hat{x}_{f_i})}{\partial \hat{x}_{f_i}} & \cdots \end{bmatrix} \tag{6.38}$$

where J_{f_i} is the Jacobian of the prediction function with respect to the current state vector $\hat{x}(t|t-1)$ and C_p is the covariance of the feature. Since J_{f_i} is a function only of the i^{th}feature and the position of the robot, the rest of the elements in the matrix are zeros.

Once the i^{th} feature has been successfully matched using the data association algorithm from Section 6.3.8 the *EKF* is updated as described in Section 6.3.5 where the predicted position for the i^{th} feature is

$$h_i(t|t-1) = h_{f_i}(\hat{x}_r, \hat{x}_{f_i}) \tag{6.39}$$

6.3.7.1 Line Feature Prediction

The transformation function $h_{f_i}(\hat{x}_r, \hat{x}_{f_i})$ for the line feature with polar coordinates $\hat{x}_{f_i} = \begin{bmatrix} \rho & \phi \end{bmatrix}^T$ and robot current position $\hat{x}_r = [\ x \quad y \quad \theta\ ]^T$ is defined as

$$\hat{z}_l = h_{f_i}(\hat{x}_r, \hat{x}_{f_i}) \tag{6.40}$$

$$\hat{z}_l = \begin{pmatrix} \rho_z \\ \phi_z \end{pmatrix} = \begin{pmatrix} \rho - x\cos\phi - y\sin\phi \\ \phi - \theta \end{pmatrix} \tag{6.41}$$

where $\hat{z}_l = \begin{bmatrix} \rho_z & \phi_z \end{bmatrix}$ are the predicted parameters of the line for the current robot position. According to Eq. 6.37 the covariance C_p of the predicted line feature at position i in the state vector can be composed by the covariance $C(t\,|t\,)$ and the Jacobian J_{f_i} that is defined as

$$J_{f_i} = \begin{pmatrix} -\cos\phi & -\sin\phi & 0 & \cdots & 1 & x\sin\phi - y\cos\phi & \cdots \\ 0 & 0 & -1 & \cdots & 0 & 1 & \cdots \end{pmatrix} \tag{6.42}$$

for a line feature.

6.3.7.2 Point Feature Prediction

The transformation function $h_{f_i}(\hat{x}_r, \hat{x}_{f_i})$ for the point with Cartesian coordinates $\hat{x}_{f_i} = \begin{bmatrix} x_p & y_p \end{bmatrix}^T$ and robot current position $\hat{x}_r = [\ x \quad y \quad \theta\ ]^T$ is defined as

$$\hat{z}_p = h_{f_i}(\hat{x}_r, \hat{x}_{f_i}) \tag{6.43}$$

$$\hat{z}_p = \begin{pmatrix} x_z \\ y_z \end{pmatrix} = \begin{pmatrix} (x_p - x)\cos\theta + (y_p - y)\sin\theta \\ -(x_p - x)\sin\theta + (y_p - y)\cos\theta \end{pmatrix} \tag{6.44}$$

where $\hat{z}_p = \begin{bmatrix} x_z & y_z \end{bmatrix}$ are the predicted parameters of the line for the current robot position. According to Eq. 6.37 the covariance C_p of the predicted point feature at position i can be composed by the covariance $C(t\,|t\,)$ and the Jacobian J_{f_i} that is defined as

$$J_{f_i} = \begin{pmatrix} -\cos\theta & -\sin\theta & -x_l\sin\theta + y_l\cos\theta & \cdots & \cos\theta & \sin\theta & \cdots \\ 0 & 1 & -x_l\cos\theta - y_l\sin\theta & \cdots & -\sin\theta & \cos\theta & \cdots \end{pmatrix} \tag{6.45}$$

$$x_l = x_p - y \tag{6.46}$$

$$y_l = y_p - y \tag{6.47}$$

for a point feature.

6.3.8 Data Association

Features are matched using the current position of the robot as the frame of reference. A nearest neighbourhood approach is used to determine the best matches for the extracted features. A pairing $p_{ij} = (x_z^i, x_p^j)$ is found if it satisfies the Chi-squared $\chi^2_{\alpha,\upsilon}$ validation test

$$M_{ij} \leq \chi^2_{\alpha,\upsilon} \tag{6.48}$$

$$M_{ij} = (x_z^i - x_p^j) C_{ij} (x_z^i - x_p^j)^T \tag{6.49}$$

where M_{ij} is the Mahalanobis distance, C_{ij} is the innovation covariance matrix for the pairing. $\chi^2_{\alpha,\upsilon}$ is a value from a distribution where υ is the number of parameters of the feature and α is the probability level below which the pairing is discarded. Because of the presence of occlusion in indoor environments it is allowed that more than one observed line matches a map line.

For a line to line pairing the Mahalanobis distance (Eq. 6.49) is defined as

$$M_{ij} = \begin{bmatrix} (\rho_z - \rho_p) & (\phi_z - \phi_p) \end{bmatrix} \left[C_z^{\rho\phi} + C_p^{\rho\phi} \right]^{-1} \begin{bmatrix} (\rho_z - \rho_p) \\ (\phi_z - \phi_p) \end{bmatrix} \tag{6.50}$$

where the predicted line $x_p(\rho_p, \phi_p)$ has the covariance $C_p^{\rho\phi}$, and the observed line $x_z(\rho_z, \phi_z)$ has covariance $C_z^{\rho\phi}$.

For point features the Mahalanobis distance (Eq 6.49) is defined as

$$M_p = \begin{bmatrix} (x_z - x_p) & (y_z - y_p) \end{bmatrix} \left[C_z^{xy} + C_p^{xy} \right]^{-1} \begin{bmatrix} (x_z - x_p) \\ (y_z - y_p) \end{bmatrix} \tag{6.51}$$

where the predicted point $x_p(x_p, y_p)$ has the covariance C_p^{xy}, and the observed point $x_z(x_z, y_z)$ has covariance C_z^{xy}.

For the case of line features is necessary to verify that the pairs of matched lines overlap. The line model implemented maintains the extremes of the line. The extremes are parameterized by the signed distance along the line where the origin is the point on the line closest to the origin of the map frame. Following Bosse (2003) the overlap of two lines l_i and l_j with endpoints (s_i, e_i) and (s_i, e_i) is calculated as

$$overlap = \frac{\min(e_i, e_j) - \max(s_i, s_j)}{\max(e_i, e_j) - \min(s_i, s_j)} \tag{6.52}$$

where $s_i < e_i$ and $s_i < e_i$. *Overlap* is the percentage of overlapping between the pair of lines. If the *overlap* is larger than a threshold the pair of lines is matched. The threshold is user defined variable.

6.3.9 Efficiency and Data Association Management

Typically the data association is carried out by comparing every measured feature against every mapped feature. For the proposed model two separate matching processes are carried out, one to match line to line features and one to match point to point features. As Bosse (2003) noted in his implementation this process can be speeded up by sorting the line features on increasing distance from the origin and discarding those features whose Mahalanobis distance is above a user defined threshold.

For the sorted feature lists (one for points and for line features), when an unmatched measured feature F_z is further from the origin than a mapped feature F_M, no other subsequent mapped feature will match the measured feature. Once a measured feature has been processed, the next measured feature could start its matching process with the first mapped feature that was labelled as close to the previously processed feature. A pair of lines is labelled as close if their Mahalanobis distance is below a user defined threshold. The details of the implementation can be found in (Bosse, 2003).

6.3.10 Use of Structural Information

Typical indoor environments present many orthogonal features (corridors have parallel walls, rooms are rectangular). The use of this information improves the estimation of the map built. Although structural information is known *a priori* and not a proper observation of the environment it can be easily incorporated into the exploration system.

Structural information has been integrated in previous *EKF-SLAM* approaches as *virtual observations* between pairs of features. The covariance associated with the *virtual observations* is user defined (typically a large value). Large covariance values generate a slower convergence to the proper structural form.

In Newman et al.'s (2002) work at every *EKF* update step a number of random pairings are chosen and a constraint is applied after a validation test has been passed. In Losada's work (Rodriguez-Losada and Matia, 2003) for every new feature detected in the environment one structural constraint is applied. The constraint is applied to the most probable mapped feature. As pointed out by Losada it is conceptually more correct to apply constraints as new features are processed, to guarantee that only one constraint is applied for each new feature. In our work constraints are applied in the same way as in Losada's research. Three different constraint types are modelled: parallelism, perpendicularity and colinearity.

Every new added line feature is tested for compatibility ($\chi^2_{\alpha,\upsilon}$ Chi-Squared criterion

from Section 6.3.8) for the three constraint types against the previously mapped lines. The Mahalanobis distance for the constraints is summarized in Table 6.1.

The line features $\hat{x}_i = \begin{bmatrix} \rho_i & \phi_i \end{bmatrix}^T$ and $\hat{x}_j = \begin{bmatrix} \rho_i & \phi_i \end{bmatrix}^T$ are defined in the global reference frame. The constraints are the product of local relations such as corridors with long parallel walls and rectangular shaped offices; bearing this in mind and taking advantage of the fact that the features have already been sorted according to their distance to the origin (Section 6.2.9) the tests for parallelism and colinearity can be speeded up by testing only for close lines. In the case of the perpendicularity constraint this is not possible and the lines have to be tested against all of the previous lines. The distance to the origin ρ for pairs of perpendicular lines are likely to be outwith the threshold for close lines.

Parallelism	$M_{ij} = \begin{bmatrix} 0 & (\phi_i - \phi_i) \end{bmatrix} \left[C_i^{\rho\phi} + C_j^{\rho\phi}\right]^{-1} \begin{bmatrix} 0 \\ (\phi_i - \phi_j) \end{bmatrix}$
Perpendicularity	$M_{ij} = \begin{bmatrix} (\rho_i - \rho_j) & (\phi_i - \phi_j \pm \pi/2) \end{bmatrix} \left[C_i^{\rho\phi} + C_j^{\rho\phi}\right]^{-1} \begin{bmatrix} (\rho_i - \rho_j) \\ (\phi_i - \phi_j \pm \pi/2) \end{bmatrix}$
Colinearity	$M_{ij} = \begin{bmatrix} (\rho_i - \rho_j) & (\phi_i - \phi_j) \end{bmatrix} \left[C_i^{\rho\phi} + C_j^{\rho\phi}\right]^{-1} \begin{bmatrix} (\rho_i - \rho_j) \\ (\phi_i - \phi_j) \end{bmatrix}$

Table 6.1: Mahalanobis distance for structural constraints.

A nearest neighbour approach is used to determine the best pairing for each of the new lines. Typically for a new line more than one constraint can be applied, but the application of all of them is impractical. In general, the *virtual observation* function $H_{ij} = \begin{bmatrix} \rho_{virtual} & \phi_{virtual} \end{bmatrix}^T$ for the pairing $p_{ij} = (x_i, x_j)$ where i is the recently added line and j is the previously mapped line is defined according to Table 6.2.

Parallelism	$H_{ij} = \begin{pmatrix} 0 \\ \phi_i - \phi_j \end{pmatrix}$
Perpendicularity	$H_{ij} = \begin{pmatrix} \rho_i - \rho_j \\ \phi_i - \phi_j \end{pmatrix}$
Colinearity	$H_{ij} = \begin{pmatrix} \rho_i - \rho_j \\ \phi_i - \phi_j \end{pmatrix}$

Table 6.2: Virtual observation function for structural constraints.

The Jacobian J_{ij} of the *virtual observation* is a function only of the states of i and j and the rest of the elements of the Jacobian are zero values

$$J_{ij} = \frac{\partial H_{ij}}{\partial_x} = \begin{bmatrix} \cdots & \frac{\partial H_{ij}}{\partial x_i} & \cdots & \frac{\partial H_{ij}}{\partial x_j} & \cdots \end{bmatrix} \tag{6.53}$$

Table 6.3 presents the Jacobian for the structural constraints.

Parallelism	$J_{ij} = \begin{bmatrix} \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ \cdots & 0 & 1 & \cdots & 0 & -1 & \cdots \end{bmatrix}$
Perpendicularity	$J_{ij} = \begin{bmatrix} \cdots & 1 & 0 & \cdots & -1 & 0 & \cdots \\ \cdots & 0 & 1 & \cdots & 0 & -1 & \cdots \end{bmatrix}$
Colinearity	$J_{ij} = \begin{bmatrix} \cdots & 1 & 0 & \cdots & -1 & 0 & \cdots \\ \cdots & 0 & 1 & \cdots & 0 & -1 & \cdots \end{bmatrix}$

Table 6.3: Jacobian of the virtual observation functions from Table 6.2 for structural constraints.

6.4 Summary

This chapter has presented the Map Building Module. The module builds a feature based representation of the environment. An *EKF* approach is used to update the robot and feature locations. The *EKF* has two stages: prediction and update. At the prediction stage the robot location is predicted based on a kinematic model and odometric information. At the update stage re-observed features are used to update the state. The *EKF* state is typically composed of the robot and feature locations. This representation has been augmented to allow the extraction of information from multiple locations. The state is then composed by the last *m sensing* positions of the robots and the features. A *sensing* position is a position where the robot performs sensor measurements. The initial state contains only the initial position of the robot. The robot moves through the environment while performing sensor measurements. Line and point features are extracted from sensor measurements. An efficient *RANSAC* (Random Sample Consensus) approach is used to extract the feature parameters.

The data association process to determine the pairings between the extracted features and the mapped features is solved using a Nearest Neighbour approach. Matched features are used to update the estimates of the robot and feature locations. Unmatched features are added to the feature map by augmenting the state vector. Additional map management processes are carried out. Such processes include the fusion of two objects that are hypothesized to be the same object. This is common for line features that are initially represented as short segments because they are partially occluded. The state vector size of the *EKF* decreases or increases as features are added or deleted

from the map. Models to integrate structural information by means of *virtual observations* were defined.

Chapter 7

The Simulator and the Validation of the Simulated Robots

7.1 Introduction

This chapter presents the simulator that we used to test our BERODE (BEhavioural ROle DEcentralized) architecture and the simulation models for the robot's sensors, odometry and communication devices. Our simulation models are reasonable and conservative approximations to the data obtained in our experiments from the hardware sensor devices.

Section 7.2 introduces the simulator and the simulated office-like environments used throughout the rest of this thesis. In the simulations we assume unlimited bandwidth for the communication models. The delays caused by retransmissions in the *MANET* (Mobile *ad hoc* network) are modelled. Interference causes errors and data retransmissions causing additional delays in the diffusion of information. In the current implementation the effect of interference is modelled by delaying the messages a random time with a certain probability.

Most if not all of the current communication technologies are based on *RF* and *LOS* technologies. For this reason, we propose the use of the *RF* and *LOS* communication models to assess BERODE.

Section 7.3 presents the simulation model for *RF* communication. This model is based on Rappaport's model (Rappaport, 2001) where the strength of a signal is calculated based on the *path loss* in decibels (dB). The *path loss* is the amount of power lost by a signal due to the transmission distance, the number of obstacles in the direct path of the signal and the properties of these obstacles (e.g. material, density). The

RF model has been validated by comparing the signal strength maps generated by the simulation model against measured data in an office environment using a Bluetooth device. Our experimental results suggest that the simulation model is accurate and conservative. In our testing environment in 85% of the sampled positions the difference in signal strength between the simulated and the measured signals was below 5%. The model is conservative because it underestimates the signal strength by 2% in average.

Section 7.4 presents the *LOS* simulation model. In this model any obstacle in the direct path of the signal blocks the entire signal. This is a conservative simplifying assumption. We have validated this model by comparing the communication coverage obtained for the simulation model against the coverage obtained from an infrared communication device which mainly works when there is a line of sight between transmitter and receiver. The *LOS* simulation is conservative because the area in which communication can be reliably achieved (above 80%) is smaller for the simulation model compared to that of the real infrared communication coverage.

To have meaningful simulations it is important to model relevant aspects of the type of performance achieved by a real robot. In Section 7.5 we present the simulation model for robot motion. Our motion model is based on parameters obtained from experiments with a Koala robot. Real robots tend to veer towards one side due to the uneven wheel traction. The simulated robots incorporate this veering phenomenon. The accuracy in the odometry measurements for the simulated robots is based on the experimental measurements obtained with the Koala robot.

One of the goals of this work is to validate through simulation the suitability of the use of *low cost* robots to build maps of the environment. In Section 3.6 (page 45) we discussed the suitability of *low cost* sensors for map building purposes. We concluded that sonar and infrared sensors were the most suitable sensors because of their cost and features. Section 7.6 presents the simulation model for the sonar and infrared sensors. Our simulation models are conservative with respect to the uncertainty and obstacle detection reliability observed in our tests with the sensor devices.

Section 7.7 presents the *low cost* sensing platform that was designed and built to validate the map building module described in the previous chapter. This platform is based on sonar and infrared sensors. Section 7.8 discusses the implementation of the map building algorithm for sonar and infrared sensors. Section 7.9 presents two investigations to validate the proposed map building module. In the first investigation the real experimental robot builds a map of a corridor. In the second investigation a simulated environment that contains an exploration loop is used to highlight the advantage

of combining sonar and infrared sensors over the use sonar alone information. Section 7.10 presents the conclusions from the experiments with the hardware devices.

7.2 The Simulator and the Simulated Environments

The experiments for multiple robots were conducted only in simulation using the Webots simulator (Michel, 2004). The simulator has two types of controllers: robot and supervisor. The robot controller is the controller implemented in the simulated robots.

The supervisor controller is used as the communication manager. The supervisor controller handles the inter-robot communication and records statistics about the status of the network. The supervisor has knowledge of the environmental map. The supervisor uses this map to simulate the behaviour of the signals in the *LOS* and *RF* communication models. The communication model for the *MANET* (Mobile *ad hoc* network) that the robots form is implemented as follows:

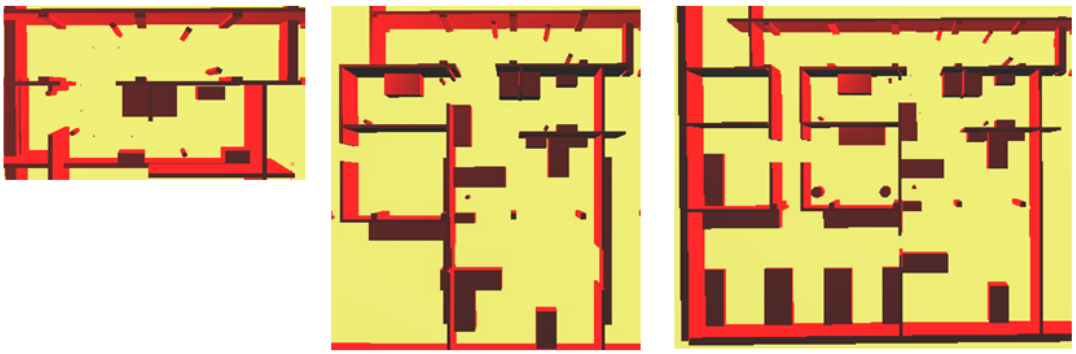
- The loss of information for a pair of robots is simulated by randomly dropping communication messages with a certain probability (5% in our implementation). The simulated robots retransmit lost information (Section 5.10, page 130).
- The communication bandwidth for the robot connections is large enough to cope with the exchange of information regardless of the robot positions.
- Delays in the process of transmission and reception are modelled. For a direct connection the reception of a message is delayed $t_{transmission}$ seconds regardless of the distance between the transmitter and receiver.
- The delay in the reception of a message for robots that are at *k-hops* of distance in the *MANET* is delayed $t_{transmission}$**k-hops* seconds.
- If two robots go out of range and their connection is part of the MST control network the information that flows through the connection is held until the connection is re-established.

The assumption of unlimited bandwidth is reasonable because the available bandwidth for current communication technologies is much larger than the bandwidth required by the robots. The bandwidth for Bluetooth ranges from 120Kb/s–723 Kb/s, Ethernet ranges from 1Mbps-4Mbps (Insider, 2005). Moreover inexpensive transceivers have a bandwidth of 40 Kb/s (Radiometrix, 2004). For instance, in

McLurkins' research he used an infrared communication system with a bandwidth of 60b/s to disperse a large group of robots throughout an indoor environment (McLurkin and Smiths, 2004). He used fifty six robots in his experiments; however his robots did not share mapping information. In BERODE mapping information is shared, but this doesn't pose a problem for the available bandwidth of current technologies because most information is only transmitted locally.

In this research it is important to model the communication delays because the robots use the received information to coordinate their actions. Interference causes errors and data retransmissions thus generating delays. We argue that the effect of the loss of communication and interference in the performance of the robot network can be appreciated by implementing larger delays in the reception of information. In the current implementation the effect of interference is modelled by delaying the messages a random time with a certain probability.

The supervisor records statistics about the status of the network and estimates the required bandwidth for each robot. The bandwidth is estimated (in Bps) from the number of messages that the robots received in a time span using the proposed application level protocol (Section 5.10.1, page 132). Appendix C presents the details of the bandwidth estimation. The controllers were implemented using Java.



a) Small (10m*5.5m) b) Medium (12m*16m) c) Large (20m*16 m)

Figure 7.1: Office like environments used in the simulations.

Figure 6.1 presents the three environments used in the simulations. In the simulations the simulated robots are grouped in the upper left area. Each simulated robot has a direct connection with at least one simulated robot. These environments are based on the layout of sections of the *IPAB* (Institute of Perception Action and Behaviour). These sections were selected because they have connected rooms. The struc-

ture formed by these connected rooms poses a difficult scenario for robots that have to maintain a communication network. The non-Explorer robots generate short terms plans to move to the locations where the communication quality is best. In connected rooms a robot could potentially get trapped due to the presence of local maxima for the communication quality. In an open area with only a few scattered obstacles these local maxima are less likely to be present therefore it is easier to maintain the communication network while exploring these areas.

The following sections present the *RF* and *LOS* communication models used in our simulations. These models have been validated by comparing the simulation models against measured data. The simulation models have been shown to be reasonable and conservative approximations to the measured data from the real devices in typical scenarios.

7.3 Simulation Model for Radio Frequency

To have meaningful results from the experimental simulations with Radio Frequency (*RF*) communication is necessary to take into account the propagation characteristics of real *RF* signals. *RF s*ignal propagation is difficult to predict especially when there is no line of sight path between the transmitter and the receiver.

In the *RF* implementation of BERODE the simulated robots are assumed to be able to accurately measure the signal strength of the signals. As explained in the previous section the supervisor controller is used as the communication manager. This controller has knowledge of the environmental map and the robots' positions. Using this information and an *RF* signal propagation model the supervisor controller determines the strength of the robots' signals. The supervisor sends this information to the simulated robots which use this information to keep the network connected.

The strength of a signal is defined in terms of the *path loss* that occurs between the transmitter and receiver. The *path loss* is the amount of power that a signal loses between the transmitter and receiver measured in decibels (dB). Previous research (Keenan and Motley, 1990; Seidel and Rappaport, 1992b) has demonstrated that a simple and surprisingly accurate way to predict *path loss* inside a building is to count partitions for a line drawn between transmitter and receiver. An experimentally derived attenuation factor may then be applied to each partition which intersects the line. Our simulation model for *RF* communication is based on this simple prediction model. The following sections describe the simulation model and present the experiments carried

out to obtain the attenuation factor for different types of obstacles present in the IPAB institute. Section 7.3.3 presents the validation of the *RF* simulation model against real measurements. We used a Belkin Bluetooth device to validate our simulation model. We found that the simulation model was an accurate and conservative approximation to the real measured data.

7.3.1 The Signal Strength Model for Radio Frequency

We used Rappaport's model to determine the strength of the *RF* signals (Rappaport, 2001). In this model the strength of a signal is defined in terms of the *path loss* that occurs between the transmitter and receiver. The *path loss* is a measure of the attenuation of the signal and is measured in decibels. The *path loss* between a pair of robots depends on the distance between the robots, the number of obstacles between them and the properties of the obstacles (material and density). The *path loss* model is defined as

$$PL(d) = FSL(d) + \sum_i AGS_i \tag{7.1}$$

where

$$FSL(d) = PL(d_0) + 20\log\left(\frac{d}{d_0}\right) + \alpha d \tag{7.2}$$

where $PL(d_0)$ is the *path loss* in dB at a small distance d_0 from the transmitter, d is the distance between the transmitter and the receiver, $PL(d)$ is the total *path loss* for d, $FSL(d)$ is the *path loss* in free space, α is a constant that depends on type of environment (e.g. office building, outdoors) and AGS_i is the attenuation for the i^{th} obstacle between the transmitter and receiver. This attenuation is the amount of power that a signal loses (in dB) by passing through the obstacle and depends on the material and density of the obstacle. The supervisor controller uses a grid map of the environment where the grid cells can represent materials with differing *RF* properties. *AGS* is then the attenuation for an obstacle cell of a certain material. In our simulations we use experimentally derived attenuation factors for several materials (in dB/m) and compare them with the values provided by manufacturers of wireless cards at 2.45GHz (MaxStream, 2003). We conducted experiments to check the validity of attenuation factors for our *RF* Belkin Bluetooth device. In these experiments we used a Belkin Bluetooth class I device which has a maximum free space communication range of 100m according to the manufacturer. In order to simulate this device the parameters from Eq 7.2 were set to: d_0=1m, $PL(d_0)$=30dB and $\alpha = 0.2$. These parameters were

chosen by comparing the path loss predicted at various distances (in free space) by the model with the specification for our Bluetooth device.

The model from Eq. 7.2 does not take into account the effects of multi path propagation. Multi path propagation occurs because some obstacles reflect and scatter the transmitted signal in ways that are difficult to predict. When there is *LOS* between transmitter and receiver the signal along the *LOS* dominates the effects of multi path propagation. In this work the multi path effects are modelled by adding Gaussian Noise (with mean zero and standard deviation σ =5dB) to Eq. 7.2 when there is no *LOS* between transmitter and receiver. Although this model does not predict signal strength fluctuations accurately the general effect is similar to that of the multi path propagation.

In our research we compare various communication ranges. We refer to the communication range as the distance d_{range} that a transmitted signal disperses in free space. The signal power (in dB) necessary for a communication range is calculated using Eq. 7.2. In the simulations is assumed that a communication link exists when $FSL(d_{range}) - PL(d) > 0$.

As previously mentioned it is assumed that the robots can measure the signal strength. This value is typically available in wireless *RF* technologies and is referred as the *Received Strength Signal Level* (*RSSL*). In the simulations the signal strength is calculated as

$$SSL(d) = \frac{FSL(d_{range}) - PL(d)}{PL(d)} * 100 \tag{7.3}$$

which is the percentage of the power signal (in dB) available at the distance d.

7.3.2 Calculating the attenuation factor for the obstacles

The manufacturers of wireless devices provide tables of the attenuation factors of several materials at a frequency of 2.45GHz (MaxStream, 2003). We conducted experiments to check the validity of the attenuation factors for our *RF* Belkin Bluetooth device. This device has a maximum output power of 20dB and a receiving sensitivity of -70dB. The maximum transmission distance for this device is 100m for free space communication.

We experimentally obtained the attenuation factor for the following obstacles: walls, load bearing columns, metallic file cabinets, wood doors and desks. We used a typical desk which was accompanied by an assortment of books, computer equipment and a wood drawer. The legs of the desk are made of steel. These types of obstacles

were the most common static obstacles that we found in our IPAB institute.

To measure the signal strength we used a Belkin Bluetooth device and an Agilent 437B *RF* Power Meter. The Belkin Bluetooth device was set in the connectivity mode in which the device broadcasts information to try to connect to another device. The power meter was used to measure the strength of the signals sent by the Bluetooth device. The attenuation in signal strength due to the obstacle is determined by measuring the difference in signal strength when there is an obstacle between the transmitter and the receiver compared to the signal strength in free space (no obstacle). For the desk obstacle the transmitter and the receiver were located at right angles to the long side of the desk where there was most visible blockage in the direct line of sight.

Figure 7.2 shows the experimental procedure used to determine the attenuation factor for the obstacles. The transmitter (Bluetooth device) and the receiver (*RF* power meter) were located perpendicular to the obstacle. The devices were placed at a height of 0.2m because that is the height at which they are mounted on the real Koala robot used in our investigations. The transmitter was initially located at a distance d_t=0.25m to the obstacle while the receiver was located at a distance d_r=0.25m to the obstacle. The transmitter was moved in steps of 0.25m until the distance to the obstacle was 1m. This procedure was repeated for distances from the receiver to the obstacle ranging from 0.25m to 1m in steps of 0.25m. Ten measurements were recorded for each transmitter and receiver position. For each position the *RF* power meter measured the signal strength of the broadcasted information during a sampling period of thirty seconds.

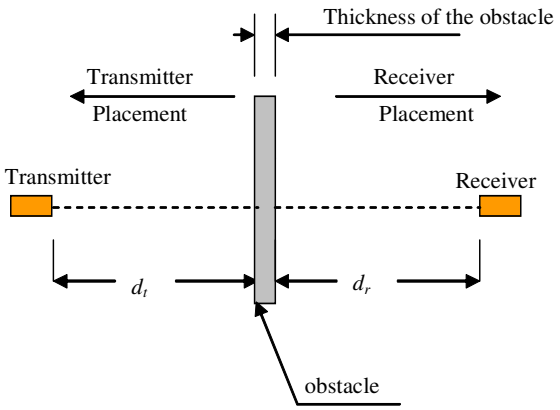


Figure 7.2: Procedure used to calculate the attenuation factor for an obstacle. The transmitter and the receiver were located perpendicular to the obstacle. The transmitter was located at a distance d_t to the obstacle while the receiver was located at a distance d_r to the obstacle. The transmitter and the receiver were moved away from the obstacle.

Table 7.1 shows the experimental results of the attenuation of the *RF* signal for the different types of obstacles and their thickness. The attenuation factor is the ratio between the measured attenuation and the thickness of the obstacle measured in dB/m.

Obstacle type	**Thickness (m)**	**Attenuation (dB)**	**attenuation/thickness (dB/m)**
Brick wall	0.18m	1.101±0.23	5.61dB/m
Load Column	0.3m	2.094±0.27	6.98dB/m
Wood door	0.1m	0.178±0.05	1.78dB/m
Wood desk	1.0 m	2.12±0.17	2.12dB/m
Metallic File cabinet	0.5m	5.89±0.38	11.98dB/m

Table 7.1: Attenuation for different obstacle types obtained experimentally following the procedure described in Figure 7.2.

Table 7.2 shows a table of the attenuation for some materials provided by the manufacturers of wireless devices (MaxStream, 2003). Although the types of obstacles from the two tables are not the same it is observed that the results from our measurements are reasonably similar for materials that are alike. For instance, the brick wall from our office is made of brick and concrete (white DryWall finish), the attenuation value obtained in our tests is a value in the range of the concrete and brick values provided by the manufacturers (slightly less than the average of both materials).

In the following section we show experiments that validate the proposed simulation model which uses the experimentally derived attenuation factors obtained in this section.

Obstacle type	**Thickness**	**Attenuation (dB)**	**attenuation/thickness (dB/m)**
Lumber	0.76m	2.8dB	3.68dB/m
Concrete	1.02m	12dB	11.76dB/m
Concrete	2.03m	23dB	11.33dB/m
Brick	2.67m	7dB	2.67dB/m
Masonry Block	2.03m	12dB	5.91dB/m

Table 7.2: Attenuation for different obstacle types provided by MaxStream (2003).

7.3.3 Validation of the RF simulation model

In order to validate the *RF* simulation model we compared this model against a measured *RF* signal obtained from our Bluetooth device. We used a section of the IPAB

institute to validate the measurements. Figure 7.3 shows the layout of the IPAB institute where the origin of the coordinate system is in the bottom left part of the map. The dimensions of the rooms were obtained from the builders' plans. The layout shows the furniture that is fixed. Furniture that is moved around (e.g. plastic chairs) is not shown in this layout. The walls are constructed of cinder block and concrete headers. Floors are concrete covered with durable carpet, and office doors are fabricated from wood. The rooms contain standard office desks made of wood with standard steel legs. Most desks are accompanied by a typical assortment of books, computer equipment, knick knacks, and a limited number of lamps, and fans. The rooms also have metallic file cabinets and some of them have load bearing columns.

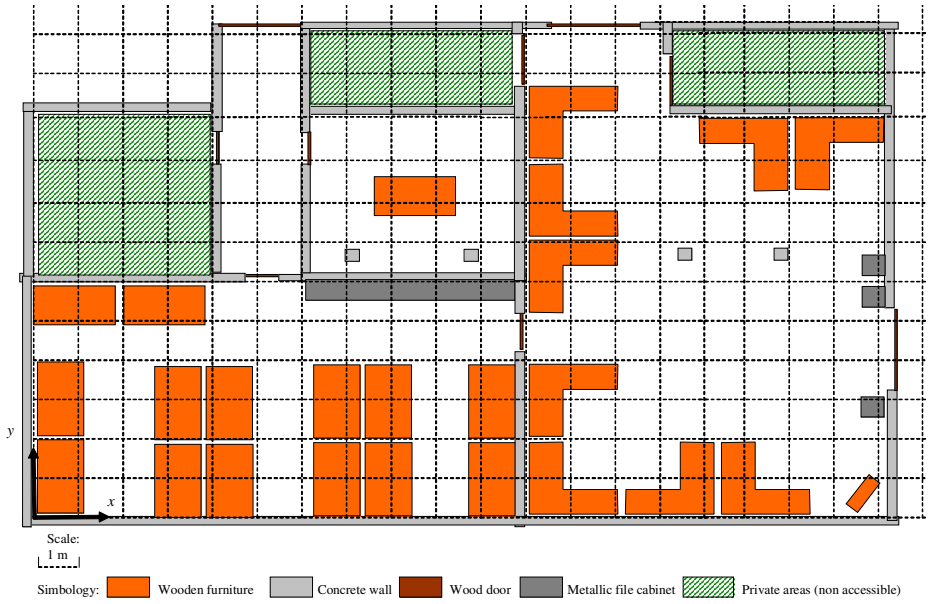


Figure 7.3: Layout of the office environment used in the validation of the *RF* and *LOS* simulation models.

To measure the signal strength we used a Belkin Bluetooth device and an Agilent 437B *RF* Power Meter. The Belkin Bluetooth device was set in the connectivity mode in which the device broadcasts information to try to connect to another device. The power meter was used to measure the strength of the signals received from the Bluetooth device. The Bluetooth device is located in a fixed position while the *RF* power meter is moved throughout the office environment in steps of 1m in the x and y directions to obtain the sample measurements. Once the *RF* power meter is located at a measurement position it measures the signal strength of the broadcasted information during a sampling period of thirty seconds.

Multipath propagation occurs when radio frequency (*RF*) signals take different paths from a source to a destination. A part of the signal goes to the destination while another part bounces off an obstruction, then goes on to the destination. When the reflected signals are combined at the receiver, the signal strength is high. To alleviate the multi-path and propagation effects in the measurement process, the receiver is rotated 90 degrees for each sample position, until one revolution is complete. The number of samples for each position is twenty (five for each orientation). The receiver is then moved to another location within the measurement area.

To compare the measured *RF* signal with the simulated signals we produced a simulation model of the environment from Figure 7.3. In this model we used the experimental measurements from Section 7.3.1 of the loss of signal strength for a signal that traverses an obstacle. The simulation model was described in Section 7.3.2 where Eq. 7.1 is used to quantify the amount of lost power between the transmitter and receiver. As previously explained the Bluetooth device used in our experiment has a maximum output power of 20dB and a receiving sensitivity of -70dB. The measured signal strength is then a value in the range of 0 to -90 dB being -90dB when the robot is at the maximum distance where communication is possible. This distance is 100m for our Bluetooth device when there is a clear line of sight between transmitter and receiver.

In our tests we placed the transmitter in two different positions and obtained the sample measurements for each position. Figure 7.4 shows the signal strength map generated by the simulation *RF* model for the transmitter located at position (2, 0). The effects of interference and multi-path propagation are incorporated into the simulation model by adding random noise to the simulated signal strength measurements. Without the addition of noise the signal strength monotonically decreases with distance, although it is irregular due to the irregularity of the radio-translucent obstacles. As seen in Figure 7.4 when noise is added some slight non monotonic discontinuities are visible.

Figure 7.5 shows the signal strength map for the measured *RF* signal. For this map there is not a monotonic decrease in general terms as in the case of the simulated *RF* map. This is due to the effects of multi-path propagation and interference of the real *RF* signals which create discontinuities in the signal strength decrease. For instance in the right middle side of the large room (right side of the environment) the signal strength is in the range of -5dB to -10dB however locations that are more distant to the transmitter have better signal strength (bigger values).

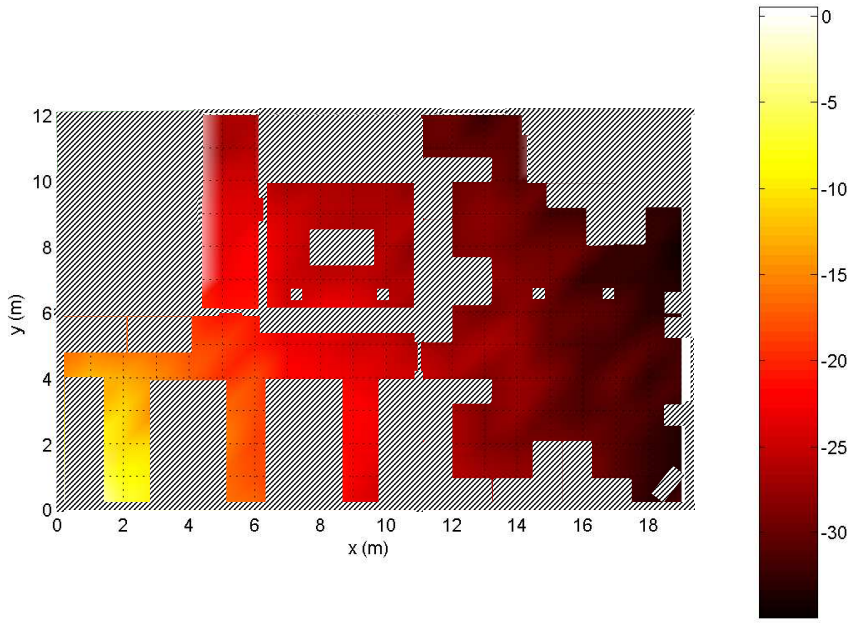


Figure 7.4: Signal strength map for a simulated *RF* signal generated from position (2,0) using the model from Eq.7.1. The maximum signal strength value is 0dB. The obstacles of the simulated office environment are shown hashed.

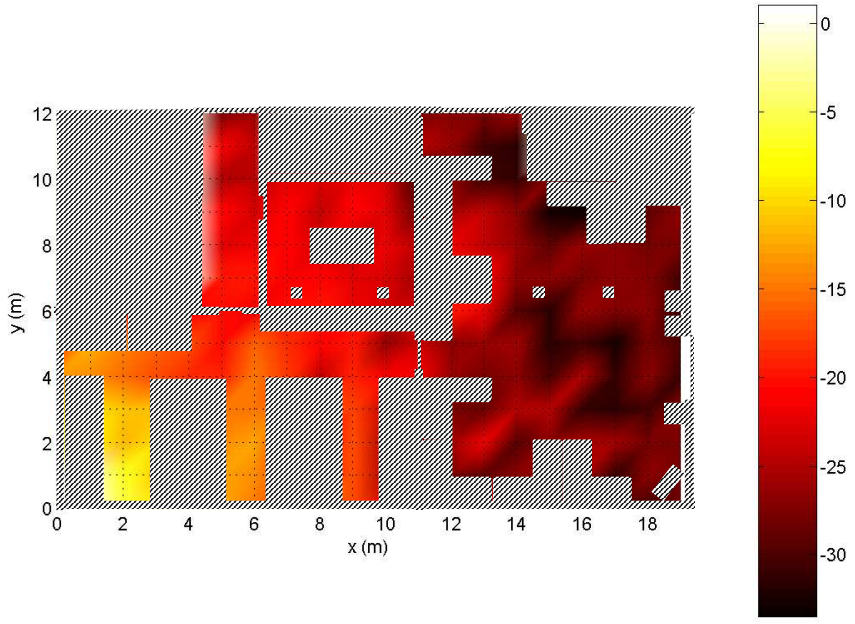


Figure 7.5: Signal strength map for a measured *RF* signal generated from position (2,0). The maximum signal strength value is 0dB. The obstacles of the office environment are shown hashed. The signal strength was measured.

Figure 7.6 and Figure 7.7 show the percentage of signal strength difference (simulated minus measured) between the simulated and the measured signals for the transmitter at positions (2,0) and (11,4) respectively. From these figures it is observed that the simulation model is conservative because for most of the sampled positions the model underestimates the signal strength (negative percentages). For the data from Figure 7.6 the percentage of difference is in the range of [3%,-6%] with the simulation on average $2.21 \pm 1.51\%$ less than the measured signal. For the data from Figure 7.7 the percentage of difference is in the range of [2%,-8%] with the simulation on average $2.06 \pm 0.96\%$ less than the measured signal. In general we can observe that the percentage of difference is larger when the line of sight between the transmitter and the receiver is blocked by a larger total depth of obstacles. This is expected because in the simulation *RF* model we use an average value for each type of obstacle obtained from our tests with different types of obstacles. Moreover, the effect of multi path is modelled by adding Gaussian Noise (with mean zero) when there is no line of sight between transmitter and receiver whereas in reality reflected signals behave in complicated ways which sometimes result in communication *hot spots* that are difficult to predict even for more sophisticated communication models (Molkdar, 1991). Although our simulations do not specifically model *hot spots*, we model the effects of multi-path and interference by adding Gaussian random noise which does sometimes produce *hot spots*.

From Figures 7.6 and 7.7 it is observed that for most of the tested positions (85.4%) the percentage of difference is smaller than 5%. For figure 7.6 in 80.2% of the measured positions the difference is smaller than 5% while for figure 7.7 the percentage is 90.6%.

In this section we have validated the *RF* simulation model. We have shown that in the tested environment the model was accurate and conservative; accurate because in 85.4% of the tested positions the difference between the simulated and the measured signals was below 5%; conservative because it underestimates the signal strength by 2% on average. This suggests that the simulation model is an accurate and conservative approximation to the real measured data.

7.4 Simulation Model for LOS Communication

The *LOS* communication model assumes an infinite communication range where any obstacle in the direct path between the transmitter and the receiver blocks the signal.

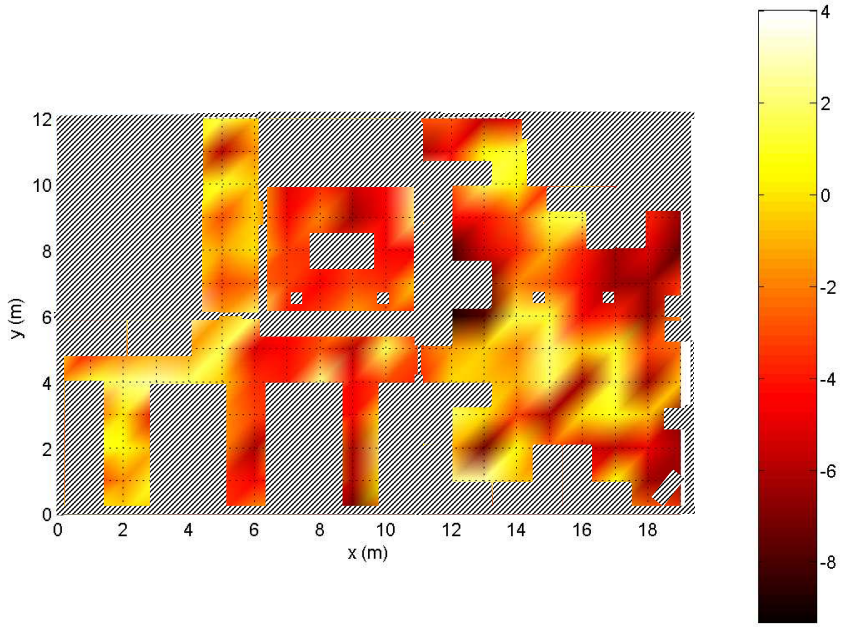


Figure 7.6: Percentage of signal strength difference (simulated minus measured) between the simulated and the measured *RF* signal for a transmitter located at position (2,0). Note that the simulation model is conservative because for most positions the model underestimates the signal strength (negative percentage).

In practice this means that communication range exceeds available sight lines, which in the simulated office environments are quite short (Figure 7.1). The supervisor controller uses a grid map representation of the environment. The grid cells of this map can represent materials with different *RF* properties but opaque to infrared. If the signal between a pair of robots traverses an occupied cell then there is no communication between them, otherwise there is a communication link. If there is a communication link it is assumed that the bandwidth is large enough to cope with the exchange of information regardless of the link positions.

7.4.1 Validation of the LOS model

The implementation of the *LOS* model of BERODE was proposed because technologies such as *low* cost infrared communication devices work reliably when there is an uninterrupted line-of-sight (*LOS*) path between the transmitter and receiver. Infrared communication has as advantages low power consumption and low cost compared with other communication technologies (Kahn, 1997). For these reasons, in previous

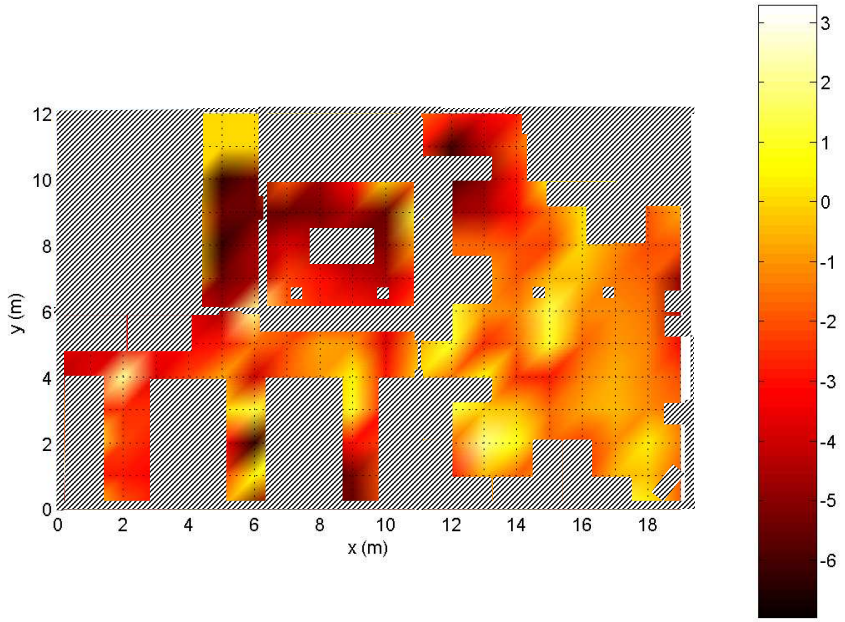


Figure 7.7: Percentage of signal strength difference (simulated minus measured) between the simulated and the measured *RF* signal for a transmitter located at position (11,4). Note that the simulation model is conservative because for most positions the model underestimates the signal strength (negative percentage).

research infrared communication has been used to achieve swarming behaviours by groups of robots (McLurkin and Smiths, 2004).

Infrared communication devices are classified according to their design as *LOS* or diffuse devices. *LOS* devices rely upon the existence of an uninterrupted line of sight (*LOS*) path between the transmitter and receiver, while diffuse devices generally rely upon reflection of the light from the ceiling or some other diffusely reflecting surface. The required transmission power for diffuse devices is higher compared to that of *LOS* devices. Diffuse devices are designed to increases link robustness and ease of use, allowing the link to operate even when barriers, such as people or cubicle partitions, stand between the transmitter and receiver. *LOS* devices are designed to maximize power efficiency by having a narrow field of view, typically in the range of 15°- 60°. The coverage of the entire communication circumference is typically achieved by using a ring of infrared transmitters. Alternatively one can employ a transmissive diffuser, such as a thin plate of translucent plastic to disperse the signal from a *LOS* infrared device to achieve the 360° coverage with a single transceiver. Due to the dispersion of the signal the communication range is substantially reduced. The amount of reduction

depends on the transceiver's field of view.

To validate the suitability of the *LOS* model for infrared communication devices we compared the communication coverage from our simulation model against the coverage obtained from an infrared communication device. In the experimental tests we used a Palm m505 handheld device as the transmitter and a Lego Brain Brick as the receiver (Figure 7.8). We incorporated an OmniRemote infrared module to the handheld device to increase the communication range. This device is inexpensive (£20) and increases by 400% the communication range of the handheld built in IR transceiver without requiring additional batteries. The OmniRemote module comes with software that enables the use of the handheld device as a universal control for TVs, VCRs, Lego robots among other devices. The OmniRemote module has a field of view of 30°. According to the manufacturer tests the OmniRemote is capable of controlling the Lego brick up to a distance of 50 feet (15.24m) when the transmitter is pointing in the direction of the Lego Brain Brick. Our preliminary tests show that the transmitter (Palm with the OmniRemote attached) was able to communicate with the Lego Brain Brick up to a distance of 18.5m. In this test the receiver (Lego Brain Brick) and the transmitter were placed on the floor at an initial distance of 5m. The transmitter sent a command that produced a beep sound by the receiver. This command was sent three times and the communication was considered to be successful if the beep sound was heard at least once. The transmitter was then moved away in steps of 0.5m until communication the beep sound was no longer heard. It is worth mentioning that the transmission range for the Lego brick is much smaller than 18.5m because the transmitter of the Lego brick is much less powerful.

The *LOS* simulation model has a 360° communication coverage. To validate this model using a single transmitter and receiver we used *LOS* infrared devices with conical reflective dispersers. The 360° can be achieved by placing a reflective surface on top of the transmitter. This surface disperses the signal having as result the 360° coverage in the horizontal plane having as downside a reduction in the communication range due to the attenuation of the signal. We used an aluminium conic surface to reflect the infrared signals. Figure 7.9 illustrates how the transmitter and receiver are positioned to face straight up and all the incoming and outgoing infrared signals are reflected by an aluminium cone.

The environment used to validate the *LOS* model was the bottom left room of the environment from Figure 7.3. To compare the measured *LOS* signal with the simulated signals we produced a simulation model of the environment. The transmitter was

a) Omniremote module

b) Omniremote module attached to the handheld Palm III device

a) Lego Brain Brick

Figure 7.8: Photographs of a) the Omniremote module, b) the Palm III handheld device with the Omniremote module attached and c) the Lego Brain Brick used in the infrared communication tests.

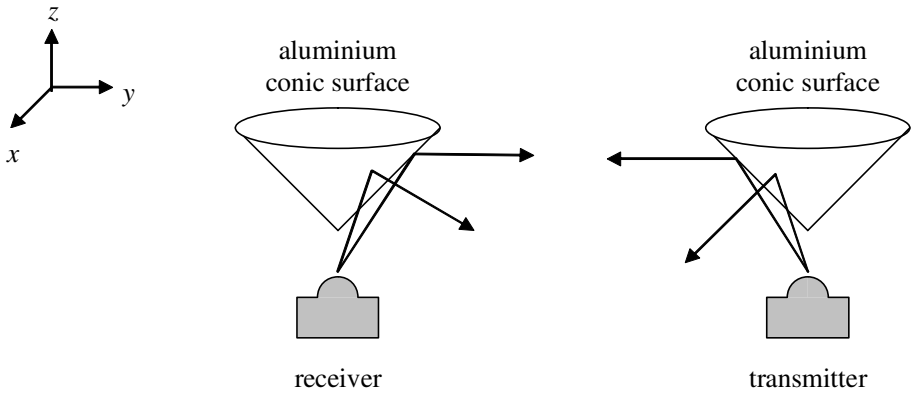


Figure 7.9: The transmitter and receiver are positioned to face straight up and all the incoming and outgoing infrared signals are reflected by an aluminium cone. The result is a 360° coverage in the horizontal plane.

located in a fixed position while the receiver was moved throughout the environment in steps of 0.5m in the x and y directions to obtain the communication samples. A communication sample consisted of the transmitter sending a command that produced a beep sound by the receiver. Communication was considered as successful if the receiver emitted the beep sound. Five sample measurements where taken at each location. As explained in Section 7.2 the loss of communication is simulated by randomly dropping communication messages. A random 5% of the messages are dropped in the simulations.

Figures 7.10 and 7.11 show the comparison between the simulated *LOS* commu-

nication model and the measured infrared communication for a transmitter located at positions (5.5, 3) and (3.5, 4.5) (measured in meters). The figures show the outline of the polygon which encloses all connected test positions with at least 80% of communication success rate (i.e. outliers are omitted). In this experimental comparison the messages were broadcasted, there was no protocol to retransmit information when it was lost. Inside the polygon area the communication is reliable. From the figures it is observed that this area is smaller for the simulation model compared to that of the measured infrared communication. The coverage of the real infrared device is much larger compared to the simulated *LOS* model because the real infrared signals bounce from the obstacles; however the reception of information is less reliable when there is no line of sight between transmitter and receiver because the signals are attenuated after multiple reflections and the receiver fails to detect these signals.

The *LOS* simulation model assumes infinite communication range when there is an uninterrupted *LOS* path between transmitter and receiver. In office-like environments the sight lines are short because the rooms are full of obstacles (e.g. desks). As observed in our tests from this section the simulation model is conservative because the communication coverage is smaller than the measured area. Moreover, in the implementation of BERODE for the *LOS* model the robots are constrained to remain in close locations to reduce the likelihood of losing communication due to obstacle obstruction. In our implementation (Section 8.6, page 239) a robot is constrained to remain at a distance smaller than 3.5m from the robot connections that it has to keep within communication range. Therefore although the range is assumed as infinite the robots always remain in close positions (below 3.5m).

In this section we have validated the *LOS* model. We conclude that the simulation model is a conservative approximation to the real infrared communication coverage because the area in which communication can be reliably achieved is smaller for the simulation model compared to that of the real infrared communication coverage.

7.5 Simulation Model for Robot Motion

To have meaningful simulations it is important to model relevant aspects of the type of performance achieved by a real robot. Our simulations are based on the parameters obtained from experiments with a Koala robot. This robot was used in our validation test conducted in the IPAB corridor (Section 7.9). The Koala robot uses a differential drive mechanism, consisting of two drive wheels mounted on a common axis. Each

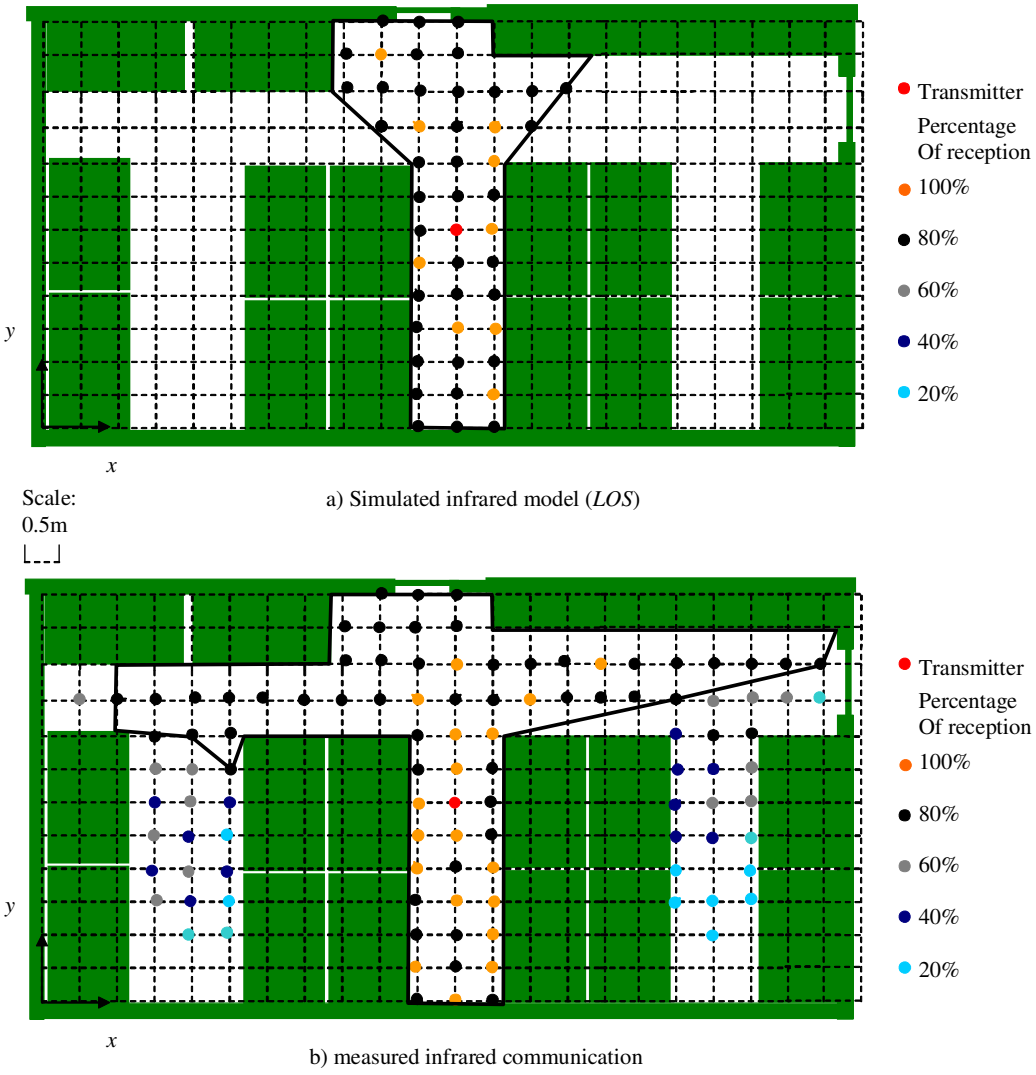


Figure 7.10: Comparison between the simulated *LOS* communication model and the measured infrared communication for a transmitter located at position (5.5,3). For each sampling position five communication samples were taken. The polygon encloses the area where there was at least 80% of success in the communication for all the tested positions. The layout of the office room is shown in green colour.

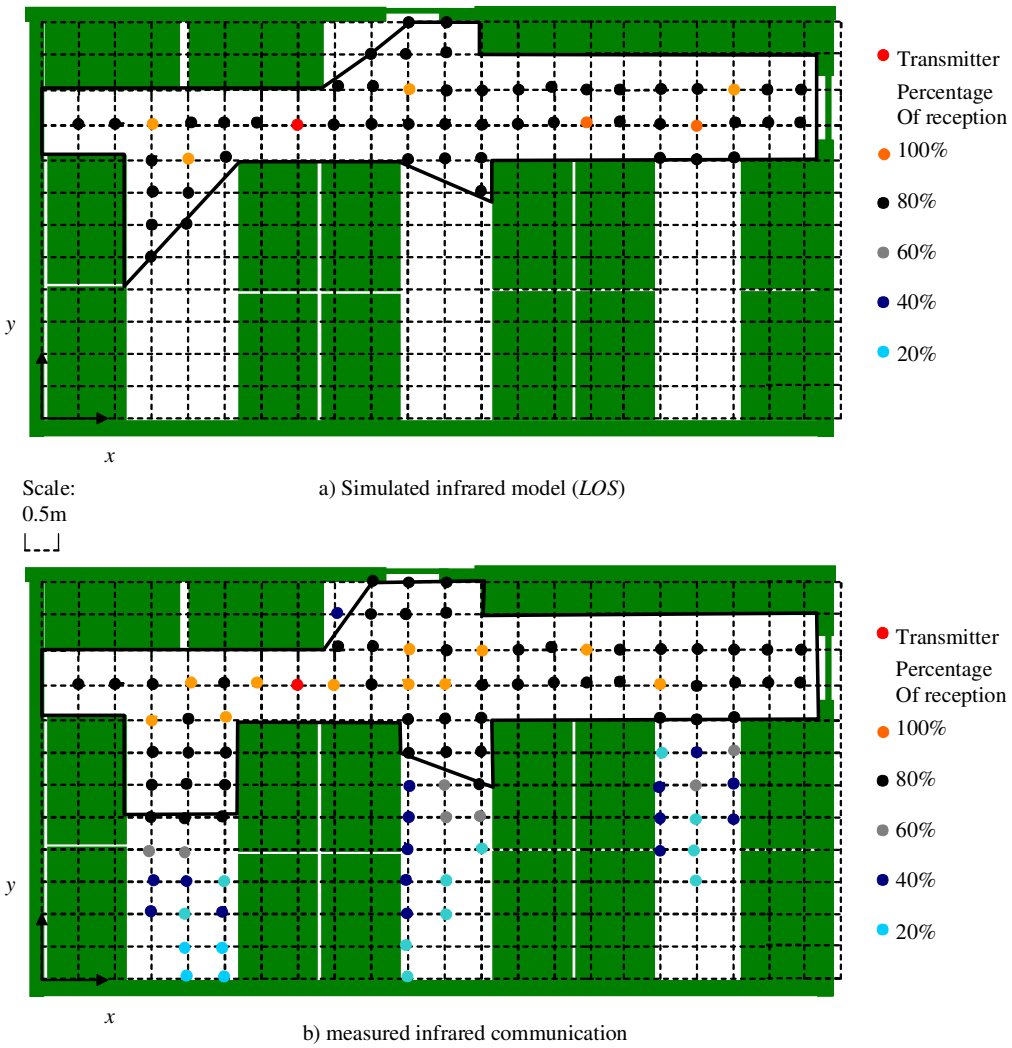


Figure 7.11: Comparison between the simulated *LOS* communication model and the measured infrared communication for a transmitter located at position (3.5,4.5). For each sampling position five communication samples were taken. The polygon encloses the area where there was at least 80% of success in the communication for all the tested positions. The layout of the office room is shown in green colour.

wheel can be driven independently either forward or backward. It can be difficult to make a differential drive robot move in an approximate straight line because the drive wheels are independently driven. Even when the wheels turn at exactly the same rate a robot will often veer to one side because one wheel gets less traction than the other. Our simulated robots model this veering phenomenon.

Each wheel has a shaft encoder which measures the rotation of a shaft attached to the motor. The differential motion of the wheels is obtained from these measurements. These measurements are corrupted by errors due to slippage, skidding and sensor pre-

cision. Frequently the uncertainty of the odometry measurements from the left and right wheel differs. Our simulated robots model this asymmetry in the uncertainty of the odometry measurements.

According to the manufacturer the wheelbase distance for the Koala robot is b=28.5cm. Because the robot has fat rubber wheels which make footprint rather than point contact, the effective wheelbase will vary depending on the nature of the surface. The effective wheelbase must be estimated from odometry experiments. We determined the effective wheelbase for our robot from experiments on carpet and concrete surfaces. These surfaces are common types of surfaces in office environments and were the two surfaces present in the physical environment that we based our simulations (Figure 7.1).

We experimentally determined the uncertainty on the odometry measurements and quantified the amount of veering for our Koala robot. We used the results of these experiments to model our simulated robots.

In the first experiment as suggested by the manufacturer we determined the effective wheelbase of the robot by commanding the robot to rotate a fixed number of revolutions. The precise number of revolutions was measured carefully. The measurements from the left and right shaft encoders were then obtained and the effective wheelbase was calculated. We used two surfaces to test the robot: carpet and concrete.

The distance d that the robot's wheels moved during the rotation is calculated from the arc segment formula $d = \lambda b/2$, where λ is the angle rotated (rad) and b is the effective wheelbase of the robot. Table 7.3 presents the odometry measurements obtained where d_l and d_r are the distances that the robot moved according to the measurements obtained from the left and right shaft encoders. d_{av} is distance that the robot moved according to the average of the encoder measurements (left and right). $b_{effective}$ is the effective wheelbase calculated using the arc segment formula $b_{effective} = 2d_{av}/\lambda$ using the average distance d_{av}. The average effective wheelbase from the experimental data was b=29.3cm which is close to the manufacturers value.

To determine the uncertainty on the odometry measurements and to quantify the veering phenomenon for our robot we conducted an experiment where the robot was instructed to advance in a straight line for a fixed distance d. This experiment was conducted in the concrete and carpet surfaces as the previous experiment. Figure 7.12 shows the procedure used to determine the amount of veering. Our Koala robot had a tendency to veer towards its left side. We instructed the robot to travel for the distances d=1, 2, 3 m. For each distance we performed ten trials. We measured the distance (e)

that the robot apart from the perpendicular to the ideal travel path. We attached a chalk to the centre of the robot wheelbase to measure the distance d_m that the robot travelled forming an arc. Table 7.4 shows the measured distances for the experiment.

Surface	**Revolutions**	d_{av} **(m)**	d_l **(m)**	d_r **(m)**	$b_{effective}$**(m)**
Concrete	5	4.556	4.565±0.022	4.549±0.015	0.290
Concrete	10	9.108	9.112±0.016	9.094±0.022	0.290
carpet	5	4.644	4.651±0.032	4.621±0.045	0.296
carpet	10	9.287	9.277±0.051	9.294±0.048	0.295

Table 7.3: Odometry measurements for a robot that rotates on itself a number of revolutions. d_l and d_r are the distances obtained from the left and right shaft encoders. d_{av} is distance that the robot moved according to the average of the encoder measurements (left and right). $b_{effective}$ is the effective wheelbase distance for the robot.

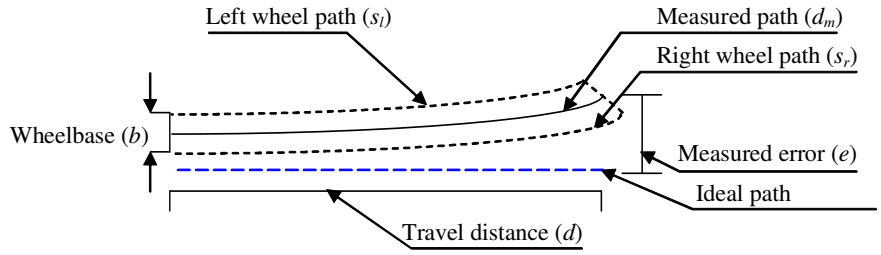


Figure 7.12: Measurement of the drift for the Koala robot. The robot is instructed to move in a straight line a distance d. The robot advances this distance veering towards its left size because of the uneven wheel traction. d_m is the distance that the robot travels forming an arc due to uneven wheel traction.

d	e (m)	d_m (m)	r (m)
3.0	0.212±0.024	3.021±0.042	21.54
2.0	0.095±0.014	2.044±0.021	21.42
1.0	0.025±0.003	1.016±0.023	21.65

Table 7.4: Measured distances for the drifting experiment. d is the distance that the robot was instructed to travel. d_m is the distance that the robot travels forming an arc due to uneven wheel traction. r is the curvature ratio for the robot determined from the measurements.

To calculate the amount of veering we calculated the curvature ratio of the arc formed by the robot path (Figure 7.13). The angle ω for the travelled path d_m around

the arc is determined as follows

$$\psi = \tan^{-1}(d/e) \tag{7.4}$$

$$\omega = 180 - 2\psi \tag{7.5}$$

where d and e are the measured distances from Table 7.4. The curvature ratio is then calculated from the arc length formula

$$r = d_m/\omega \tag{7.6}$$

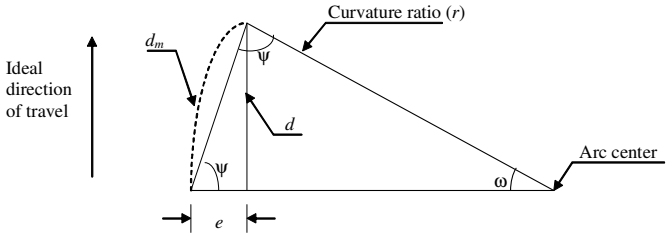


Figure 7.13: The robot forms an arc as it travels with a curvature ratio r. d_m is the distance travelled by the robot along the arc. e is the distance that the robot drifted away from the ideal direction of travel. d is the distance that the robot travelled in the ideal direction of travel.

Table 7.4 shows the curvature ratio obtained for each distance for the average measured distances d_m and e. The average curvature ratio for the three distances is r=21.53m. The ratio between the arc lengths for the right and left wheels is the amount of veering. The amount of veering is determined using the average curvature ratio r, the robot wheelbase $b_{effective}$ (obtained experimentally) and the arc lengths for the left s_l and right s_r wheels of the robot (Figure 7.13). This is calculated by equalizing the arc angle of both wheels

$$\omega_l = \omega_r \tag{7.7}$$

$$\frac{s_l}{r_l} = \frac{s_r}{r_r} \tag{7.8}$$

Where $r_l = r - b_{effective}/2$ and $r_r = r + b_{effective}/2$ because the robot veers towards its left side. The amount of veering is then calculated by solving for s_l and s_r as follows

$$\frac{s_l}{r - b_{effective}/2} = \frac{s_r}{r + b_{effective}/2} \tag{7.9}$$

$$\frac{s_l}{s_r} = \frac{r - b_{effective}/2}{r + b_{effective}/2} \tag{7.10}$$

For our experimental robot the amount of veering found was $s_l/s_r = 0.9868$. To simulate the veering phenomenon the wheels of the simulated robots were driven at different speeds with this experimentally obtained ratio.

Table 7.5 shows the average of the percentage of error in the measurements obtained from the left and right odometry measurements for the different distances and surfaces. This percentage is calculated as

$$encoder_{error} = \frac{\sum_{i=1}^{tests} \left| d_i^{encoder} - d_m \right| / d_m}{tests} * 100\% \tag{7.11}$$

where *tests* is the number of tests (ten for each experiment), d_m is the manually measured distance (Table 7.4) and $d_i^{encoder}$ is the distance obtained from the odometry measurements from the wheel encoder for the i^{th} test. It is observed that the percentage is below 2% for the left and right odometry measurements. It is also observed that odometry error in the measurements is larger for the left wheel compared to that of the right wheel. This difference is due to many factors such as encoder precision, wheel height, wheel wearing, etc. that affect the precision of the odometry measurements. Our simulated robots model this observed asymmetry in the errors.

Surface	***d* (m)**	**Average error for left wheel odometry (%)**	**Average error for right wheel odometry (%)**
Concrete	1.0	1.82%	1.67%
Concrete	2.0	1.87%	1.70%
Concrete	3.0	1.75%	1.65%
Carpet	1.0	1.67%	1.34%
Carpet	2.0	1.80%	1.72%
Carpet	3.0	1.78%	1.67%

Table 7.5: Average error for the measurements obtained from the left and right wheel odometry measurements for different surfaces and distances for the experiment from Figure 7.3.

The average error was below 2% in the measurements over different surfaces. In the simulated environments a constant surface is assumed. To ensure a conservative model we assumed a larger error in the simulation. The inaccuracies on the odometry measurements are simulated by adding random Gaussian noise with zero mean and standard deviation of 3% and 3.5% for the left and right wheel measurements respectively.

In this section we have determined the error in the odometry measurements from the wheels' encoders and the amount of veering for our experimental robot. In our

simulations the veering due to uneven wheel traction was modelled by setting different speeds for the wheels of the robot. The error in the odometry measurements from the wheels' encoders is simulated by adding random Gaussian noise. The simulated robots incorporate the veering phenomenon and the asymmetric error measurements observed in our experimental robot.

7.6 Simulation Model for the Robot Sensors

One of the goals of this work is to validate through simulation the suitability of the use of *low cost* robots to build maps of the environment. In Section 3.6 (page 45) we discussed the suitability of *low cost* sensors for map building purposes. We concluded that sonar and infrared sensors were the most suitable sensors because of their cost and features. We designed and built a *low cost* sensing platform to validate the map building module described in the previous chapter. This platform is based on sonar and infrared sensors and is presented in the following section.

This section presents the simulation model for the sonar and infrared sensors. We used the Devantech Sonar SRF04 and the Sharp Infrared Sensors (Models GP2D120 and GP2Y0A02YK) in our tests. The accuracy of each sensor was evaluated for different types of obstacles and materials. We used the following four types of obstacles: walls, poles, edges and corners. These types of obstacles are some of the most common types of obstacles that are present in indoor environments. The materials used in the tests were white drywall, wood and glass because the environment used in our simulations resembles a section of the IPAB institute which has white dry walls, wood doors and windows. Our simulation models are conservative with respect to the uncertainty and obstacle detection reliability observed in our tests with the sensor devices.

7.6.1 Testing of the Sonar Sensors

The Devantech Sonar SRF04 sensor used in our experiments has a beam-width of 45°. The accuracy in the sonar sensor measurements depends on the incidence angle. For this reason we used two angles of incidence $\omega = 0^\circ, 45^\circ$ for the test with the wall obstacles. Poles and corners are difficult to detect with sonar sensors because often too little signal is reflected back to obtain a measurement. The measurements obtained from the sonar sensor in corner configurations are frequently the result of specular reflections. As a result of this a corner may appear to be at a further distance than it is

in reality. It is important to mention that previous research (Robinson et al., 2004) has found that the Devantech Sonar suffers less from the problem of specular reflections than the Polaroid 6500 sonar sensors which are of common use in robotic applications.

An experiment to determine the uncertainty of the sonar sensor measurements for the different types of obstacles and materials was conducted. In the experiment the sensor was initially placed at 10cm from the obstacle, and then moved away from the obstacle in intervals of 10cm until the maximum range distance was reached. The maximum range for the SRF04 is 3.0m according to the manufacturer data. Figure 7.14 shows the measurement procedure for the different types of obstacles used.

The sonar sensor measures the distance to the closest obstacle by sending sonar pulses and measuring the time of flight of the pulses. When no obstacles are detected the time of flight is set to a maximum time which is double the time of the maximum detection distance.

We used two sonar sensors in the *low cost* platform that we built (Section 7.7). The platform is on top of a servo motor that can rotate only 180°, therefore one sonar is placed at the front of the platform and the other at the rear of the platform to cover the 360° circumference. We used both sonar sensors to obtain an average performance in our test. We took ten valid measurements at each location (five for each sonar). A measurement was valid when the obstacle was detected. We used the valid measurement to determine the uncertainty of the sensor measurements and the invalid measurements to determine the obstacle detection reliability of the sensor. The uncertainty of the sensor measurement is determined from the measured error. The measured error is the absolute difference between the obstacle distance and the received from the sonar sensor. Table 7.6 shows the average measured error and the obstacle detection reliability for the sonar sensor for all the range of tested distances to the obstacle.

Figure 7.15 shows the measured error for the different types of obstacles as a function of the distance. For the wall obstacles the graph shows the average for the three types of material tested. Figure 7.16 shows the reliability for the different types of obstacles as a function of the distance. By comparing the results from the perpendicular wall obstacle against the wall with a 45° incidence angle it is observed that as expected the measured error and the obstacle detection reliability depends on the angle of incidence.

The behaviour of the sonar sensor observed in our tests can be described as follows: The measured error increases linearly as the distance to the obstacle increases, and the obstacle detection reliability decreases linearly as the distance to the obstacles increase.

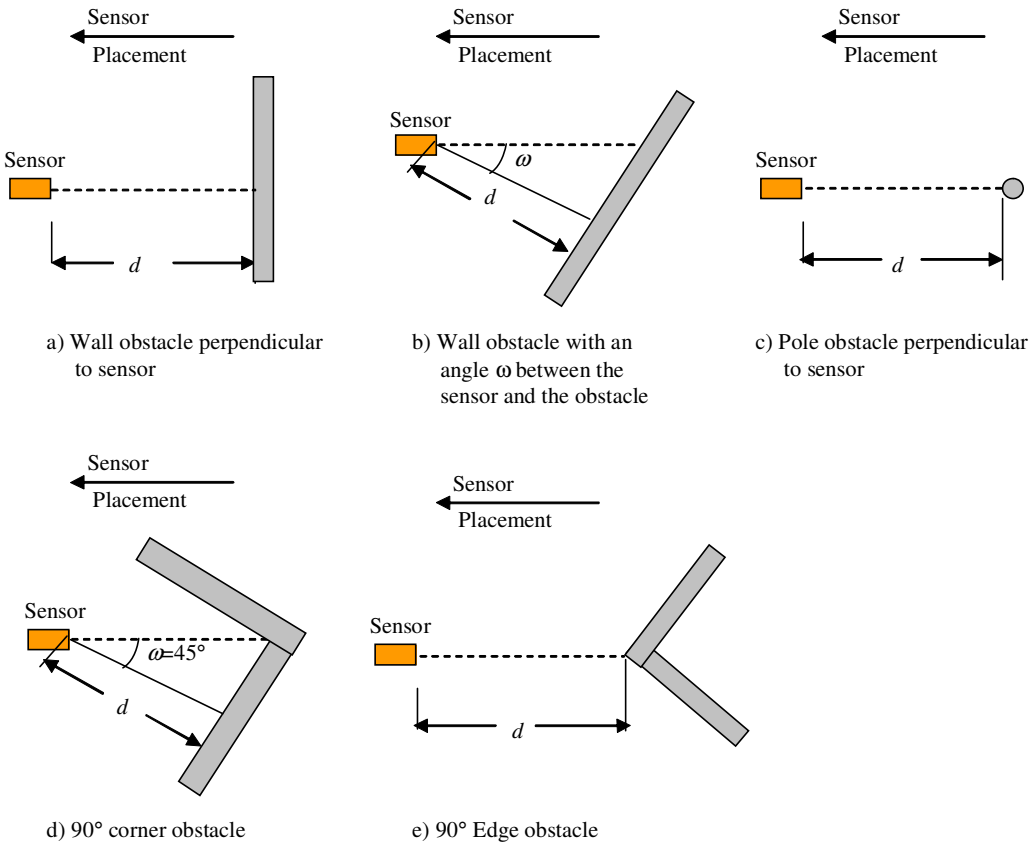


Figure 7.14: Procedure used to determine the accuracy of the sensor for different types of obstacles. The sensor was located at a distance d and moved away from the obstacle in the sensor placement direction.

Type of obstacle	**Material**	**measurement error (cm)**	**Reliability (%)**
Perpendicular wall	White drywall	1.2±0.15	98.6%
Perpendicular wall	Wood	1.4±0.21	97.5%
Perpendicular wall	Glass	1.3±0.17	97.7%
Wall with 45° incidence angle	White drywall	2.3±0.24	91.1%
Wall with 45° incidence angle	Wood	2.1±0.19	92.7%
Wall with 45° incidence angle	Glass	2.5±0.16	93.7%
Round pole (diameter *d*=2cm)	Aluminium	2.1±0.23	81.4%
90° corner obstacle	White drywall	2.9±0.17	94.2%
90° edge obstacle	White drywall	3.5±0.24	84.6%

Table 7.6: Average measured error and reliability of the sonar sensor measurements for different types of obstacles and materials.

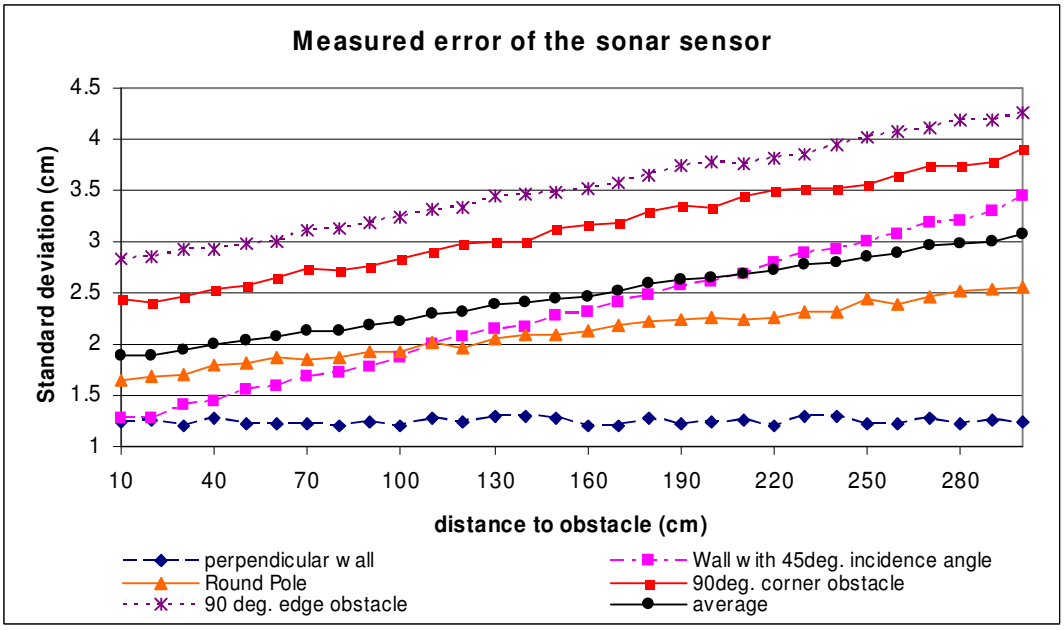


Figure 7.15: Measured error of the sonar sensor for different types of obstacles and its average. In average the measured error has a linear increase as the distance to the obstacle increases. With the exception of the perpendicular wall obstacle, for all the trends show a linear increase in the error as the distance to the obstacle increases.

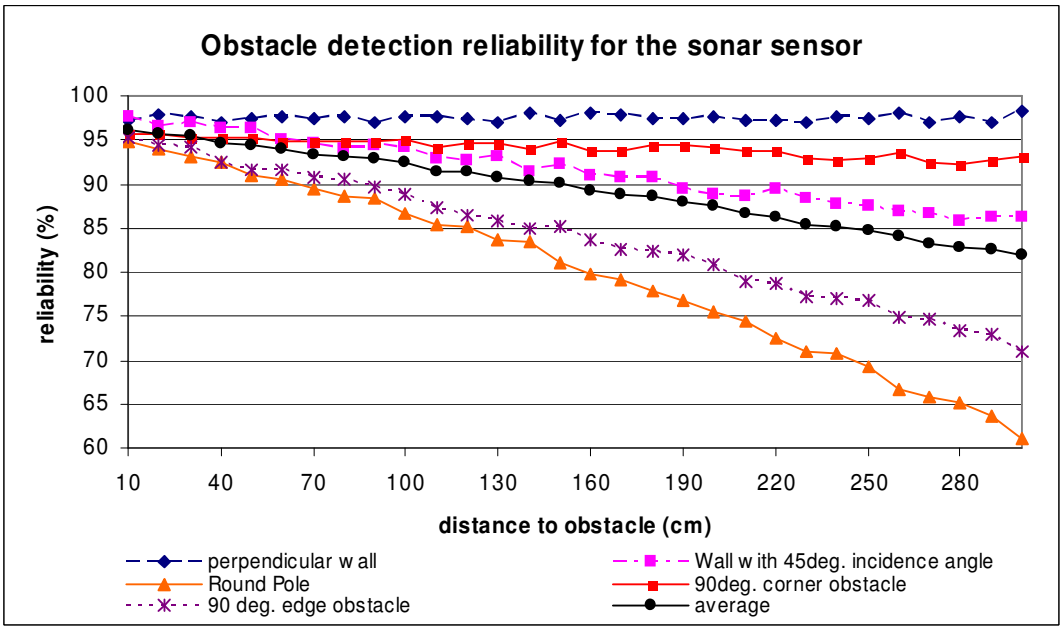


Figure 7.16: Reliability of the sonar measurements for different types of obstacles and its average. Reliability has a linear decrease as the distance to the obstacle increases. For all the types of obstacles there is a decrease in the obstacle detection reliability.

In the following section we describe the simulation model for the sonar which is based on the observed behaviour.

7.6.2 The Simulated Sonar Sensors

The simulated sonar sensor is based on the results from the tests of the previous section. The Webots simulator does not provide a model for the sonar sensors. The simulator provides a model for infrared sensors and uses a ray tracing algorithm to measure the exact distance to the closest obstacle. The sonar sensor used in our tests has a beam width of 45°. Our simulation model of this sonar consists of a collection of infrared sensors. An infrared sensor is orientated in the direction of the sonar beam centre and the subsequent sensors are placed with an increase in orientation of 2° on both sides of the beam centre until the orientation with respect to the beam centre is 22°.

The distance to the obstacle d_o and the angle of the obstacle with respect to the centre of the simulated sonar beam ω_o are obtained from the infrared sensor in the collection that has the smallest distance to the obstacle. When there is no obstacle within the range of the sonar the distance is d_{max} (d_{max}=3.0m for our simulated sonar). Our simulation model assumes that the sonar responds only to the first obstacle detected.

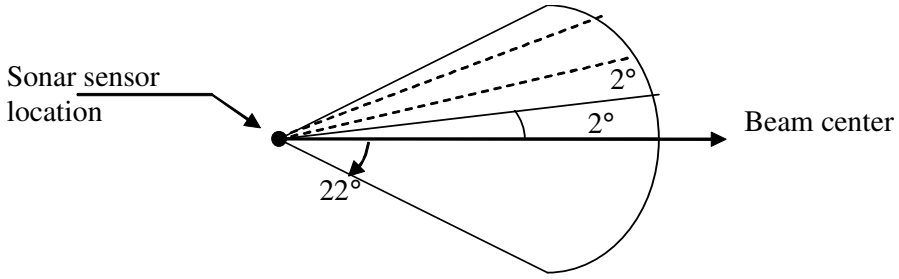


Figure 7.17: The simulated sonar sensor modelled as a collection of infrared sensors for the Webots simulator. The infrared sensors return the distance to the closest obstacle using a ray tracing algorithm. The measurement for the sonar is the smallest distance from the collection of the infrared measurements.

From the previous section we observed that the reliability of obstacle detection and the error in the sonar measurements depend on the distance and the angle of incidence. Our simulated sonar measurements are modelled as a function of these two variables. For our simulated measurement we first calculated the probability of detecting the obstacle and then calculated the obstacle distance as a function of this probability.

The obstacle detection reliability for the simulated sonar is modelled using the

following equation

$$P_{detect}(\omega_o, d_o) = 0.95 - 0.20(d_s/d_{max}) - 0.25(\omega_o/22) \quad (7.12)$$

where $P_{detect}(\omega_o, d_o)$ is the probability that the obstacle will be detected. The $0.20(d_s/d_{max})$ factor models the decrease in the reliability as a function of the distance. This factor models the observed linear decrease for the average performance of the sonar with the different types of obstacles tested (Figure 7.16). The $0.25(\omega_o/22)$ factor models the decrease in the reliability as a function of the incidence angle. In our tests we corroborated that the reliability in obstacle detection decreases when the angle of incidence is larger. This was observed in Figure 7.16 for the wall obstacle with two different incidence angles.

The addition of uncertainty for the sonar measurement is carried out using the following equation

$$d_{sonar} = d_o + gaussian(\delta_{base}) + gaussian(\delta_{distance} * (d_o/d_{max})) \quad (7.13)$$

where d_{sonar} is the distance that integrates the uncertainty and is measured in centimetres. $gaussian(\delta)$ is a random Gaussian distribution with mean zero and standard deviation δ. The uncertainty in the sonar measurements have a base uncertainty δ_{base} and an uncertainty that is a function of the distance $\delta_{distance} * (d_s/d_{max})$. For our simulated sonar we selected the values of $\delta_{base} = 3$cm. and $\delta_{distance} = 2$cm. because these values are a conservative approximation for the worst case of the measured error found in our tests with the sonar device (Figure 7.15).

The integration of uncertainty and obstacle detection reliability for the simulated sonar measurements is done using the algorithm 7.1. The variable *rand*() is a random variable with a linear distribution in the range [0,1].

7.6.3 Testing the Infrared Sensors

The Sharp infrared sensors ((Models GP2D120 and GP2Y0A02YK) used in our experiments have a beam-width of 2°. Their range is 0 to 0.3m for the Sharp GP2D120 and 0.25m to 1.5m for the Sharp GP2Y0A02YK. These sensors are very insensitive to ambient light according to Konienko et al. (2005) tests. This was confirmed in our experiments. The error in the Sharp sensors increases exponentially with respect to the distance. When there is no obstacle detected the measured distance is a large value (above 1.5m for the Sharp GP2Y0A02YK and above 0.3m for the Sharp GP2D120).

Algorithm 1 Addition of uncertainty and obstacle detection reliability for the sonar sensor

procedure add_Uncertainty(d_o, ω_o)
$detection = rand()$
if $P_{detect}(\omega_o, d_o) < detection$ **then**
  $d_{sonar} = d_{max}$
else
  $noise_{base} = gaussian(\delta_{base})$
  $noise_{dist} = gaussian(\delta_{distance} * (d_o/d_{max}))$
  $d_{sonar} = d_o + noise_{base} + noise_{dist}$
end if
return d_{sonar}

To determine the uncertainty of measurements from the Sharp sensors we conducted the same experiment as for the sonar sensors (Section 7.6.1). The same types of obstacles and materials were used in both experiments. One of each type of these sensors were mounted as a pair on a sensing platform to cover the entire range from 0 to 1.5m (Section 7.6.1.). We used the three pairs of sensors in this test to obtain an averaged performance because these three pairs of sensors were used in the *low cost* platform that we built (Section 7.6.1).

In the experiment the platform was initially placed at 2cm. from the obstacle, and then moved away from the obstacle in intervals of 2cm. until the maximum range distance for the Sharp GP2D120 was reached (30cm.). Afterwards intervals of 5cm. were used until the maximum range distance for the Sharp GP2Y0A02YK was reached. Figure 7.14 shows the measurement procedure for the different types of obstacles and materials used. Table 7.6 shows the list of the types of obstacles and the material that they are made of.

In the test we took fifteen valid measurements at each location (five for pair of sharp sensors). A measurement was valid when the obstacle was detected. We used the valid measurement to determine the uncertainty of the sensor measurements and the invalid measurements to determine the obstacle detection reliability of the sensor. The uncertainty of the sensor measurement is determined from the measured error. The measured error is the absolute difference between the obstacle distance and that received from the sonar sensor.

Figure 7.18 shows the measured error for four of the seven obstacles used in the tests. It is observed that the error is much larger for the obstacle made of glass (glass

wall obstacle) than for the other obstacles made of other material. This is to be expected because infrared sensors have difficulty in detecting obstacles made of glass because the signal emitted by these is poorly reflected by this surface. In our tests we did not find a large difference in the measured errors with respect to the type of obstacle, only with respect to the type of material. This can be observed in Figure 7.18 for the round pole, the corner and the edge obstacle. Figure 7.19 shows the measured error of the pair of Sharp infrared sensors grouping them by their type of material: glass and non-glass. The average error was obtained by averaging over the same collection of typical obstacle types as we used for our sonar, of which two were glass and seven non-glass. It is observed that regardless of the type and the material of the obstacle the error increases exponentially with respect to the distance for the infrared sensor.

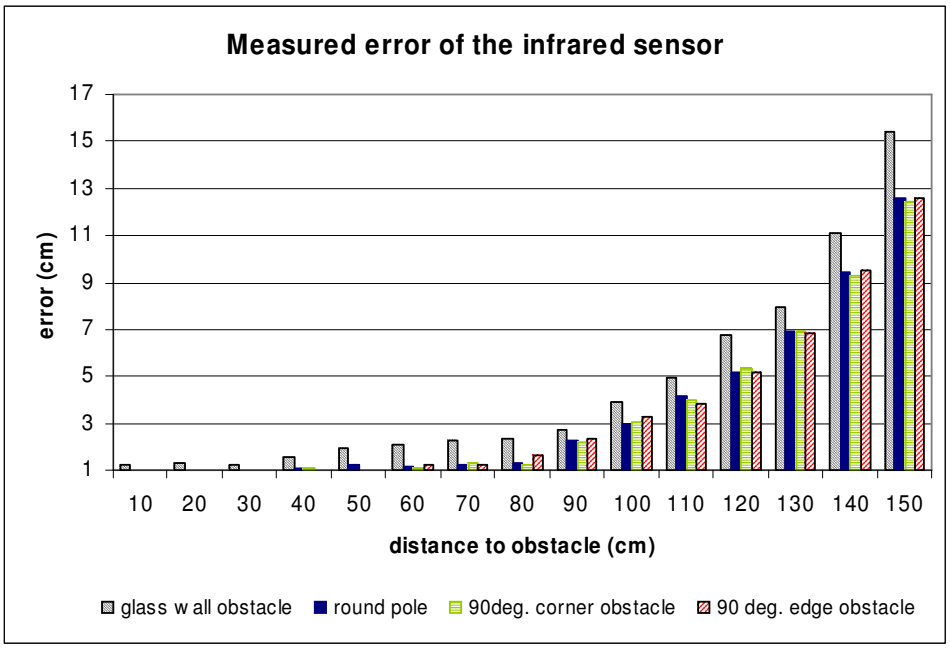


Figure 7.18: Measured error of the Sharp infrared sensor for different types of obstacles. With the exception of the glass obstacle the measured error is similar for the different types of obstacles. For all the trends the error increases exponentially with respect to the distance to the obstacle.

As in the case of the measured error in our tests we did not find a large difference in the obstacle detection reliability with respect to the type of obstacle, only with respect to the type of material. This can be observed in Figure 7.20 for the round pole, the corner and the edge obstacle. Figure 7.21 shows the obstacle detection reliability of the pair of Sharp infrared sensors, grouping them by their type of material: glass and non-glass. The error was averaged over the nine types of obstacle, of which two were

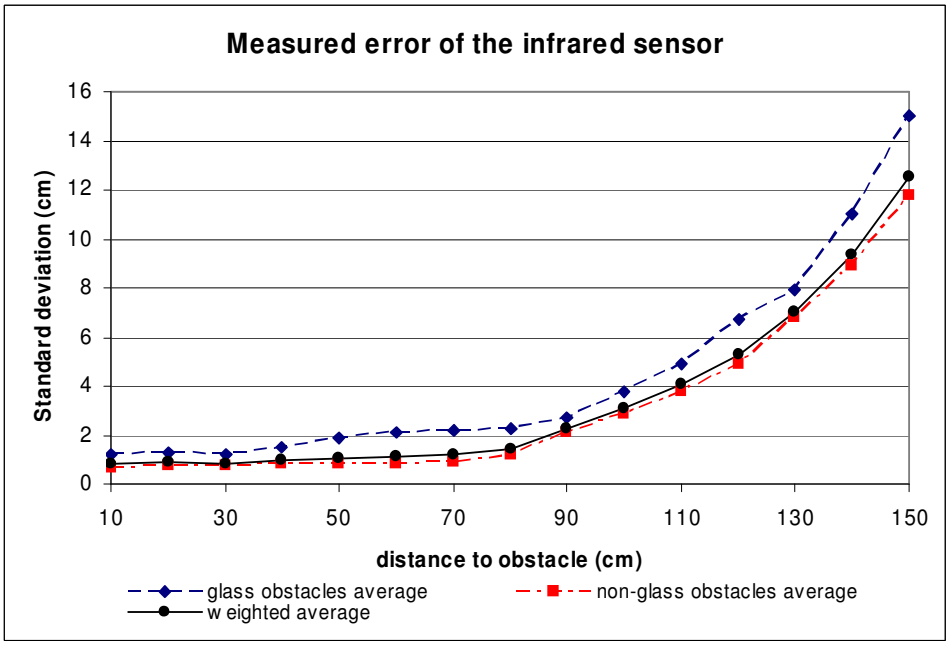


Figure 7.19: Measured error of the Sharp infrared sensor for glass obstacles and non-glass obstacles. Two types of glass obstacles and seven different types of non glass obstacles were used in to measure the error. The average is closer to the non-glass average because there were more non-glass obstacles. For all the trends the error increases exponentially with respect to the distance to the obstacle.

glass. It is observed that the reliability in the detection of an obstacle decreases when the distance to the obstacle increases.

7.6.4 The Simulated Infrared Sensor

The simulated infrared sensor is based on the results from the tests of the previous section. The simulator provides a model for infrared sensors that uses a ray tracing algorithm to measure the exact distance to the closest obstacle d_o. When there is no obstacle within the range of the sensor the distance is d_{max} (d_{max}=1.5m for our simulated infrared sensor).

The simulated infrared measurements are modelled as a function of the distance to resemble the behaviour observed in the tests of the previous section. For our simulated measurement we first calculated the probability of detecting the obstacle and then calculated the obstacle distance as a function of this probability.

We want to avoid an over-optimistic simulation which would give better results than could be achieved in practice. The obstacle detection reliability for the simulated

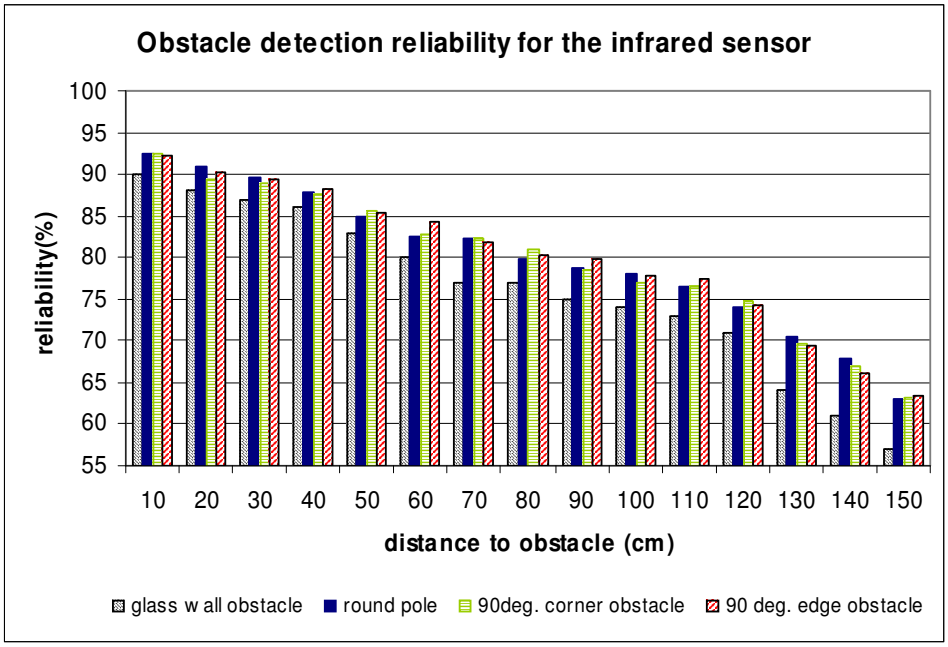


Figure 7.20: Obstacle detection reliability of the Sharp infrared sensor for different types of obstacles. With the exception of the glass obstacle the reliability is similar for the different types of obstacles. For all the trends the error increases exponentially with respect to the distance to the obstacle.

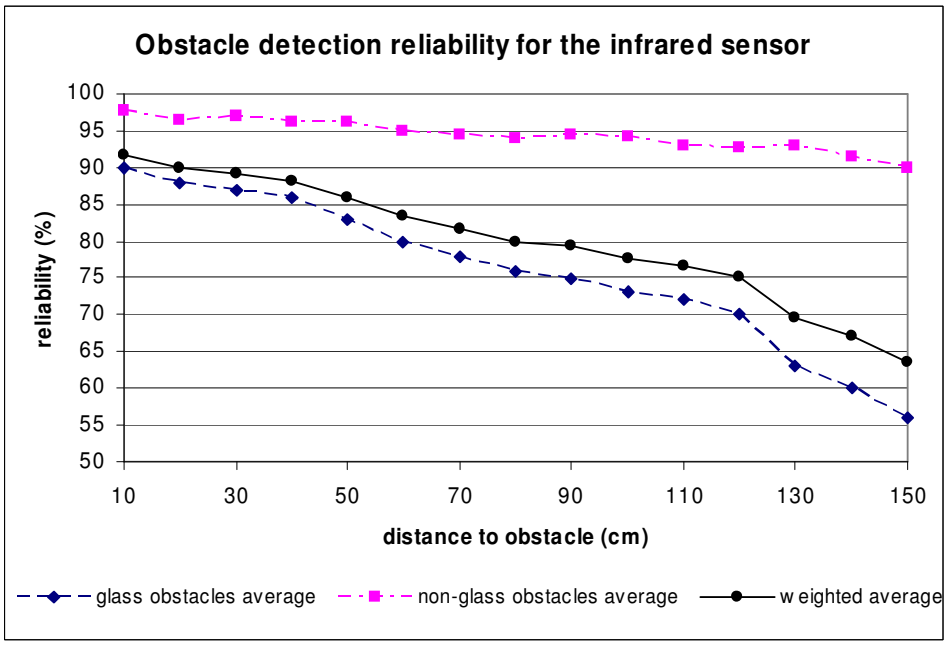


Figure 7.21: Obstacle detection reliability for the Sharp infrared sensor for glass obstacles and non-glass obstacles. Two types of glass obstacles and seven different types of non glass obstacles were used to determine the obstacle detection reliability. The average is closer to the non-glass average because there were more non-glass obstacles.

infrared sensor is modelled using the following equation

$$P_{detect}(d_o) = 0.90 - 0.35(d_s/d_{max}) \tag{7.14}$$

where $P_{detect}(d_o)$ is the probability that the obstacle will be detected. The $0.35(d_s/d_{max})$ factor models the decrease in the reliability as a function of the distance. This factor models the worst case of obstacle detection reliability for the different materials and obstacle types found in our tests (Figure 7.21). The worst reliability was found for the obstacles made of glass. The simulation model is then conservative with respect to the tested materials and obstacle types.

The Sharp infrared sensors have an exponential error with respect to the measured distance. The uncertainty added to the simulated infrared measurements was then determined from the following exponential equation[1]

$$\nu(d_0) = 0.16e^{0.03d_0} + 1.5 \tag{7.15}$$

where $\nu(d_0)$ is measured in centimetres. This equation models the error observed in our measurement tests for the worst case (glass material). Figure 7.22 shows the measurement error generated by the exponential equation (conservative fitting trend) and the measured error for the tested obstacles. It is observed that the equation model is conservative because the expected error is always larger than the error measured in the tests.

The integration of uncertainty and obstacle detection reliability for the simulated infrared measurements is done using the algorithm 7.2. $d_{infrared}$ is the distance that integrates the uncertainty and is measured in centimeters. The variable *rand*() is a random variable with a linear distribution in the range [0,1]. $gaussian(\nu(d_0))$ is a random Gaussian distribution with mean zero and standard deviation $\nu(d_0)$.

7.6.5 Validation of the Sonar and Infrared Models

In order to validate the sonar and infrared simulation models we compared the simulation models against real measurements obtained from the sonar and the infrared sensors. We used the experimental environment from Figure 7.23 to validate the measurements. The environment contains the types of features that we expect to find in an office environment such as poles (chair legs), corners and edges. The wall obstacles are made of different materials. To have a good comparison basis for the sensor

[1] The curve derived from linear least squares fitting. After the curve was fitted to the data it was manually adjusted to be conservative.

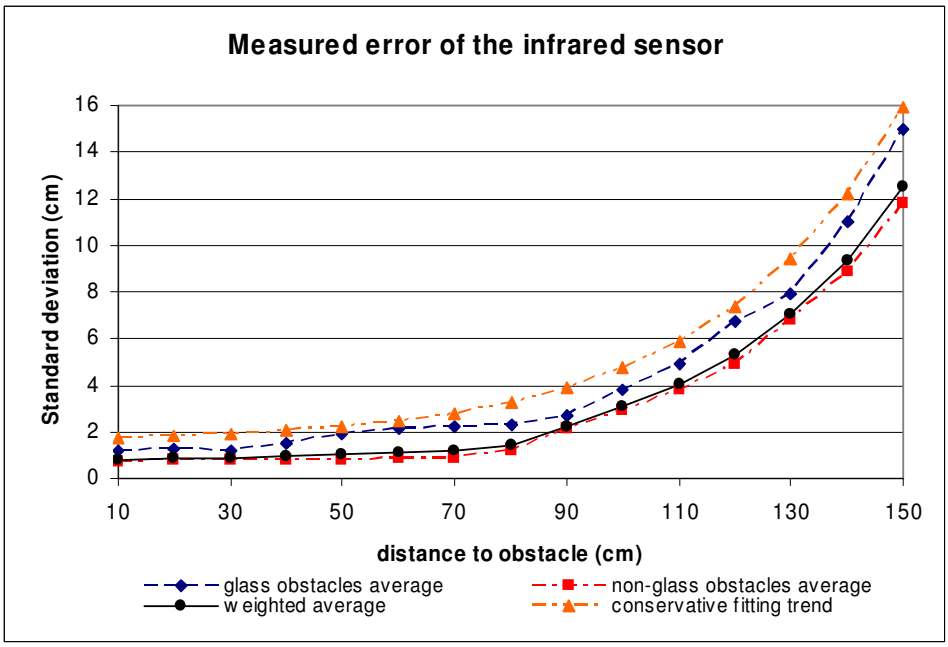


Figure 7.22: Measured error of the Sharp infrared sensor for glass obstacles and non-glass obstacles. Two types of glass obstacles and seven different types of non glass obstacles were used in to measure the error. The average is closer to the non-glass average because there were more non-glass obstacles. The conservative fitting trend is derived from the glass obstacle trend.

Algorithm 2 Addition of uncertainty and obstacle detection reliability for the infrared sensor

procedure add_Uncertainty(d_o)
$detection = rand()$
if $P_{detect}(d_o) < detection$ **then**
  $d_{infrared} = d_{max}$
else
  $noise = gaussian(\upsilon(d_o))$
  $d_{infrared} = d_o + noise$
end if
return $d_{infrared}$

measurements the environment size was selected so that for a sensor scanning in most of the directions there were obstacles within the range of the sensor. In the simulation of this environment the obstacles are assumed to be built of the same material because the simulated robot sensors measure the distance to the obstacle but the material of the obstacle sensed is unknown. To avoid over-optimistic simulations our sensor models are based on the worst case for the materials that we tested in our experiments.

The sonar was mounted on top of a servo motor to produce scans of the environment. The platform is described in Section 7.7. This platform has two sonars, one at the front and one at the rear. The platform is rotated 180° to obtain a complete scan of the circumference. To avoid cross talking between the front and the rear sonar for our platform we fire them sequentially 65milliseconds apart. The detection cycle for the sonar sensors times out after 40ms.

We took 360° scans at several locations in our experimental environment, in this section we present the results obtained for two testing positions (Figure 7.23). In the scans we took ten measurements for each sensor orientation and then rotated the platform in steps of 15°.

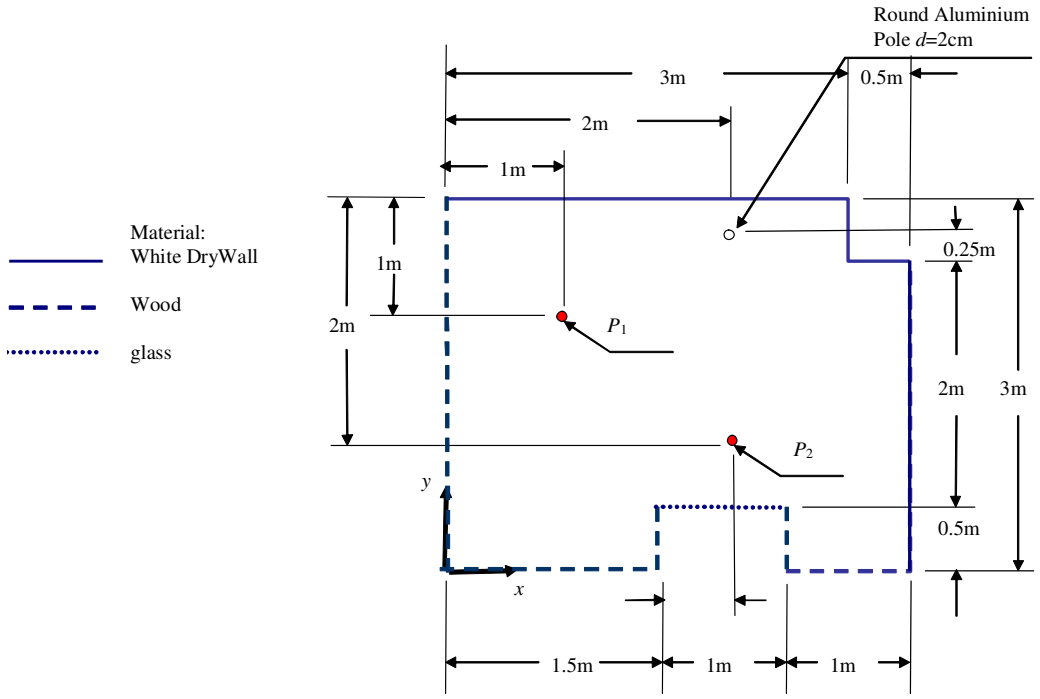


Figure 7.23: Environment used to validate the sonar and infrared simulation models. The measurements are hand measured. The walls of the environment are made of wood, glass, and white DryWall. p_1 and p_2 are testing positions to assess the performance of the simulation models.

Figures 7.24 and 7.25 show the average measurement distance for the real and the simulated sonar scans for the testing positions p_1 and p_2 from Figure 7.23. The line orientation is the orientation of the centre of the beam. The lines show the average of ten measurements for each location. When there was no obstacle detection the maximum distance was returned. The red dotted lines indicate that in at least four measurements there was no obstacle detection. It may appear that the measured distance at some orientations is shorter than it should be; this is because the obstacle could be at the edge of the 45° beam, but we show it as though it was along the central beam axis. It is observed that for the red dotted lines the incidence angle in general terms is larger than for the other orientations. In the absence of an obstacle the sensors report a maximum distance. Including this in the average distance makes the average sensitive to obstacle detection rate. Nevertheless, the scans from the simulated and the real sonar are qualitatively similar testifying to the goodness of this simulation.

Figure 7.26 and 7.27 show the histograms of obstacle detections for the testing positions p_1 and p_2 for the real and the simulated sonar scans. By correlating the scans from Figures 7.24 and 7.25 with the histograms it is observed that the real sonar fails to detect the obstacles more frequently for larger distances with large angles of incidence. When the obstacles are perpendicular to the orientation of the measurement and the distance is small the real sonar almost never fails to detect the obstacle. From the histograms it is observed that the simulated sonar scans resemble the obstacle detection pattern of the real sonar scans. In the real scans in 71.87% of the measurements an obstacle was detected while in the simulated scans in 73.33% of the measurements an obstacle was detected.

These examples show that the simulation model for the sonar sensor reflects the behaviour of the real sonar sensor. The scans obtained with the simulated and the real sonar sensor are qualitatively similar and the obstacle detection reliability was similar (71.87% against 73.33%).

Figures 7.28 and 7.29 show the average measurement distance for the real and the simulated infrared scans for the testing positions p_1 and p_2 from Figure 7.23. The red dotted lines indicate the orientations that are out of the range of the infrared sensors. In the absence of an obstacle the sensors report a maximum distance (1.5m). Including this in the average distance makes the average sensitive to obstacle detection rate. For this reason it may appear that the measured distance is larger than it should be. It is observed that for both scans (real and simulated) the error in the measurements increases when the distance between the sensor and the obstacle increases. This is

because the real infrared sensors have an exponential increase in error with respect to the distance. The experimental environment contains a glass wall on the middle bottom section of the environment. The detection of glass obstacles is difficult to achieve for the real infrared sensor. This is observed in the scan with the real sensor in Figure 7.29. The errors in the measurements for the glass are large even though the distance to the wall is small (0.5m in the y direction) and the error is exponential for the infrared sensors.

Glass obstacles were the worst case of the tested types of obstacles. The simulated robots can't distinguish obstacle types, so we have to use an average. The average for the simulated measurements is only slightly better than that observed in this real worst case; nevertheless the infrared simulation model produces qualitatively similar scans to the scans from the real infrared sensor. In other words the simulation is conservative, and better performance can be expected in reality.

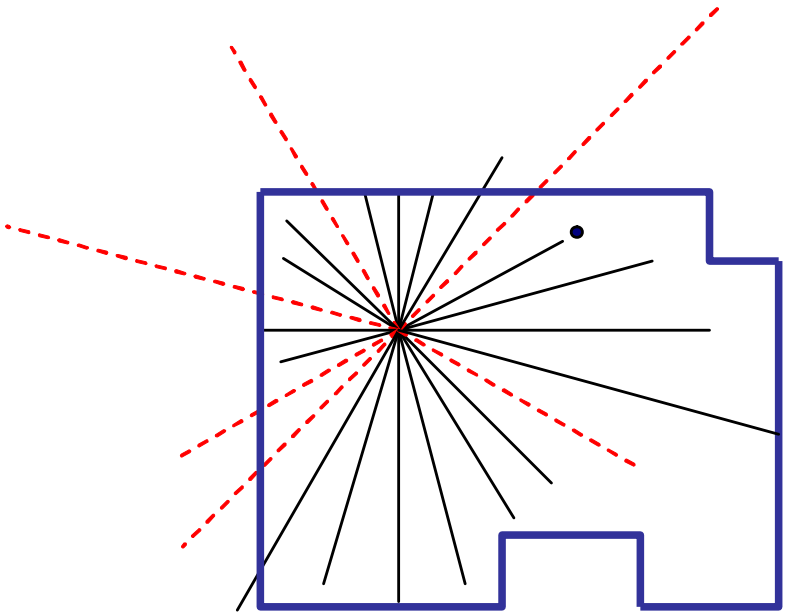
Figure 7.30 and 7.31 show the histograms of obstacle detections for the testing positions p_1 and p_2 for the real and the simulated infrared scans. For some orientations the detection of obstacle was not possible because the obstacles were out of the infrared sensing range (e.g 270° - 300°).

By correlating the scans from Figures 7.28 and 7.29 with the histograms it is observed that the real infrared sensor fails to detect the obstacles more frequently for larger distances. This pattern is also observed for the simulated infrared measurements. For instance, in Figure 7.30 in the orientation range from 30° to 135° the left wall is the closest obstacle. The number of obstacle detections peaks at the 90° orientation (shortest obstacle distance for the orientation range) and decrease linearly on both sides of this orientation range.

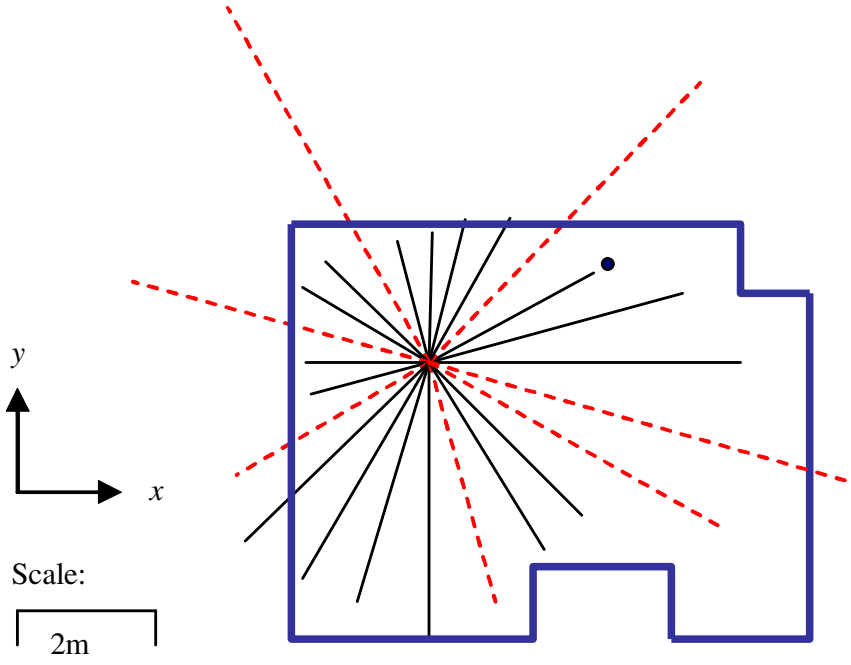
The percentage of measurements in which an obstacle was detected was 51.38% and 50% for the real and the simulated scans. This percentage was calculated discarding the orientations in the scan for which the obstacles were out of the sensing range.

These illustrative examples show that the simulation model for the infrared sensor reflect the behaviour of the real infrared sensor. The scans obtained with the simulated and the real infrared sensors are qualitatively similar and the obstacle detection reliability was similar (51.38% against 50.00% for the tested positions).

In this section we have validated the simulation model for the sonar and infrared sensors through experimentation with the real sensors. The simulation models are based on the worst conditions observed in our tests with the sensor devices. In the



a) Real Sonar scan



b) Simulated Sonar scan

Figure 7.24: Real and simulated sonar scans for the testing position p_1 from Figure 7.23. The lines show the average for ten measurements. The sonar has a beam width of 45°. The line orientation is the orientation of the centre of the beam. The red dotted lines indicate that at least four measurements there was no obstacle detection.

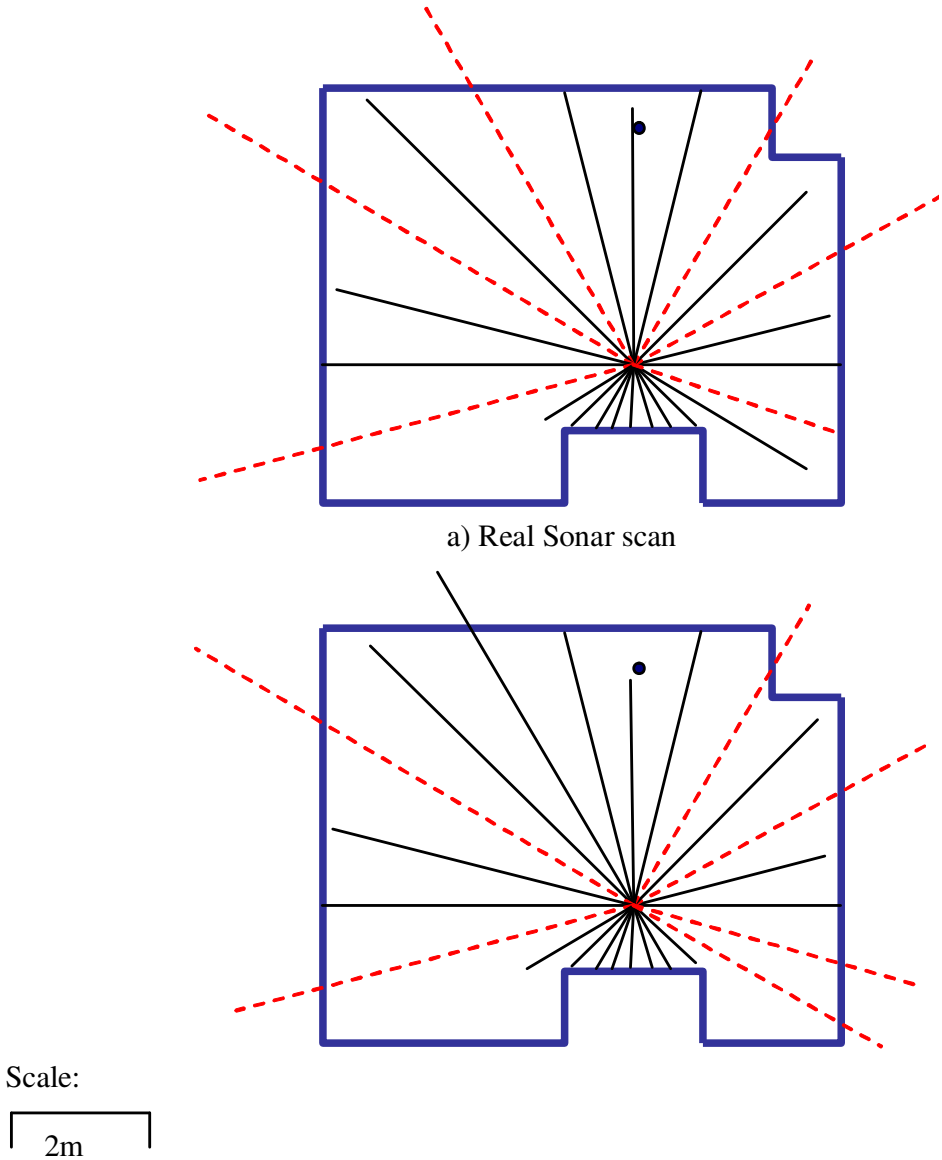


Figure 7.25: Real and simulated sonar scans for the testing position p_2 from Figure 7.23. The lines show the average for ten measurements. The sonar has a beam width of 45°. The line orientation is the orientation of the centre of the beam. The red dotted lines indicate that at least four measurements there was no obstacle detection.

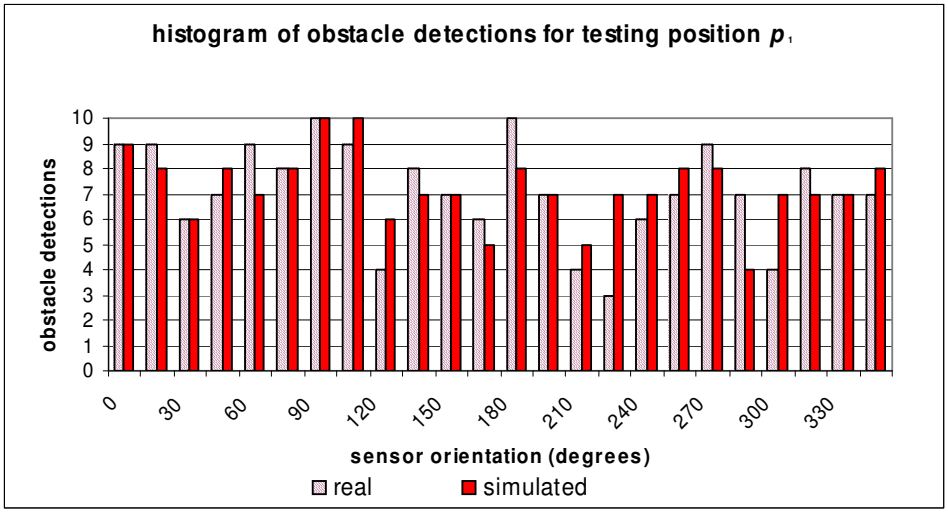


Figure 7.26: Histogram of obstacle detections for the real and the simulated sonar scan in the testing position p_1 (Figure 7.24). The *x* axis shows the orientation of the sonar sensor in degrees. The *y* axis shows the number of obstacle detections. Ten measurements were taken for each orientation of the sensor

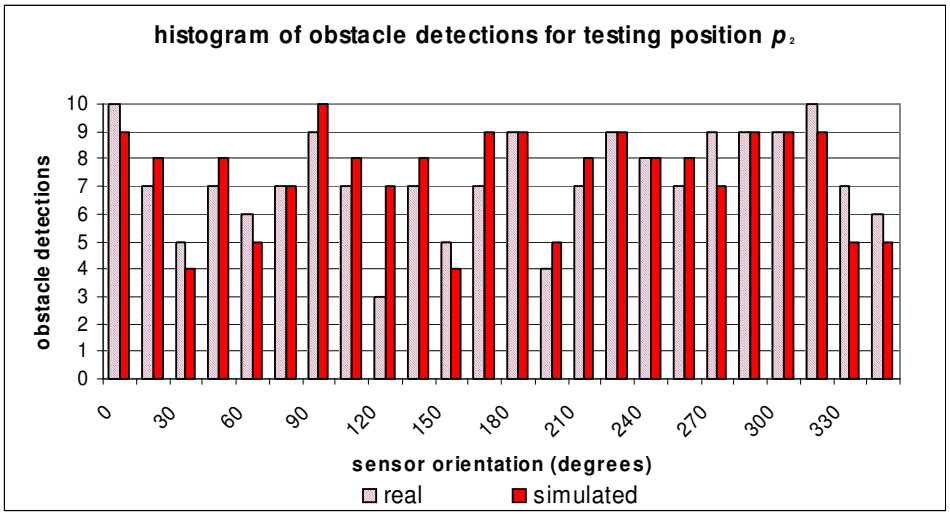
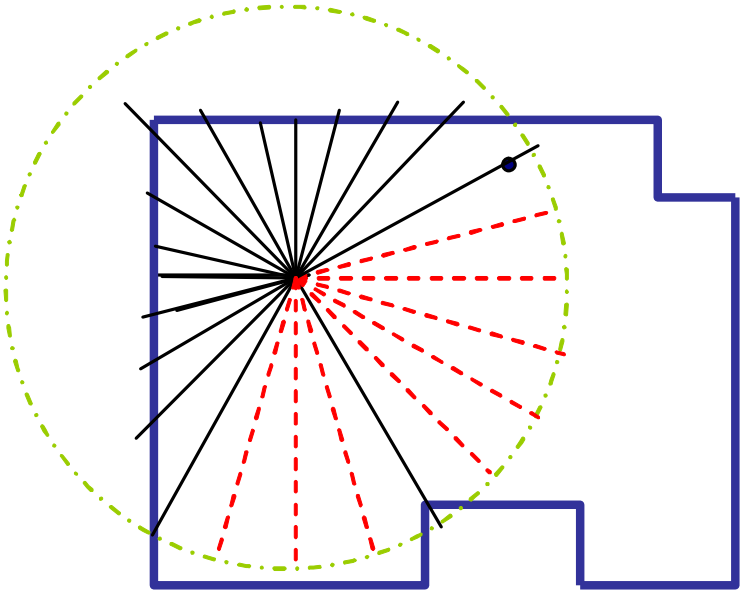
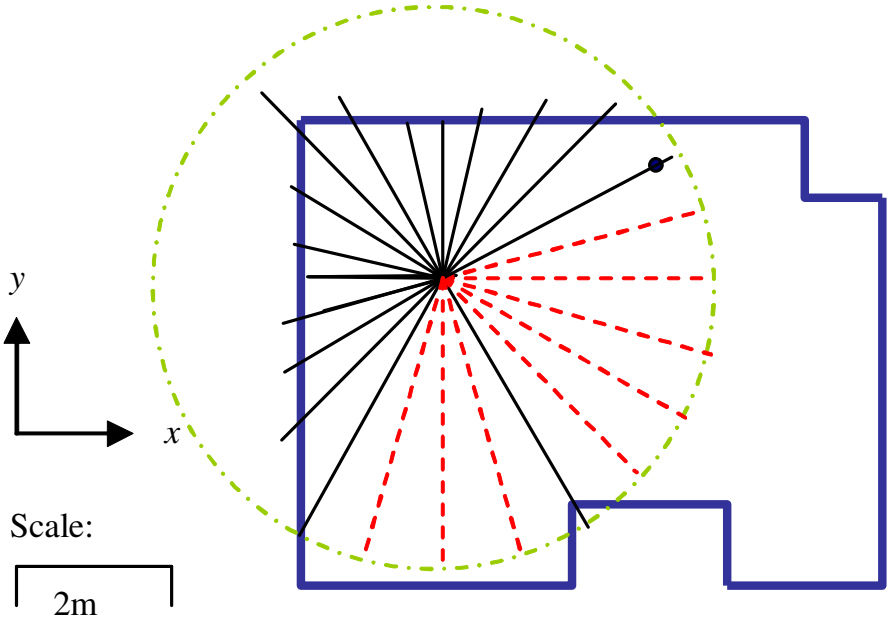


Figure 7.27: Histogram of obstacle detections for the real and the simulated sonar scan in the testing position p_2 (Figure 7.25). The *x* axis shows the orientation of the sonar sensor in degrees. The *y* axis shows the number of obstacle detections. Ten measurements were taken for each orientation of the sensor



a) Real infrared scan



a) Simulated infrared scan

Figure 7.28: Real and simulated infrared scans for the testing position p_1 from Figure 7.23. The lines show the average for ten measurements. The red dotted radial lines indicate the orientations that are out of the infrared sensor range. The circumference shows the maximum range for the infrared sensors at that location.

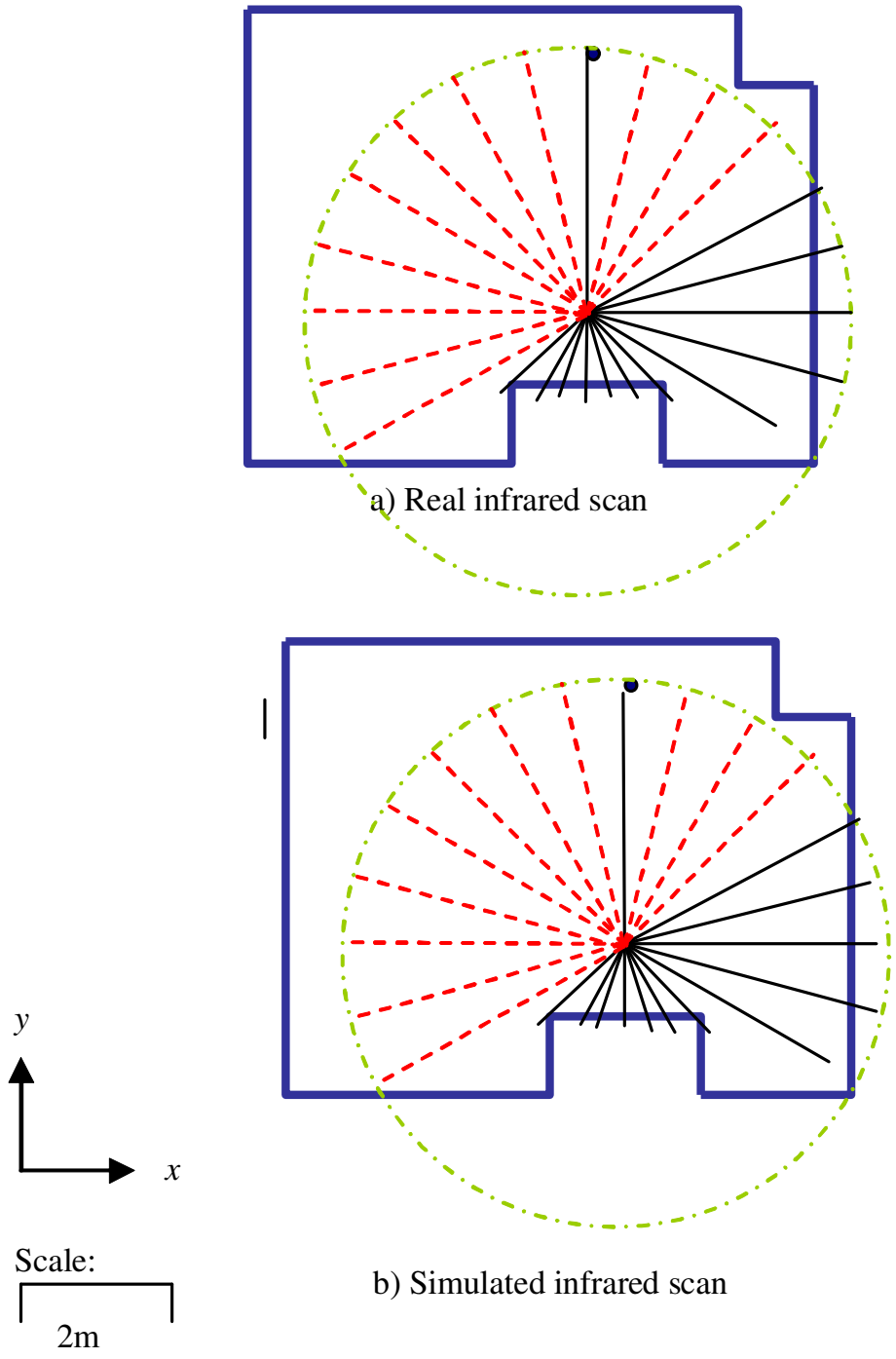


Figure 7.29: Real and simulated infrared scans for the testing position p_2 from Figure 7.23. The lines show the average for ten measurements. The red dotted radial lines indicate the orientations that are out of the infrared sensor range. The circumference shows the maximum range for the infrared sensors at that location.

examples shown in this section we have observed that the simulation models resemble the average performance of the sensor devices in terms of sensor uncertainty and obstacle detection reliability. It is concluded that our simulation models are reasonable and conservative approximations of the experimental data obtained from the devices.

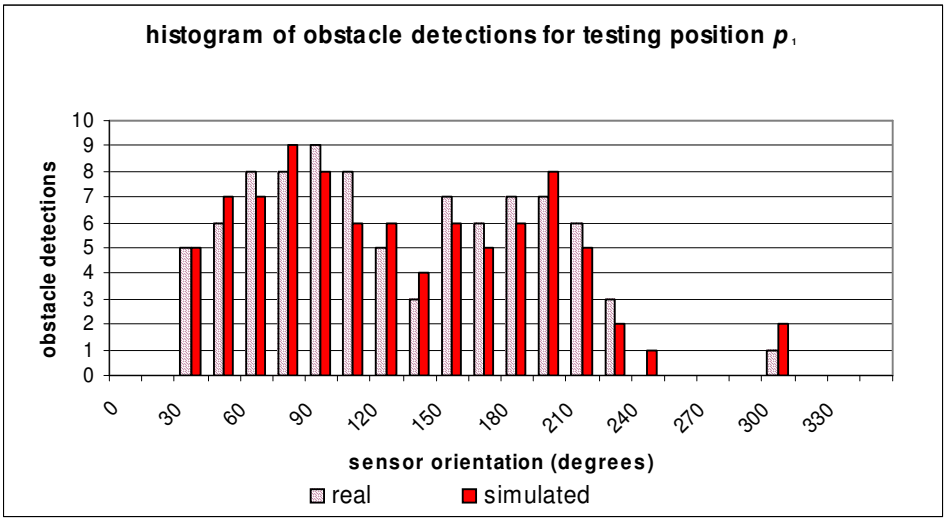


Figure 7.30: Histogram of obstacle detections for the real and the simulated infrared scan in the testing position p_1 (Figure 7.28). The *x* axis shows the orientation of the infrared sensor in degrees. The *y* axis shows the number of obstacle detections. Ten measurements were taken for each orientation of the sensor. For some orientations the detection of obstacle was not possible because the obstacles were out of the infrared sensing range (e.g 270° - 300°).

7.6.6 Summary

The previous sections have presented the simulator that we use to test the BERODE architecture in simulation. The *LOS* and *RF* communication models that were used in our simulations have been described. The simulated *LOS* and *RF* communication models modelled the delays caused by retransmissions in the *MANET* (Mobile *ad hoc* network). The effect of interference was modelled by delaying the messages a random time with a certain probability. In the simulated *LOS* model any obstacle in the direct path of the signal blocked the entire signal. The effect of multi path reflections for the *RF* model was incorporated by adding Gaussian Noise (with mean zero) when there is no *LOS* between transmitter and receiver. We tested our hardware devices in an office environment. Our simulation models were shown to be reasonable and conservative approximations to the data obtained in our experiments from the hardware communication devices.

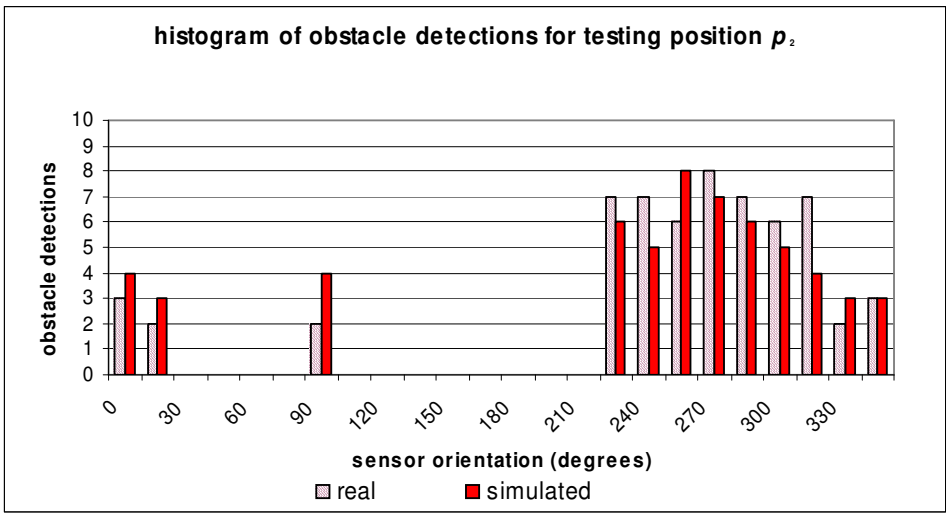


Figure 7.31: Histogram of obstacle detections for the real and the simulated infrared scan in the testing position p_2 (Figure 7.29). The *x* axis shows the orientation of the infrared sensor in degrees. The *y* axis shows the number of obstacle detections. Ten measurements were taken for each orientation of the sensor. For some orientations the detection of obstacle was not possible because the obstacles where out of the infrared sensing range (e.g 105° - 210°.

.

The simulated robots are based on the parameters obtained from experiments with a Koala robot. Our simulated robot incorporates the veering phenomenon observed in real robots due to the uneven wheel traction.

We built an experimental *low cost* sensing platform based on sonars and infrared sensors[2]. The simulation models for the sonar and infrared sensor model the accuracy and obstacle detection reliability observed in tests with hardware devices. We tested the hardware devices using different types and obstacle materials. Our simulation models are based on the worst case observed for the tested obstacles.

The following section described the *low cost* sensing platform. The map building module described in Chapter 6 has been implemented for this platform. Section 7.8 describes the adaptation of this module to cope with the sensor limitations. Section 7.9 presents investigations to validate the map building module for the *low cost* platform.

[2] Our platform described in section 7.7 (page 207) uses Devantech SRF04 sonars and Sharp GPxx infrared sensors.

7.7 Design and Implementation of a Low Cost Sensing Platform

One of the goals of this work is to validate through simulation the suitability of the use of *low cost* robots to build maps of the environment, inspired by the recent success of small *low cost* robot platforms such as RoboMote (Sibley et al., 2002), Swarm-Bots (McLurkin and Smiths, 2004) and MilliBots (Grabowski et al., 1999). Section 4.6 (page 81) presented a discussion on *low cost* sensors and their suitability for map building purposes. It was concluded that sonar and infrared sensors are the most suitable sensors because of their cost and features. Sonar sensors are cheap, small and provide a good sensing range. Map building has been successfully achieved with these sensors. However, mapping cluttered environments with sonars alone has proven to be difficult because of the wide beam-width associated to sonar measurements. The use of infrared sensors can simplify the mapping task without incurring high costs: computational and economic.

This thesis proposes a servo-rotated sensing platform suitable for small size robots. The advantage of using a servo motor is that the environment can be sensed with fewer components being cheaper, smaller and lighter. Cheaper because instead of having rings of sonars and infrared sensors to sense the complete circumference a few of them can be arranged to sweep the environment in an efficient fashion. Fewer components can be arranged to fit smaller spaces more easily. Even more important is the weight factor, because as more components are added to the platform the weight increases therefore the robot has to have more powerful motors. More powerful motors require larger batteries. Additionally more sensor components demand more power consumption because each component drains power even when it is not being used.

The servo motor allows the robot sensors to sweep its circumference without having to rotate its wheel base. This is extremely important because the main source of uncertainty in the robot position originates from rotation movements. Small uncertainty in the orientation causes large uncertainty in the robot position.

Moreover, the servo motor is especially useful to acquire measurements from the infrared sensors. Infrared sensors return narrow subtended angle measurements; as a consequence more measurements are required to measure the entire circumference. The trade-off is that the exploration process is slowed down because the robot has to stop frequently to obtain measurements from its entire circumference. An algorithm to speed up the measurement process by efficiently sweeping the circumference is

described in Section 7.8.1.

The design was decided based on the trade-off between efficiency and cost. In principle the multiplication of sensors does not affect the general functionality of the module. However from a research point of view the development of efficient methods to sense an environment with a few sensors is more interesting.

As discussed in Section 4.6 (page 81) bumpers are the cheapest sensors but they are unsuitable for mapping office environments. On the other hand laser sensors are big and expensive; they are unsuitable for small robots.

The sensing components are mounted on top of a servo motor to enable a 360° field of view. The servomotor rotates 180° therefore at least two sonar and infrared sensors are required to have the 360° field of view. For safety reasons a robot has to sense frequently its circumference to avoid collisions with other robots and the environment. Sonar sensors are useful for these purposes as large areas can be covered by a single sonar. The robot can move forward and backward. We propose that the two sonars are placed in the forward and backward direction to achieve maximum safety. Although only two infrared sensors are necessary to sense the complete circumference we propose the use of three infrared sensors to improve the safety. It is proposed that one of the infrared sensors is placed on the forward direction while other two cover the sides by being rotated 90°. The infrared sensors are less useful than sonars for the safety purposes because of their pointed measurements. However safety is enhanced by covering the sides of the robot rather than only the direction of travel. The platform (Figure 7.32) is then composed of two sonars (Devantech SRF04), three infrared (*IR*) sensors (Sharp GP2Y0A02YK), and three infrared sensors (Sharp GP2D120). The sonars' beam-width is $\beta = 45^\circ$ and their range is 0 to 3m. The infrared sensors have a beam-width of $\beta \cong 2^\circ$ and their range is 0 to 0.3m for the Sharp GP2Y0A02YK and 0.25m to 1.5m for the Sharp GP2D120. These infrared sensors are used in pairs to cover the entire range from 0 to 1.5m. The design of the relative orientation between the sensors minimizes the amount of rotation required to sweep the complete circumference while providing safety with a minimum number of components. The platform is inexpensive with a cost of around £50.

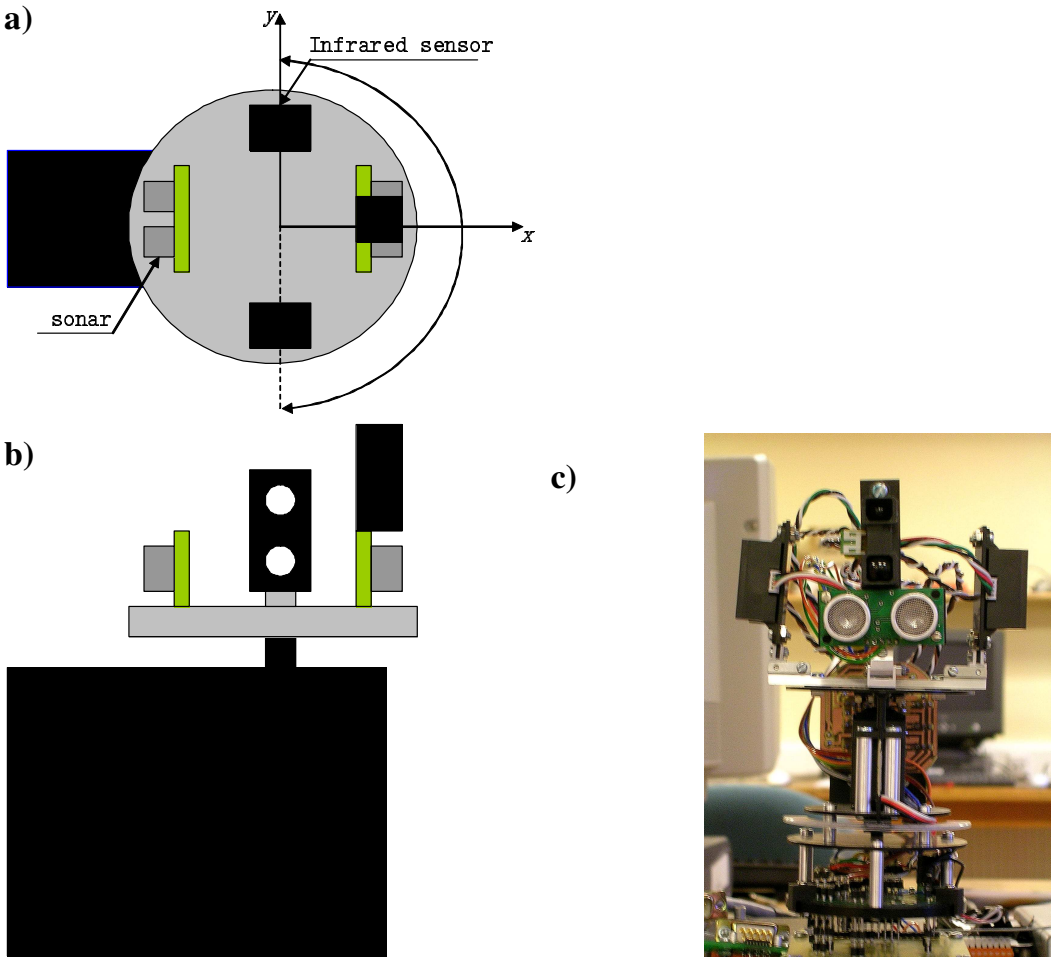


Figure 7.32: *Low cost* sensing platform. a) and b) show schematics of the platform built, c) shows a snapshot of the platform built. The sensors are mounted on top of a servo motor to acquire the entire circumference.

7.8 Implementation of the Location Algorithm for the Low Cost Sensing Platform

The previous chapter described the Map Building Module used by the robots to build a feature map representation of the environment as they traverse it. The feature map is updated using an Extended Kalman Filter (*EKF*) (Smith et al., 1988). The *EKF* produces an estimate (along with its uncertainty) of the position of the features and the robot. The environment is represented as a set of line and point features. Features are extracted from raw sensor data. The extracted features are used to update the *EKF*.

The proposed *low cost* platform integrates features extracted from sonar and infrared sensors in separate processes. The uncertainty and the measurement range from

the sonar and infrared measurements differ. A model to extract features from low level combined measurements is not practical. An infrared scanning algorithm that efficiently uses the infrared sensors is required because of the shorter range and narrower beam-width of these sensors compared to the sonar sensors. The execution of this algorithm depends on the number of features extracted by sonar. In the feature extraction process the robot pauses frequently to sense its 360° circumference by rotating the sensor platform in steps of $\Delta\theta_1$ degrees. If necessary, a second sweep in steps $\Delta\theta_2$ ($\Delta\theta_1 >> \Delta\theta_2$) to acquire more infrared information is executed. When the Sonar Feature Extraction process has extracted enough features (more than 2 for the current implementation) the second sweep is not executed to speed up the exploration.

Line features are extracted from the infrared measurements (Figure 7.33). Point features are not extracted from the infrared sensors because point features are not sensed as such, but are higher abstractions from edges.

Line and Point features have 2 *DOF* (Degrees of freedom). As explained in Section 6.3.3. (page 144) the extraction of features from raw sonar measurements cannot be achieved from a single position because each sonar measurement has only one *DOF*, the distance to the nearest surface. The bearing is only known within some range defined by the beam-width, typically 25° to 45°. However features can be extracted by grouping measurements from multiple close positions.

The extraction of features from sonar measurements is achieved by maintaining a temporal window of measurements. This window stores the sonar measurements of the most recent sensing positions. A sensing position is a position where the robot paused to take measurements. A *RANSAC* (Random Sample Consensus) algorithm is then used to extract the features. The basic idea in *RANSAC* is to allow probable precision to be traded against computational time, by randomly selecting pairs of readings, generating a hypothesis, and then counting how many of the remaining measurements agree with the hypothesis. If the number of measurements that agree with the hypothesis is above a user defined threshold the hypothesis is accepted.

7.8.1 Areas of Interest for Infrared Sensors

For the available sensing platform the range of the infrared sensors is considerably smaller than that of the sonar sensor, and the error in the infrared sensors is exponential with respect to the distance. For these reasons it's sensible to prune the use of the infrared sensors by determining at which orientations sonar suggests a possible line.

A fine sweep of that orientation range is performed if it is within *IR* range. This saves sensing time and computational resources. The proposed Data Collection and Perceptual Grouping process (Figure 6.1, page 142) is then adapted as shown in Figure 7.33 to efficiently collect the data.

Sonar readings are used to determine where lines may be within the infrared range. Ideally, sonar readings return the closest obstacle from a β beam-width ($\beta \cong 45^\circ$ for the available sonars) which determines the sweeping ranges for the infrared sensors. Algorithm 3 uses two arrays as inputs: $sonar_r$ and $sonar_\theta$ that contain the distance and angles for the set of sonar readings taken at the current position. The parameter $\Delta\theta_{infrared}$ is the step size to be used when sweeping the space with the infrared sensors ($\Delta\theta_{infrared} = 5^\circ$ in the current implementation), $DIST_{MAX}$ is the distance threshold to consider a certain orientation within the range of the infrared sensor ($DIST_{MAX}$=0.8 in the current implementation). Inf_{array} stores the shortest distance for each angle with incremental orientation $\Delta\theta_{infrared}$. The range(s) are extracted based on the array Inf_{array} by identifying consecutive measurements below $DIST_{MAX}$. Once the sweeping ranges have been determined the second sweep with a finer resolution to acquire infrared measurements for these ranges is executed (Figure 7.33).

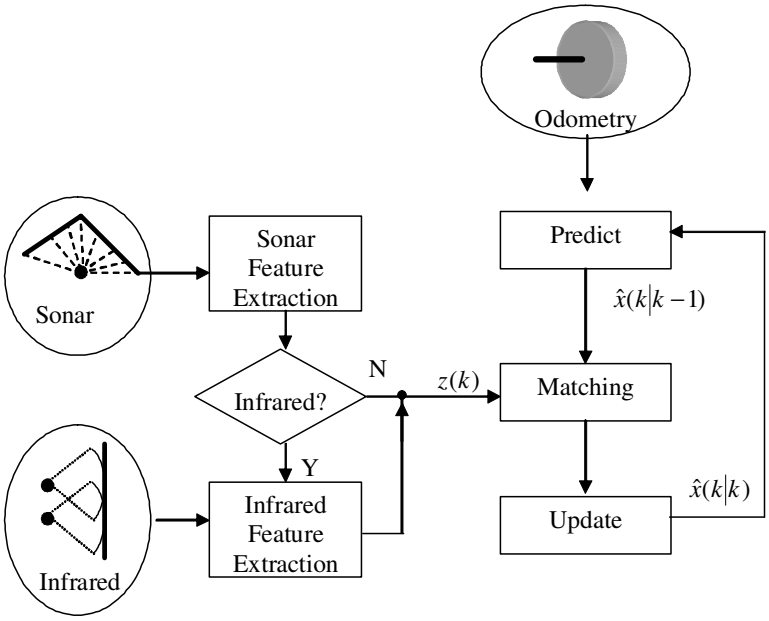


Figure 7.33: Feature extraction for the low cost sensing platform. Line and point features are extracted from the sonar sensor. If necessary additional line features are extracted from the infrared sensors. The extracted features are used to update the *EKF*.

Algorithm 3 Algorithm to obtain the areas of interest for the infrared sensors based on the sonar measurements. The infrared sensors have a shorter sensing range than the sonar sensors. The areas of interest are the areas for which is possible to extract information using the infrared sensors.

$Inf_{array} \Leftarrow$ array of size $360/\Delta\theta_{inf}$
for $i = 0$ to $360/\Delta\theta_{inf}$ **do**
  $Inf_{array}[i] = DIST_{MAX}$
end for
$k = size\ of\ (sonar_{\theta})$
for $i = 0$ to k **do**
  for $j = -\beta/2$ to $\beta/2$ step $360/\Delta\theta_{inf}$ **do**
    $index = \lfloor (sonar_{\theta}[i] + j)/360/\Delta\theta_{inf} \rfloor$
    if $sonar_r[i] < Inf_{array}[index] < DIST_{MAX}$ **then**
      $Inf_{array}[index] \Leftarrow sonar_r$
    end if
  end for
end for

7.8.2 Feature Extraction for Infrared Sensors

Information obtained from the infrared sensors is represented by an array of measurements with incremental orientation taken from a single sensing position. Initially a split and merge approach was implemented (Tardos and Castellanos, 1999) as this is a common approach to extract lines from laser range data. Due to the *IR* error being exponential with respect to distance the method either failed to extract lines or the estimates were poor. Instead a *RANSAC* approach was used to generate hypotheses about the possible lines. *RANSAC* is executed as described in Section 6.3.3 (page 144); nevertheless due to occlusion, measurements from close orientations are more likely to produce good hypotheses about the localization of the line features. A hypothesis is generated by randomly selecting a reading in the array and then getting another reading with a small offset in the array of readings. The offset is drawn from a random Gaussian distribution ($\mu = 0, \sigma = dist$). Small values of *dist* will more likely return close pairs in the array than larger values (for the experiments *dist*=5).

After the line features have been extracted, the continuity of the lines is tested to find gaps in measurements associated with each line. A gap is defined as the distance in array positions between the two sequential associated measurements. If the gap is above a threshold the line is split. A final process to remove short line segments (below

0.3m in the implementation) is executed.

7.9 Investigations

We have conducted two investigations to validate the map building module proposed in the previous chapter using the *low cost* platform and its simulation model. The purpose of the investigation with the real robot is to validate the proposed hardware platform and to show the advantages of the incorporation of infrared sensors. The purpose of the investigation with the simulation model is to show the advantages of the incorporation of infrared sensors in a larger trajectory and assess its suitability for multi-robot purposes. The investigations were carried out in the corridors and rooms of the IPAB (Institute of Perception Action and Behaviour) institute. For the simulation we generated a simulation version of this environment. In the investigations the robot starts off with zero uncertainty at the initial position.

In the real world investigation the robot was driven following preplanned paths along the corridors of the institute, running close to one wall ($\cong 0.5$m) so that infrared information could be used. The robot travels 0.1m between sets of readings. The incremental angles were $\Delta\theta_{sonar} = 30^\circ$ and $\Delta\theta_{infrared} = 5^\circ$ for the sonar and infrared processes respectively. The information was stored for calibration purposes. In the simulation the robot followed a fixed path using the same parameterization as that of the experiment with the real robot.

7.9.1 Investigation 1: The Corridor

The first investigation was carried out along one of the corridors of the IPAB institute. The robot followed a preplanned path of 35m. Figure 7.34 shows the environment used in this investigation and the path that the robot followed. The robot travelled a distance of 0.1m between sensing locations. As previously explained a sensing location is a location where the robot pauses to take sonar and infrared measurements which are used in the feature extraction process. The distance between sensing locations was determined from preliminary experiments. For closer distances than 0.1m the feature extraction process extracted much fewer features. This is due to the wide beam of the sonar measurements. Most of the readings originate from a few obstacle locations when the sensing positions are very close. Due to the closeness of the positions the uncertainty in the bearing of the obstacles is reduced less. It is then more difficult to

extract the features. For larger distances the feature extraction process only extracted large line segments (above 2m). The map contained less line features and more point features. The additional points were part of lines that were not extracted because of the lack of evidence. It is desirable to have more line features because they describe the environment better. Moreover the additional points produce less compact maps.

Several temporal window sizes were tested for the multiple viewpoint feature extraction process (Section 6.3.3, page 144). A size of 15 was found to be the most adequate; for larger window sizes the number of features extracted was almost identical and the computational cost was higher while for smaller window sizes there was not enough information to extract line features most of the time. For instance for a window size of 20 only 2% more features were extracted, with larger line segments most of the time, but the computational cost was around 20% higher. For a window size of 10 the number of line features dropped to around 40% and around 30% more points were detected. All the additional extracted points belonged to lines without enough evidence for a line feature, but enough evidence about one or more points that belonged to the physical line.

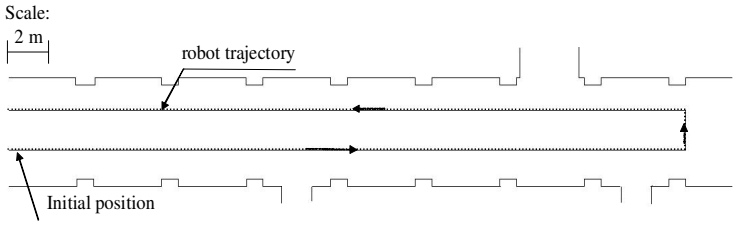
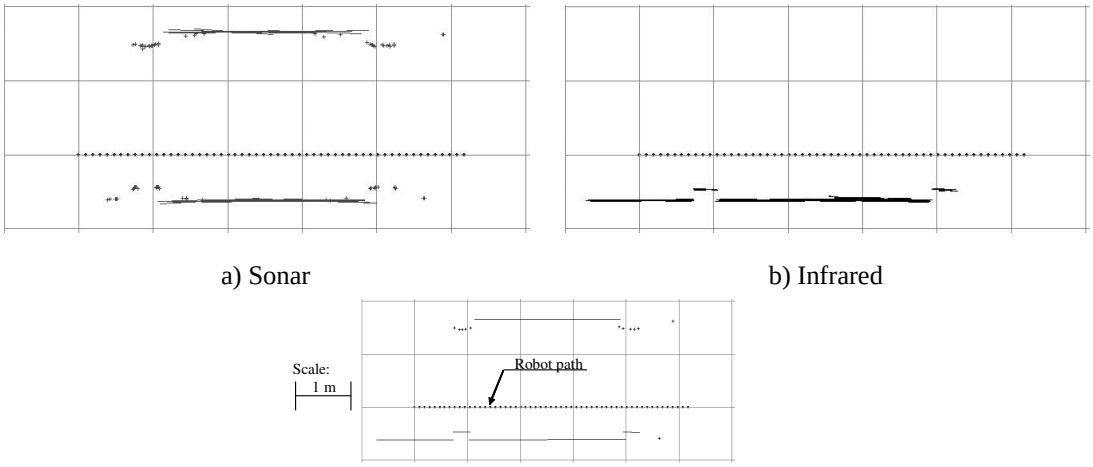


Figure 7.34: Corridor from IPAB institute. The trajectory followed by the robot is shown.

The investigation revealed that some of the extracted features appeared only once or twice, being the results of noisy data and typically were point features (Figure 7.35 (a)). Adding these features to the *EKF* increased the computational cost without benefit. Therefore we decided to add the features to the map only when they have been matched more than a certain number of times. New unmatched features are added and remain in a temporal buffer for a small time waiting to be matched by subsequent measurements. Once enough matches have been achieved they are added to the feature map. In the implementation the number of matches necessary to add a feature to the map was two for a point and three for a line feature. The points remain in the buffer for two iterations of sensor reading pauses, while the lines remain for three. Points remain in the buffer for fewer iterations because they were often the product of noise in cluttered environments. This was not the case for lines.

Figure 7.35 illustrates the feature extraction process at the start of the trial along the corridor. In Figure 7.35(a) and (b) all the extracted features before data association are shown. It is possible to extract short line segments with the infrared sensors from the side of the corridor which is closer to the robot, whereas the sonar can extract larger line segments from both sides of the corridor. The shorter segments are columns in the wall and are detected as point features by the sonar. Figure 7.35(c) shows the results after the matching process is carried out; points that belong to lines are removed.

As mentioned before, in previous approaches for sonar based sensing, infrared sensors were not used. As a consequence there is no evidence about the benefit of combining the two types of sensors in the literature.



a) Sonar b) Infrared

c) Merged features

Figure 7.35: Feature extraction process for the investigation with the real robot. a) Features extracted from the sonar sensor, b) features extracted from the infrared sensor, c) features extracted after the merging and matching processes.

Features	**Total**	**Line**	**Point**
Sonar approach	68	15	53
Combined approach	48	33	15

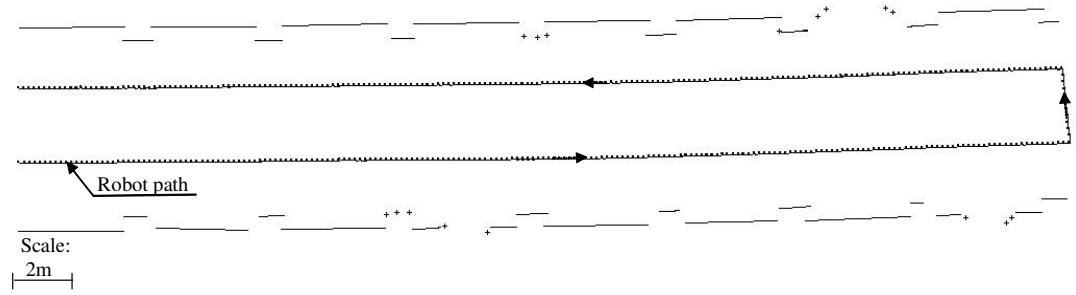
Table 7.7: Number of line and point features in the map after the robot followed the preplanned path from Figure 7.34.

Figure 7.36 shows the feature maps obtained for the combined approach (infrared and sonar) and for sonar. It is observed that the combined mapping is straighter, with more lines and fewer points (Table 7.8), giving a more accurate and compact map than

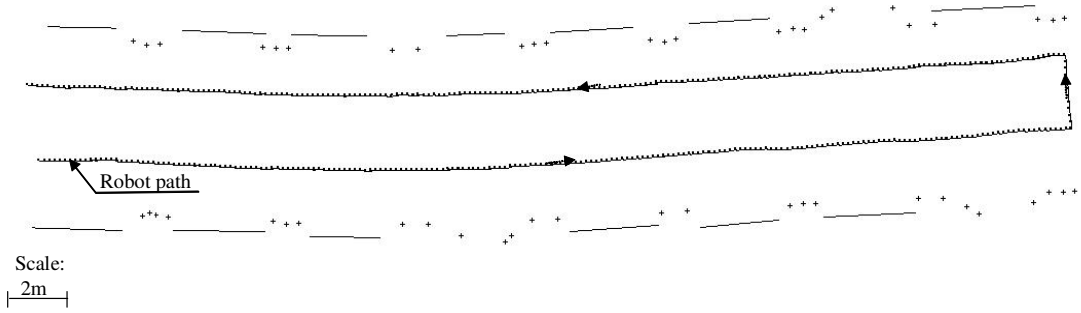
the map built with the sonar alone approach; we hypothesized that this is due to the additional observations of line features (42.4% more observations). To test this hypothesis we conducted the same investigation in simulation. The errors in the position and orientation along the trajectory were recorded. Figure 7.37 shows the plotting of the errors for the first 10m of the trajectory. It is observed that for the sonar alone approach there are three regions where the errors peak. These regions are shown in boxes on the graph and their correspondences in the simulated corridor are shown in Figure 7.38. They are places where it is difficult to extract features from the sonar sensors because the simulated robot is located parallel to the columns of the environment. The columns have a width of 0.35m and cannot be extracted as line features from sonar measurements. Most of the readings that correspond to the columns are obtained from the positions parallel to the columns. Due to the wide beam-width of the sonar measurements it is often not possible to obtain column-corresponding measurements because the measurement corresponds to other closer features inside the beam-width. In 0.35m it is not possible with the sensor sampling strategy proposed to take more than five associated readings associated with a specific line feature. In preliminary tests with the real robot we observed that the reliable extraction of line features required that at least ten sonar readings be associated with a specific line feature.

Moreover, the simulated robot has to travel around 0.6m before it is able to extract features because the robot has to gather enough evidence from multiple positions to be able to extract features. This leads to less accurate position and orientation estimations from the beginning. Due to the additional information in the combined approach environmental features are observed from the beginning of the trajectory and short segment lines are successfully extracted in the areas where the sonar alone is unable to extract features.

As the robot moves further from its initial location in the environment the error in the orientation of the new line features is larger. Once the robot starts to return to the initial location features are re-observed and the uncertainty stabilises within a range. The uncertainty of features that are more distant to the origin is larger than that of features close to the origin. The *EKF* applies large corrections to estimate the location of features with large uncertainty. The curved shape (Figure 7.36(b)) of the corridor is the consequence of the large uncertainty in these features. This curved shaped is pronounced in the sonar alone approach whereas in the combined approach the map is only slightly bent.



a) Combined (Sonar and Infrared)



b) Sonar

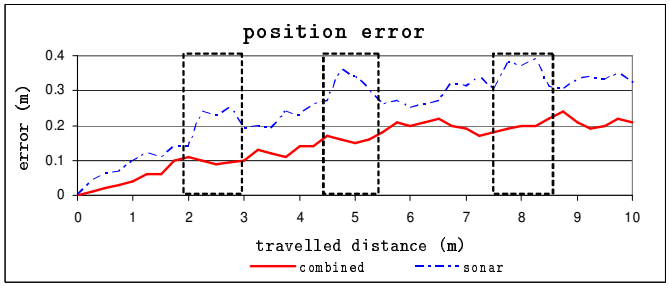
Figure 7.36: Feature map from the experimental corridor of Figure 5.7 using a) combined and b) sonar sensors.

7.9.2 Simulation: Closing a loop around the Institute

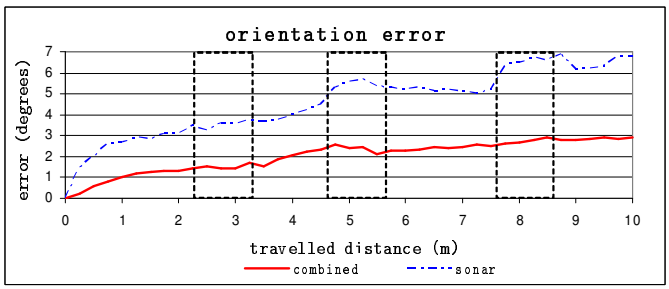
As discussed in Section 3.7 (page 48) the goal of this thesis is not the proposal of a new *SLAM* approach that copes with the problems associated with large cyclic environments and the well known scaling problems of the use of an absolute reference map for the *EKF*. Nevertheless, it is important to test the approach in a larger simulated environment that contains a medium sized loop. Loops are common in office environments. Thus it is important to test the capability of the approach to build a consistent map in this case and assess the computational cost of the proposed approach. As in the previous investigation the simulated robot follows a preplanned path and starts with zero uncertainty at the initial location.

The travelled distance in the simulated environment was 60m, where after 35m the simulated robot came back to the original location and followed a different trajectory (Figure 7.39). Figure 7.40 shows the maps built using firstly the sonar alone and secondly the combined approaches.

For the sonar alone approach the simulated robot was not able to close the loop. When the simulated robot came back to the start location the matching process failed



a) Position error



b) Orientation error

Figure 7.37: Accumulated error in a) position and b) orientation for the preplanned trajectory of a simulated robot. The rectangular dotted boxes highlight regions where the errors peak because the extraction of features was difficult.

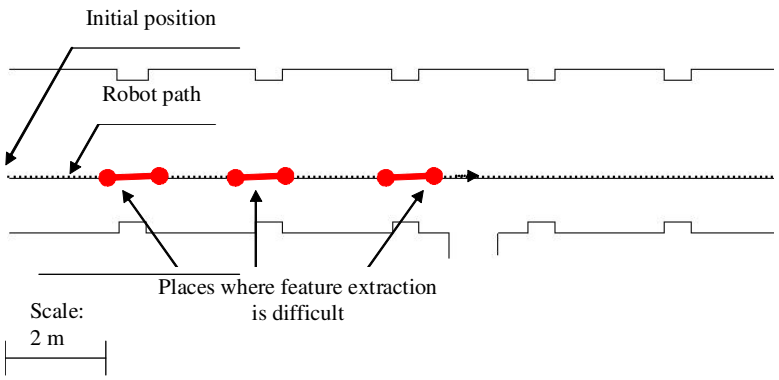


Figure 7.38: Corridor of the IPAB institute highlighting the places where the extraction of features is difficult for the sonar sensor. The path that the simulated robot has to follow is presented.

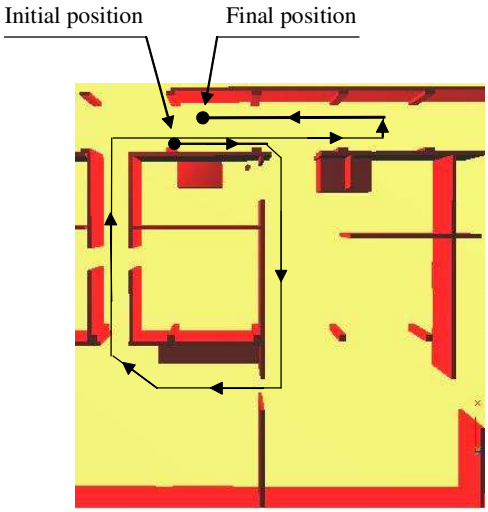


Figure 7.39: Simulated environment used in the second investigation. The preplanned path followed by the robot is shown.

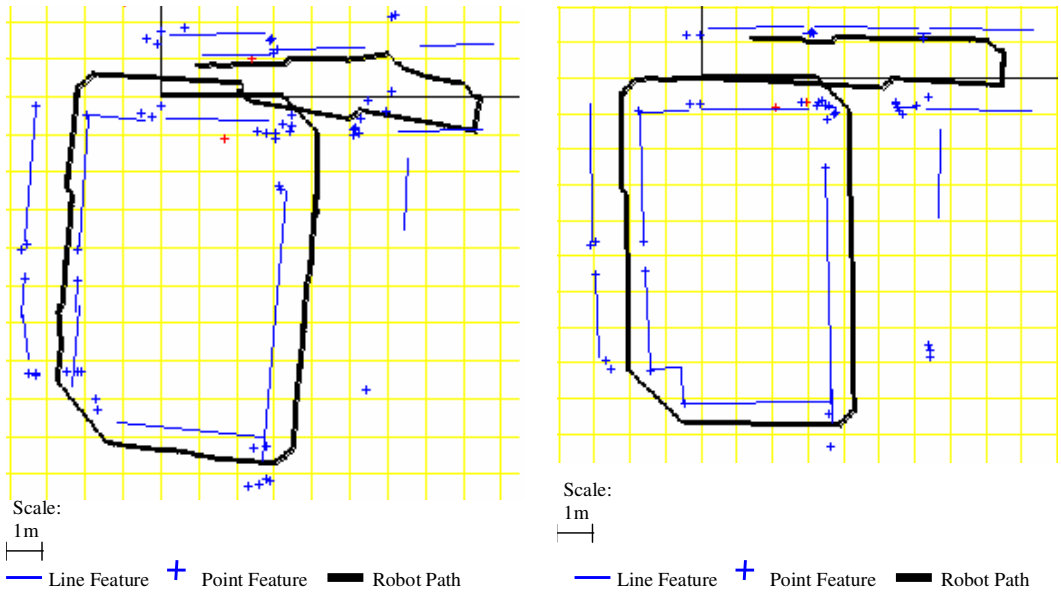
and the simulated robot started to add new overlapping features to the map. The resulting map is inconsistent. A robot navigating autonomously for exploration purposes should be able to build consistent maps. Inconsistent maps are at best inefficient for navigation, and at worst can prevent a robot returning home.

The path followed by the simulated robot and the parameters used in the comparison of the combined and sonar approaches are the same. For the sonar approach presumably more exploration would reduce the uncertainty and allow the closure of the loop. However for the purposes of comparison it is easier to compare the two approaches for the same path.

Useful reduction in uncertainty with the sonar and infrared approach allows the closure of exploration loops in less time. The important thing is reducing the uncertainty enough to allow identification of revisited features. Uncertainty can be reduced by taking more readings or better readings. With given sensors more readings can be obtained with more detailed exploration. We have taken the approach of adding another low cost sensor. In some cases, as in the example exploration loop, this will allow loop closure within a given path where sonar alone would not. More generally, with the additional IR sensors consistent maps can be made more efficiently, and with a given exploration strategy larger maps can be closed.

In Figure 7.40 we can observe that for the combined approach due to the additional information provided by the infrared sensors the simulated robot is able to properly close the loop, and maintain a consistent and more compact representation of the envi-

ronment.



a)sonar alone b) combined

Figure 7.40: Map built by the simulated robot while following a closed loop trajectory using the sonar alone (a) and the combined approaches (b). It is observed that the sonar alone approach is unable to close the loop having as a consequence an inconsistent map.

Figure 7.41 presents the number of features extracted by each approach as the simulated robot moves through the environment. It is observed that when the simulated robot has travelled 45m the sonar alone approach keeps adding new features to the map while the combined approach re-observes previously mapped features and only the new features are added to the map.

The processing time related to the distance travelled is shown in Figure 7.42; this time is not only a function of the number of features in the map, it depends also on the number of observations realized at every viewpoint. This can be observed in the initial part of the trajectory where the combined approach has a higher computational cost due to the additional observations. The higher cost is noticeable only in the first 15m because of the larger difference between the number of features in the combined and sonar alone approaches. From Figures 7.41 and 7.42 it can be observed that after the simulated robot return to the initial location (35 m) the processing time for the sonar alone approach keeps growing and becomes larger than that of the combined approach. This is because the data association process fails to identify re-observed features. This process fails because of the large uncertainty in the simulated robot and

feature locations for the sonar alone approach. These re-observed features are then added as new features and the processing time of the *EKF* algorithm increases. The computational cost of the algorithm is $O(ij^2)$ where j is the number of features and i is the number of observations. The quadratic order is only noticeable for the sonar approach because the number of features is relatively small. The inability to close the loop not only creates inconsistent maps, but also causes higher computational costs. These two problems are critical for multi-robot implementations.

In summary, although the additional observations generated for the combined approach (from the infrared sensors) will not provide a general solution to the problem of closing exploration loops it allows the simulated robots to operate in larger environments while still building consistent maps. For the testing purposes of this research in simulated multi-robot scenarios this allows experimentation in medium sized simulated environments.

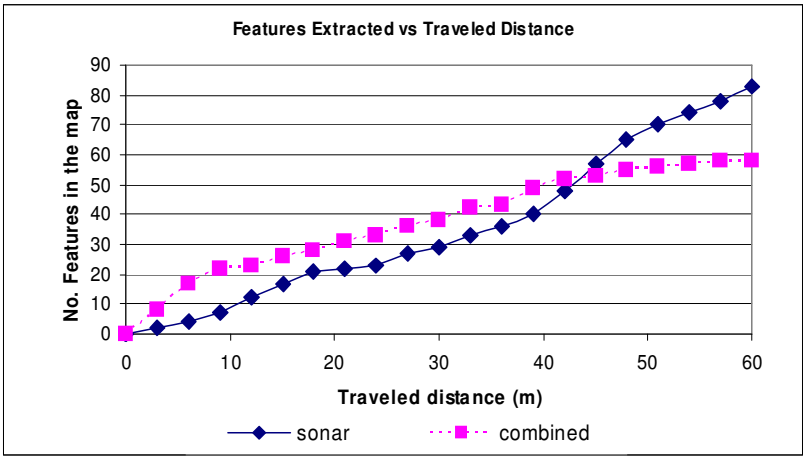


Figure 7.41: Features added to the map using the sonar and combined approaches as a function of the travelled distance by the simulated robot that follows the trajectory from Figure 5.12.

7.10 Conclusions

This chapter has presented the simulator that we used to test our BERODE (BEhavioural ROle DEcentralized) architecture and described the simulation models for the robot's sensors, odometry and communication devices. We tested our hardware devices in an office environment. We presented the *RF* and *LOS* simulation models used to test the BERODE architecture. Our simulation models were shown to be reasonable and conservative approximations to the data obtained in our experiments from

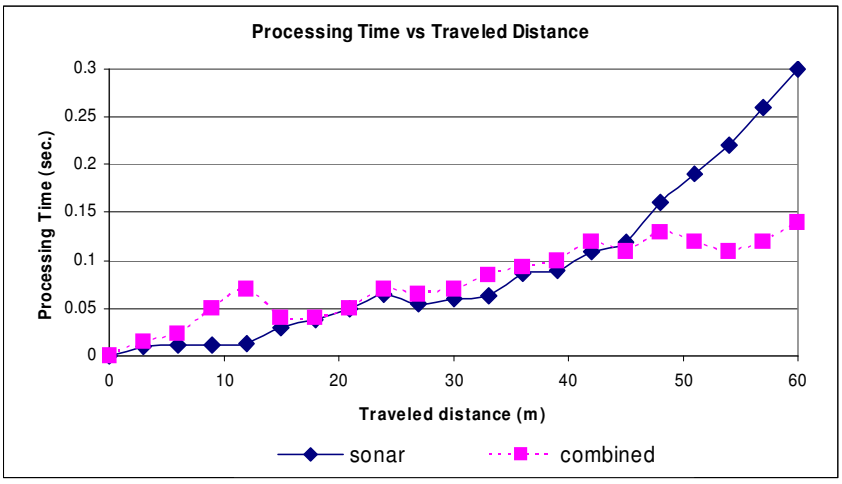


Figure 7.42: Processing time using the sonar and combined approaches as a function of the travelled distance by the simulated robot that follows the trajectory

the hardware sensor devices.

The map building module was implemented in a novel *low cost* sensing platform based on sonar and infrared sensors. This platform serves as basis for the simulations with multiple robots in the following chapters. The sensing platform is mounted on top of a servo motor. The robot senses its circumference by rotating the platform. The map building module was adapted to efficiently gather information. A technique to combine the information from sonar and infrared sensors was described and validated in two investigations.

In the first investigation a real robot was used to explore a corridor while in the second one a simulated robot was used to compare the proposed combined approach used in the platform against sonar alone information. In the first investigation the combined approach built a better representation than the sonar alone approach in that it generated a straighter representation of the corridor, with more lines and fewer points, giving a more accurate and compact map representation of the environment to that of sonar alone sensing. The sonar alone approach generated a more curved representation of the corridor. This was expected because the combined approach integrates more observations than sonar alone sensing. As proven by Newman (1999) as more observations are added to the *EKF* the map converges to the true positions of the features.

In the second investigation the simulated robot had to close a loop (the revisiting problem). The environment was successfully mapped by the combined approach and failed with the sonar alone approach. Presumably for the sonar approach more exploration would allow the closure of the loop but more time is required then to explore the

entire space. Increasing the size of the temporal window does not solve this problem because almost no additional features are detected and the computational cost is incremented. It is argued that the additional observed features from the infrared sensors were the key factor to the successful closure of the exploration loop. It is concluded that the combination of sonar and infrared produces better estimates of the robot's position than sonar alone; moreover the stochastic maps are more compact and accurate.

Chapter 8

Implementation and Improvements to BERODE

8.1 Introduction

This chapter presents the implementation of the BERODE (BEhavioural ROle DEcentralized) Architecture for the *LOS* (line of sight) and *RF* (Radio Frequency) communication models. Simulations to analyze the performance of BERODE for these models are presented in this chapter. To improve the realism of our simulations relevant measured aspects of the robot sensors and communication devices have been included in the simulated models. This is explained in detail in Chapter 7. Throughout this chapter we will refer to our simulated robots as simbots.

BERODE is based on behavioural roles. These roles reactively adapt to the dynamic conditions of the communication network formed by the simbots as they explore an environment. The communication network is maintained as a fully connected network by creating and updating an MST (Minimum Spanning Tree) control network. The MST control network is a subnetwork of the communication network containing only the necessary links to maintain the communication network connected. The roles generate reactive plans that maintain the simbot's direct connections in the MST control network. These plans use a predictive model to predict the signal quality at nearby positions. The predictions are generated by using the simbots' current map. This contains the estimated positions of the nearby simbots. The predictions are used for two purposes: network maintenance and exploration. For the purpose of network maintenance predictions are used to determine the nearby position where the overall signal quality (OSQ) is the best. For the purpose of exploration the signal quality for nearby

unexplored areas is predicted to determine if the area is safe to explore. The simbots tend to explore areas that have a safe level.

Section 8.2 describes an improved implementation of the mapping module described in Chapter 6. The improved module has additional mechanisms to cope with problems present in multi-simbot scenarios (e.g. detecting simbots as obstacles).

Most if not all of the current communication technologies are based on *RF* and *LOS* technologies. For this reason, we propose the use of the *RF* and *LOS* communication models to assess BERODE. We expect to identify and fix problems related to the implementation of BERODE for these technologies. Section 8.3 describes the implementation of BERODE for the *LOS* model. In this model the simbots can not measure the strength of the signals. The Euclidean distance is used to estimate the strength of the signals. Section 8.4 describes the predictive model for *LOS* communication which has been adapted to cope with the on/off nature of *LOS* signals. The adapted model predicts the distances to the closest obstacles for a signal. These distances are the perpendicular distances to the closest obstacle within the communication link for both sides of the link. These distances are incorporated in the reactive planner to direct the simbots towards open areas where the communication is less likely to be lost.

Section 8.5 presents the implementation of BERODE for the *RF* model. In the *RF* model part of the signal is absorbed by the obstacles. In this model it is assumed that the simbots can measure the strength of the signals. This value is typically available in *RF* technologies and is known as the *RSSL* (Received Signal Strength Level). In this implementation the proposed predictive model predicts the signal quality based on the *RSSL*.

Section 8.6 presents an initial simulation to assess the performance of BERODE for different strategies. The strategies differ in the parameters of the *virtual spring* model used to maintain the simbot network connected. The simulation revealed poor performance of BERODE when exploration was inefficient or temporarily inactive because no simbot was exploring. Section 8.7 describes these situations and presents three procedures to address the poor performance. The procedures are an exchange role mechanism, a random MST checking mechanism and a tendering mechanism. In the role exchange mechanism an Explorer simbot determines when there is a more suitable simbot to explore an area. In this case the simbots exchange their roles and the exploration is optimized. The communication delays can generate situations where there are no Explorer simbots left in the network. Time is wasted when there are no Explorer simbots in the network because the exploration is halted until a simbot transi-

tions again to the Explorer role. A random MST checking mechanism is used to detect when there are no Explorer simbots in the network. If there are no Explorer simbots the tendering mechanism is triggered. The tendering mechanism is used to determine which candidate Explorer simbot to choose. The enhanced architecture is referred to as BERODE-2. Section 8.8 compares the performance of BERODE and BERODE-2. The comparison is based in the exploration time and the communication cost. Section 8.9 presents the summary of the conclusions from the simulations.

8.2 The Map Building Module in Multi-Simbot Scenarios

The Map Building Module for BERODE implements an Extended Kalman Filter (*EKF*). The *EKF* is a feature based representation of the environment that estimates the location and uncertainty of the features and the simbot using the sensor measurements of the simbot as it explores the environment.

In BERODE the features are either internal or external. Internal features are extracted by the simbot from its raw sensor data. A *feature management* process extracts, segments and associates the features. The extracted features are then used to update the *EKF* and improve the estimates of simbot and feature location. The simbots start off at known locations and share a common global coordinate system, which is the initial position of the simbot with the smallest ID number. Having a common frame of reference allows the exchange of extracted features (Section 6.3.8, page 155). In the current implementation a simbot that observes another simbot does not use this observation to improve the estimation of its location.

External features are features that a simbot receives from other simbots in the simbot network. These features are handled by the Map Interface Module (Section 5.8, page 119). There are two types of External features: Local and Global. Local features are used to aid navigation but they are not integrated into the local feature map. Global features are integrated to the feature map. These features are integrated into the feature map using the same association process used to match internal features (Section 6.3.8, page 155).

The simulation models for the robot's sensors, odometry and communication devices are based on parameters obtained in our experiments from the hardware sensor devices (Chapter 7). Preliminary simulations show that for more than a few simbots

it was difficult to obtain good position estimates. The maps have large inconsistencies that make it difficult to assess the performance of BERODE. The main causes of inconsistencies are:

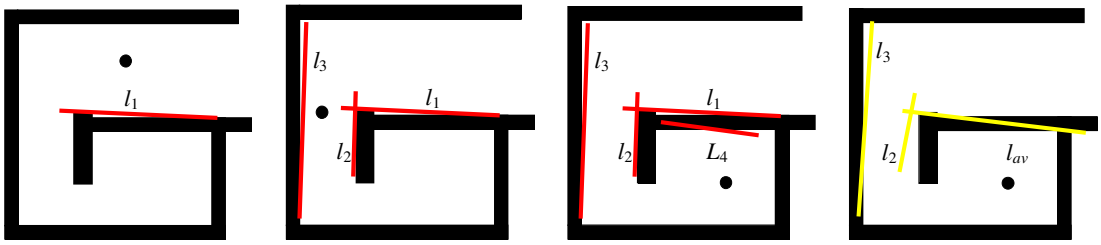


Figure 8.1: Typical feature extraction for a simbot (black circle). a) Robot extracts line l_1, b) the simbot moves and extracts lines l_2 and l_3, c) the robot moves and extracts line l_4, l_1 and l_4 are merged forming line l_{av}.

- Corridors have a tendency to be curved.
- Other simbots were detected as point features by a simbot.
- The lengths of line features are considerably overestimated.

The first cause does not present a problem on its own. Corridors tend to be almost straight and the maps are consistent. The problem arises when a simbot observes a wall from its two sides. The line feature does not have a thickness property. The wall is observed as two separate lines. Typically these lines are matched and the *EKF* filter is updated. The updated representation is distorted because the matching process fails. Frequently the outcome is that the near corridors tend to have a curved shape. This situation is presented in Figure 8.1 where the simbot initially extracts line l_1, the simbot keeps exploring the space and later it extracts lines l_2 and l_3 (Figure 8.1(b)). The simbot moves to another position and extracts line l_4 (Figure 8.1(c)). The lines l_1 and l_4 are matched and the filter is updated. The line l_{av} is the merged line (Figure 8.1(d)). It is observed that the orientation of the lines l_2 and l_{av} have a large error because of the incorrect matching. The increase in error for l_3 is small because it is more distant from l_{av} than l_2. The result of incorrect matching is that the neighbourhood of features is distorted and frequently curved.

The second cause of inconsistencies is when a simbot is detected by another simbot as an obstacle. This happens when a simbot remains nearly stationary for long periods

of time. This is frequently the case for Maintainer simbots that have to maintain several constraints. These simbots tend to remain in close positions while improving the communication safety for their constraints. The problem arises when the simbots are detected as obstacles in the middle of a corridor or in the entrance of a room. In these cases other simbots may have to take long detours or in the worst case a simbot that is inside the room may become trapped.

The exact detection of the extremes of a line feature is difficult to achieve with sonar sensors. When the estimated length of a line feature is much larger than the real length the map tends to contain inconsistencies in places such as narrow openings. Figure 8.1(a) presents an example of this problem. It can be observed that the simbot has to take a detour because of the incorrect measure. In the worst case scenario the simbot may not detect the opening to the room.

Another issue that arises when mapping an environment with multiple simbots is that walls are detected as several small segments. These small segments do not have an overlap when they are first extracted. The segments are added to the representation as two line features. As the exploration continues the segments are updated and eventually they overlap. Although these repeated features do not generate inconsistencies, the computational cost of the *EKF* is increased. Moreover the estimation is improved when the repeated features are associated.

To address these discovered problems we propose the use of the following map management mechanisms:

- *A priori* structural knowledge: The simbots use known structural tendencies (e.g. parallel walls) to improve the estimation of their locations. Every time a new feature is detected in the environment one structural constraint is applied (Section 6.3.10, page 156).

- View point checking: The proposed representation for the line feature incorporates view point information. The viewpoints are relative positions to the feature at which the feature was observed. By using the viewpoints it can be verified if two matched lines are observed from the same side. If the two lines are observed from opposite sides the match is dropped. This avoids the incorrect matching of features.

- Simbot filter mechanism: Point features originated from the simbots being detected as obstacles are filtered. The filter checks if the point is too close to the reported position of a simbot.

- Periodic extreme updating for line features: Based on *a priori* knowledge about perpendicular structure when two lines are perpendicular and their extremes are in close positions the extremes are updated. Frequently the lines are shortened.

- Periodic check of repeated features: Line features can be checked to determine if they represent the same structure (e.g. wall). The features are associated using the same association process from Section 6.3.8 (page 155). The two features are merged and the repeated states are removed from the *EKF* (Leonard et al., 2002).

The last two mechanisms are expensive in computational terms. The processes are run periodically. Recall that in the Map Building Module the line features are ordered with respect to their distance from the initial position. This is used to speed up the periodic check of repeated features. A feature is checked against close lines with respect to the ordered list. In the case of the extreme updating mechanism it is not possible to speed up the process. Each feature has to be checked against the rest of the features to verify for compatible extremes. The proposed mechanisms fixed the problems of inconsistencies in the map. The maps generated by the simbots still contain small metric differences. However, these differences are small enough to avoid situations where the simbots could take different decisions. Chapter 8 presents simulations to analyze the consistency of the maps built by the simbots.

In BERODE the simbots periodically transmit their extracted features. These features are stored in the Map Interface Module (Section 5.8, page 119). This module stores, matches and updates the recent observations of features since the last transmission period. To simplify the implementation we decided that the Map Building Module would not integrate the corrections realized by the *EKF* to these features. We expected that the omission of these corrections would not cause failures in the matching of the features. Only small differences in the parameters of the features and their uncertainty were expected.

The feature maps are used by the Planning Module to generate plans based on predictions. The feature maps are projected in a probabilistic grid map. The module predicts the signal quality of its communication constraints for nearby positions relative to the current simbot position. The simbot moves towards the position with the best OSQ (overall signal quality). The OSQ is a position where all the communication constraints are within range.

Inconsistencies in the maps generate situations where there is no OSQ because

none of the nearby positions and the simbot current position is within range of all its communication constraints based on the estimation from the projected grid map. The conflict arises when there is communication with all the constraints but the estimation disagrees. This conflict was only observed in *LOS* technologies because of the on/off nature of the signal. This conflict was resolved using a conservative approach. The simbot remains in its position for a certain time waiting for the conflict to be resolved by the map building module (e.g. the extremes of the lines are updated reducing the overestimation of the length of the lines). If the conflict is not resolved the simbot executes small movements in the direction of one of its constraints. The constraint is selected randomly. When the connection with one of the constraints is lost the simbot transitions to the Recoverer role and backtracks its movements. The proposed solution resolves the conflict having as drawback the possibility of losing communication with a communication constraint. More sophisticated solutions to resolve the conflict are discussed in the conclusion of this thesis and remain as one of our future research interests.

8.3 Implementation of BERODE for the LOS model

In the simulation model for *LOS* communication any obstacle in the direct path of the signal blocks the entire signal. In the *LOS* model the simbots are not able to measure the strength of the signals. To reduce the loss of communication episodes due to the on/off nature of *LOS* signals we propose to use the Euclidean distance between the robots as the signal strength metric.

It is argued that as the Euclidian distance between the transmitter and the receiver increases the signal is more likely to be blocked by the obstacles within the direct path of the signal. For this reason the threshold parameters are defined as a function of the Euclidian distance. These thresholds are used for the activation of the communication constraints, the *virtual force* model, and to predict the safety level for unexplored areas.

Communication constraints activate depending on the current safety level. The equation (Eq. 4.1, page 64) to determine the safety level L_i for a simbot i with control connections $n_i(z) = \{k_{i,1}, k_{i,2}, ..., k_{i,z-1}, k_{i,z}\}$ is redefined as

$$L_i = \begin{cases} Safe & if \quad \forall j \left(\kappa_{i,j} < \sigma_{safe}\right) \\ Precautionary & if \quad \forall j \left(\kappa_{i,j} \leq \sigma_{prec}\right) \wedge \exists j \ \left(\kappa_{i,j} \geq \sigma_{safe}\right) \\ Unsafe & if \quad \exists j \left(\kappa_{i,j} > \sigma_{prec}\right) \end{cases} \tag{8.1}$$

where $\kappa_{i,j}$ is the Euclidian distance between simbots R_i and R_j and the parameters σ_{safe} and σ_{prec} are the communication thresholds that represent the desired Euclidian distance (signal quality) between pairs of simbots. The active set of communication constraints ϕ_i is then redefined as

$$\phi_i = \begin{cases} \{\} & Safe \\ \{x | \sigma_{prec} \geq \kappa_{i,j} \geq \sigma_{safe}\} & Precautionary \\ \{x | \kappa_{i,j} > \sigma_{prec}\} & Unsafe \end{cases} \quad (8.2)$$

Equations 8.1 and 8.2 invert the inequality signs from equations 4.1 and 4.2 (page 64) to reflect the previously explained argument.

8.4 The Predictive Model for LOS Communication

The predictive model is used by the Planning and Network Manager Modules of the simbots. The model predicts the signal quality for nearby positions to the simbot position. The model estimates the attenuation due to the obstacles in the direct path of the signal. In this model any obstacle in the direct path blocks the signal. The predicted signal quality is then an on/off value for the signals. To avoid the frequent loss of communication between the connections in the MST control network the modules have been improved with respect to the original description from Sections 5.6 (page 102) and 5.9 (page 121). The improvements are described in the following subsections.

8.4.1 Planning Module

The predictive plans are based on the attractive/repulsive forces exerted by the simbots' communication constraints and the obstacles (Section 5.6.1, page 105). The forces are a function of the discomfort distance and the roles between the pair of simbots that form the constraint. In the *LOS* model the discomfort distance is the distance between the Euclidian positions of the simbots. The module tests nearby positions to the simbot position and moves to the position with the best OSQ (overall signal quality). The OSQ is the position where the energy generated by the forces is minimized.

The attractive/repulsive force model for *LOS* was adapted to avoid the frequent loss of communications. A couple of repulsive obstacle potentials were added to the original attractive/repulsive *virtual spring* model. These repulsive potentials are different from the repulsive potential from the closest obstacle.

The repulsive obstacle potentials are the perpendicular potentials generated by the closest obstacle within the communication link for both sides of the communication link. This will have the effect of moving the simbot to a central position between *LOS* occlusions on either side. The repulsive potentials are calculated using the same Yukawa potential function (Cohen et al., 1977) used to calculate the repulsive potential from the obstacles (Section 5.6.1.2, page 108). The distances are calculated on the projected grid map using a bounded ray tracing algorithm for computational efficiency. After a certain distance it is assumed that the potential is insignificant. This distance is user defined. Recalling from Eq. 5.5 (page 108) that the energy E for a position (x,y) is defined as

$$E(x,y) = \left| \sum_{i=1}^{n_c} \overline{U_i(x,y)} + \overline{U_{obs}(x,y)} \right| \tag{8.3}$$

where n_c is the total number of communication constraints for the simbot R_j, $\overline{U_i(x,y)}$ is the attractive/repulsive energy vector for the i^{th} communication constraint and $\overline{U_{obs}(x,y)}$ is the repulsive potential vector from the closest obstacle. $\overline{U_i(x,y)}$ is then redefined as

$$\overline{U_i(x,y)} = \overline{U_{j,i}} + \overline{Y_{j,i}^{+}} + \overline{Y_{j,i}^{-}} \tag{8.4}$$

where $\overline{U_{j,i}}$ is the attractive/repulsive vector (Eq. 5.10). $\overline{Y_{j,i}^{+}}$ and $\overline{Y_{j,i}^{-}}$ are the repulsive Yukawa potentials for the perpendicular orientations defined as

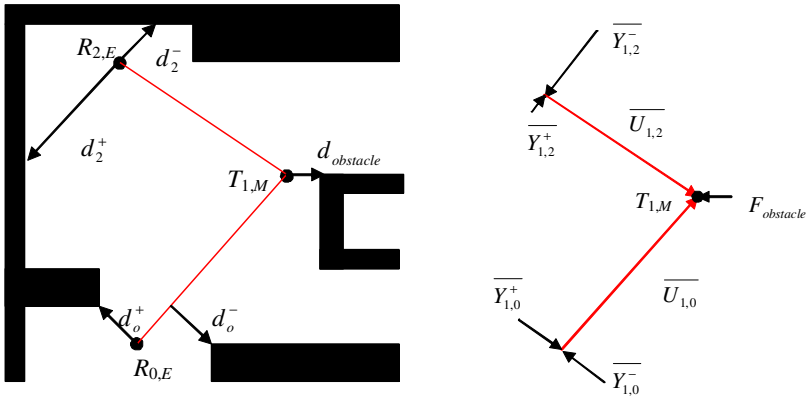
$$\overline{Y_{j,i}^{+}} = \left| Y(d_{i,j}^{+}) \right| \cos(\theta_{j,i} + \pi/2)\hat{x} + \left| Y(d_{i,j}^{+}) \right| \sin(\theta_{j,i} + \pi/2)\hat{y} \tag{8.5}$$

$$\overline{Y_{j,i}^{-}} = \left| Y(d_{i,j}^{-}) \right| \cos(\theta_{j,i} - \pi/2)\hat{x} + \left| Y(d_{i,j}^{-}) \right| \sin(\theta_{j,i} - \pi/2)\hat{y} \tag{8.6}$$

where $\theta_{j,i}$ is the orientation between the simbots R_j and R_i. $d_{i,j}^{+}$ and $d_{i,j}^{-}$ are the distances to the closest obstacles in the perpendicular orientations. $Y(d_{i,j}^{+})$ and $Y(d_{i,j}^{-})$ are calculated using the Yukawa Potential from Eq. 5.15 (page 112) for the closest obstacles in both directions.

In preliminary simulations we observed that the addition of the repulsive potentials reduced the loss of communication episodes by 30%. The simbots tend to move to positions where the signal is further from possible obstructions. Afterwards the simbots generate a new predictive plan based in the updated simbot position and projected grid map. Figure 8.2 presents an example of the Planning Module for simbot R_1 with test position $T_{1,M}$ and communication constraints $R_{0,E}$ and $R_{2,E}$. $\overline{U_{1,0}}$ and $\overline{U_{1,2}}$ are repulsive forces from the *virtual spring* model. d_0^{+} and d_0^{-} are the closest distances for $R_{0,E}$. d_2^{+} and d_2^{-} are the closest distances for $R_{2,E}$. $d_{obstacle}$ is the distance for the closest obstacle

for the testing position. $F_{obstacle}$ is the force generated by $d_{obstacle}$ which is calculated using the Yukawa Potential (Eq. 5.15). It is observed that the repulsive potentials for $R_{0,E}$ $\overline{Y^+_{1,0}}$ and $\overline{Y^-_{1,0}}$ cancel each other whereas for $R_{2,E}$ the difference between the repulsive potentials is large increasing the total energy stored. This position is unlikely to be the position with the best OSQ (overall signal quality).



a) The probabilistic grid map b) Energy diagram for the test position

Figure 8.2: Example of the predictive planning model for a simbot with testing position T_1 with *LOS* communication model. a) The closest distances from the obstacles for the repulsive potential forces. b) The force diagram for the test position T_1.

8.4.2 Network Manager Module

The Network Manager periodically verifies that the Explorer simbot is maintained in the safe level as it moves towards its current goal (Section 5.9, page 121). The manager modifies the simbot control connections to improve the safety level. The safe level is achieved when the Explorer simbot has at least one direct connection for which the signal quality is above the safe threshold σ_{safe_exp} (Section 4.3.3, page 66).

Because of the on/off nature in *LOS* communication the signal quality for the direct connections is not an available value. The signal quality is emulated using the Euclidian distance for the direct connections. As explained in Section 8.4.1 the inequalities that define (Figure 4.3, page 65) the safe $(\max(\varepsilon_{i,1},...,\varepsilon_{i,z}) > \sigma_{safe})$ and precautionary $(\kappa_{i,current} > \sigma_{prec})$ levels are inverted to reflect our argument that closer connections are less likely to be obstructed by obstacles in the environment. The module selects the direct connection for which the Euclidian distance is the smallest. This value does not give any information about the status of the connections therefore is not useful for avoiding the loss of communications. A value that predicts the status of the connection

in the near future can be used to avoid the loss of communications and improve the safety level. We propose the use of the distance to the closest obstacle for each direct connection as this predictive value. The module is adapted to select the direct connection with the maximum distance to the obstacles that is inside the safety level. The obstacle distance for the direct connection is the perpendicular distance to the closest obstacle to the line between the robot and the direct connection. The selected direct connection has to have a distance to the obstacles larger than a user defined threshold σ_{wall} to avoid the loss of communication.

8.4.3 Summary of the LOS Implementation

This section presented the *LOS* model. In this model any obstacle in the direct path of the signal blocks the entire signal. The implementations of the predictive models of BERODE for this model were described. The predictive models are used in the Planning and Network Manager Modules. In the *LOS* implementation the distances to the closest obstacles for the communication links are calculated. These distances are the perpendicular distances from both sides of the communication link. The obstacle distances are used in the predictive models to direct the simbots toward open areas where communication is less likely to be lost.

8.5 The Predictive Model for RF Communication

The predictive model is used by the Planning Module of the simbots. The model predicts the signal quality for a position using Rappaport's model (Section 7.3.1, page 166). Rappaport's model is used by the Simulation Manager to simulate the signal strength measurements. The Simulation Manager has knowledge of the positions of the simbots and the material of the obstacles. The manager calculates the strength of the signals and transmits these values to the simbots. This simulates the signal strength measurements that would be obtained from the *RF* device in the hardware implementation. In Rappaport's model the attenuation of the signal is a function of the distance between the robots, the number of obstacles between them and the attenuation for each type of obstacle. The attenuation for the obstacle depends on the material and the density of the obstacle.

The simbots estimate their position and the positions of the features that they have extracted from sensor measurements using an *EKF* (Section 8.2). The simbots trans-

mit beacon signals that contain their estimated position. The signal strength of the beacon signals is a measure that is calculated by the Simulation Manager. The Planning Module of the simbots uses the feature map and the simbots positions (obtained from the beacon signals) to predict the signal quality for close positions. Because the simbots do not have knowledge about the material of the obstacles they estimate the average attenuation assuming that all the obstacles are made of the same material. The attenuation average is the average of the attenuation per grid cell (*AGS*) for recent beacon signals received by a simbot. This assumption is reasonable because the simbots estimate the signal quality only for close positions with respect to their position. For instance a simbot receiving a beacon signal that goes through a wall uses the signal quality of the beacon signal to estimate the signal quality for positions for which the signal traverses the same wall.

The *AGS* is measured in dB/m. The attenuation per grid cell (*AGS*) for a signal is calculated on the projected probabilistic grid map as follows

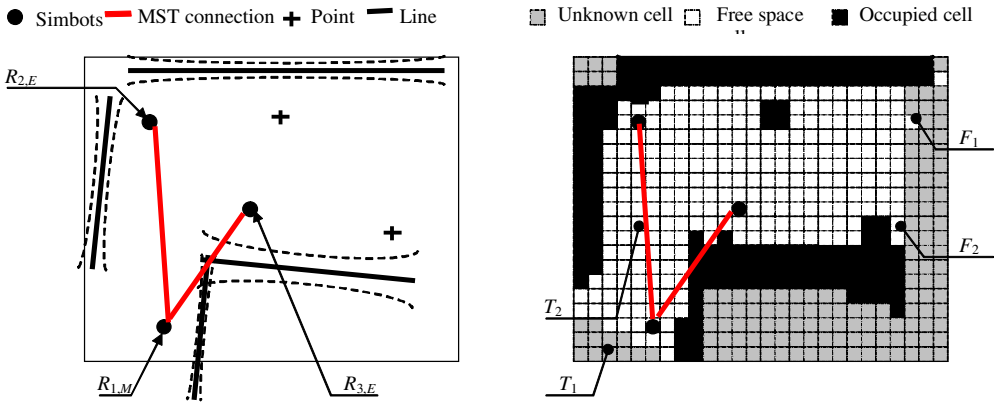
$$AGS = \frac{1}{P_{cells} * GR}(PL(d) - FSL(d)) \tag{8.7}$$

where P_{cells} is the number of grid cells that are labelled as either occupied or unknown space in the direct path of the signal in the projected probabilistic grid map. $PL(d)$ is the measured signal quality and $FSL(d)$ is the free space loss. $PL(d)$ is calculated by the Simulation Manager using the Eq. 7.3 (page 167) and sent to the simbots to simulate the signal strength measurements. $FSL(d)$ is calculated by the simbots using Eq. 7.2 (page 166). *GR* is the resolution of the grid. *AGS* is then the average value for all the obstacles in the direct path of the signal. The *AGS* value is estimated only when $PL(d) > FSL(d)$ and $P_{cells} > 0$. In some occasions $PL(d) \leq FSL(d)$ because the simulated signal quality measurements incorporate the effect of multi path propagation by adding random Gaussian noise.

As previously mentioned the predictions of the signal quality for close positions are done using Rappaport's model (Section 7.3.1, page 166). This model is defined as

$$PL(d) = FSL(d) + \sum_i AGS_i \tag{8.8}$$

where AGS_i is the attenuation for the i^{th} obstacle grid cell. As previously explained the simbots assume that the grid cells are made of the same material and estimate the attenuation using Eq. 8.7. $FSL(d)$ is calculated using Eq. 7.2 (page 166). The predicted $PL(d)$ value is used to determine the signal quality, expressed as a percentage. The percentage of signal quality is calculated using Eq 7.3 (page 167).



a) Feature map b) Probabilistic grid map

Figure 8.3: An example of the predictive model for *RF* communication. a) The feature map for the simbot network, b) the projected probabilistic grid map from the feature map.

The predictive planner uses the predicted signal quality to test close positions to the simbot position and generate a reactive plan to the position with the best OSQ (Section 5.6.1, page 105). The exploration planner uses the predicted signal quality to test the safety of the *frontiers*[1] and determine their *hierarchy* level (Section 5.6.2, page 113). Figure 8.3 presents an example of the predictive model for a simbot network of three simbots. $R_{1,M}$ is a Maintainer simbot with testing positions T_1 and T_2 which are randomly chosen by the predictive model. The occupied partitions for T_1 are zero from $R_{2,E}$ and three from $R_{3,E}$. For T_2 the partitions are zero for both $R_{2,E}$ and $R_{3,E}$. $R_{3,E}$ is an Explorer simbot with frontiers F_1 and F_2. The partitions are zero for F_1 and two for F_2.

8.5.1 Prioritization of Connections in the RF Model

In the *RF* model the simbots use the *RSSL* to calculate the MST control network when the QS predictive heuristic is used (Section 4.4.1). Simbots in the Explorer role have a Network Manager Module (Section 5.8, page 119). The module monitors and tries to improve the safety level for the simbot. The module periodically verifies that the Explorer simbot is maintained in the safe level as it moves towards its current goal. The module modifies the simbot control connection to improve the safety level.

Initial simulations revealed that the calculated MST control network frequently contain a control connection where the pair of simbots that formed the connection

[1] A *frontier* is a portion of free space that is adjacent to unknown space in the probabilistic grid representation of the environment.

did not have *LOS*. Also the Network Manager of an Explorer simbot occasionally modified the simbot control connection to a connection where there was no *LOS*. Certain environment configurations generate local minima for control connections with no *LOS*. Occasionally a simbot with a non *LOS* control connection was trapped in a local minimum having in consequence a slow down in the exploration process. Figure 8.4 presents an example of a simbot network with control connections with no *LOS*. In Figure 8.4(a) simbots $R_{4,P}$ and $R_{3,M}$ have a non *LOS* control connection. Simbot $R_{4,P}$ is a Pusher simbot that is trying to move towards $R_{3,M}$ to accelerate the movement in the exploration direction. It is observed that the simbot will get trapped in a local minimum when it reaches the wall. The exploration for the simbot network is slowed down because $R_{4,P}$ and $R_{3,M}$ remain static. The rest of the simbots could then only move in their nearby areas to avoid the loss of connections.

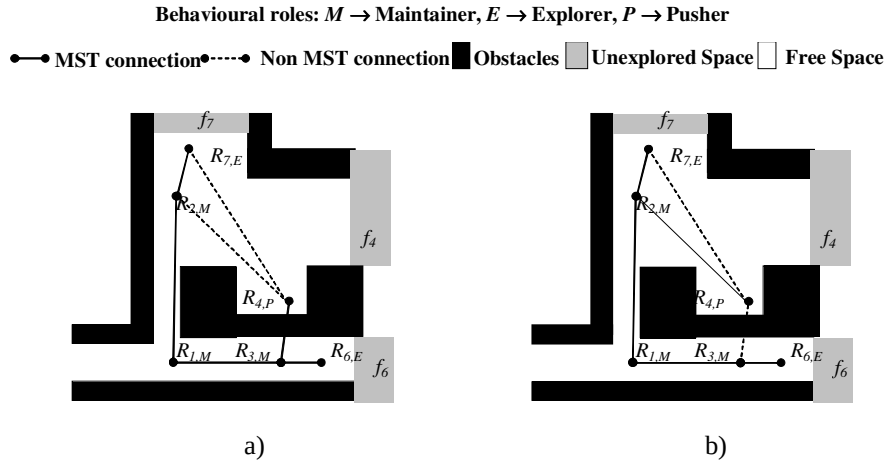


Figure 8.4: An example of a simbot network with control connections with no *LOS*. a) Simbots $R_{4,P}$ and $R_{3,M}$ have a non *LOS* control connection. b) The network is modified to have all the control connections with *LOS*.

To reduce the likelihood of these situations in the implementation of the *RF* model the connections are prioritized. The simbots calculate the partial or complete MST control network using only the connections with *LOS*. The simbots determine if a connection has *LOS* using their projected grid map. If it is not possible to calculate an MST control network where all connections have *LOS* then the network is calculated using all the connections. Figure 8.4(b) shows the MST control network that is built when connections are prioritized whereas Figure 8.4(a) shows the network built when connections are not prioritized. The Network Manager Module also implements the prioritization of the connections. The prioritization algorithms are presented in Ap-

pendix D.

8.5.2 Summary of the Implementation

The previous sections have presented the improved Map Building Module and the implementation of BERODE for the *LOS* and *RF* models. The Map Building Module has been improved for its use in multi-simbot scenarios. The Map Building Module incorporates additional map managing mechanisms to cope with problems present in multi-simbot scenarios.

In the *LOS* model the simbots are not assumed to be able to measure the signal quality while in the *RF* model the simbots are able to measure the signal quality. BERODE was implemented based on these assumptions for the *LOS* and *RF* models. We described the implementation of the predictive models of BERODE for these models. The predictive models are used in the Planning and Network Manager Modules of the simbots. In the *LOS* implementation the distances to the closest obstacles for the communication links are calculated. These distances are used in the predictive models to direct the simbots toward open areas where communication is less likely to be lost. The implemented predictive modules for the *RF* communication predict the signal quality for nearby positions to the simbot position. The prediction is based on the attenuation due to obstacles in the direct path of the signal.

The following sections present the initial simulation using the implemented *RF* and *LOS* models. Based on the results of this simulation we proposed an improved version of BERODE. Simulations to assess the improved version are presented. The chapter concludes with a summary of the results from the simulations.

8.6 Initial Simulation: String Model Parameterization

The goal of the initial simulation is to compare the performance of the simbot network for different strategies with respect to communication safety. We expect to identify a trade-off between the communication safety and the exploration time. The exploration time is the time that the simbots require to build a complete map of the environment[2]. For illustration purposes we propose three strategies. The strategies vary in their parameterization. The behaviour of each strategy can be obtained with similar parame-

[2] A map is considered to be complete once it is projected into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a used defined threshold. In our implementation this value is 0.3m.

terizations. The three proposed strategies are: Conservative, Neutral and Risky. These strategies reflect a desire to maintain a certain signal quality for the simbots. Conservative strategies try to maintain higher levels of signal quality than Risky strategies. Risky strategies on the other hand are expected to complete the exploration in less time because the simbots are less constrained to maintain a compact network. These strategies use different parameters for the activation of the communication constraints, the predicted safety level at unexplored areas, and the attractive/repulsive forces model used to maintain the simbots within communication range.

The settings for the simulation were:

		Exploration Strategy		
Parameters	**Units**	**Conservative**	**Neutral**	**Risky**
σ_{safe_exp}	m	1.5	2.5	3.5
σ_{safe}	m	0.1	0.1	0.1
σ_{prec}	m	2.5	3.5	4.5
σ_{max}	m	5.0	7.0	9.0
σ_{rep_pusher}	m	2.0	2.5	3.0
σ_{far}	m	1.0	1.5	2.0
σ_{close}	m	0.6	0.8	1.0
$\sigma_{far\_explorer}$	m	0.3	0.4	0.5
$\sigma_{collision}$	m	0.2	0.2	0.2
$t_{expiration}$	*sec.*	100	50	50
$\kappa_{compresion}$	n.a.	2	2	2
$\kappa_{stretching}$	n.a.	1	1	1
σ_{wall}	m	0.2	0.2	0.1

Table 8.1: Parameters for the *LOS* communication model for three exploration strategies.

- Simbots transmit their beacon signal every second. This signal contains the estimated simbot position and its uncertainty calculated with the *EKF*.
- The delay for the reception of a message from a direct connection was assumed to be 0.1 sec.
- For the *RF* implementation a random 5% noise is added to the signal strength measurements.
- The process of sensing an area (sensing step) using the *low cost* platform takes 1.8 seconds. Simbots transmit their extracted features in the local and the global

neighbourhood every 5 and 20 sensing steps respectively. Sensing steps are preferred over periodical updates because simbots that maintain the simbot network remain stationary for periods of time. During these periods the environment is not sensed.

The strategies were compared using the *LOS* and *RF* communication models. Table 8.1 and Table 8.2 present the parameters for the *LOS* and *RF* communication models respectively. σ_{safe_exp} is the parameter of the Network Manager Module (Section 5.9, page 121). σ_{safe} and σ_{prec} are the parameters for the activation of the communication constraints (Section 4.3.2, page 65). σ_{max},σ_{rep_pusher}, σ_{far}, σ_{close}, $\sigma_{far\_explorer}$, $\sigma_{collision}$, $\kappa_{compresion}$ and $\kappa_{stretching}$ are the parameters of the *virtual spring* model (Section 5.6.1.2, page 108). $t_{expiration}$ is the minimum role time for the Pusher simbots (Section 4.5, page 77). σ_{wall} is the distance to the obstacles used in the Network Manager Module for the *LOS* implementation (Section 8.4.1). The range for the *LOS* model was assumed to be infinite; however the simbots are very unlikely to be apart more than 3m because the safe threshold (σ_{safe}) is set to 2.5m (Table 8.1). When the simbots are at a distance larger than the safe threshold they will try to move towards each other. For the *RF* model the transmission power was assumed to be 60dB, which allows communicating to a distance of up to 5m in free space.

Each exploration strategy was tested in the small and medium environments from Figure 7.1 (page 164) using three heuristics to calculate the MST control network for both communication models (*LOS* and *RF*). Section 4.4.1 (page 70) described the heuristics.

The Connectivity and Position heuristics were used in both models. The Pos. Predictive heuristic was used for the *LOS* model and the QS Predictive heuristic was used for the *RF* model. The size of the team of simbots was $n = 1,...8$ in the small environment, and $n = 1,..,12$ in the medium environment. For each team the size of the local network was varied from $k = 1,...,n$. Each combination of strategy, team size (n) and local network was run for 10 successful trials. Failed trials occurred because the robots got stuck and failed to finish the exploration. The success rate at finishing the exploration in the trials was 95.3%.

Two metrics are used to compare the strategies: the exploration time and the percent of time fully connected. The percent of time is the percentage of the time that the simbot network remains fully connected.

Figure 8.5 shows the exploration time for the *LOS* model in the medium environment for the three proposed strategies. The graph shows the average exploration

		Exploration Strategy		
Parameters	**Units**	**Conservative**	**Neutral**	**Risky**
σ_{safe_exp}	dB %	40	30	20
σ_{safe}	dB %	80	80	80
σ_{prec}	dB %	15	10	5
σ_{max}	dB %	20	15	10
σ_{rep_pusher}	dB %	20	15	10
σ_{far}	dB %	40	30	20
σ_{close}	dB %	60	50	40
$\sigma_{far\_explorer}$	dB %	75	65	55
$\sigma_{collision}$	dB %	95	95	95
$t_{expiration}$	*sec.*	100	50	50
$\kappa_{compresion}$	n.a.	2	2	2
$\kappa_{stretching}$	n.a.	1	1	1

Table 8.2: Parameters for the *RF* communication model for three exploration strategies.

time for a number of simbots n for all the values of $k = 1, ..., n$. It is observed that, as expected the Risky strategy takes less time to build the maps than the Neutral and Conservative strategies. For team sizes larger than eight simbots the exploration time for the Neutral and Risky strategies is very similar. For simbot teams smaller than seven simbots there is a linear decrease in the exploration time with respect to the number of simbots. For all the strategies there is a logarithmic decrease in the exploration time with respect to the number of simbots. Similar trends were found for the *LOS* model in the small size environment and for the *RF* model in the small and medium size environments.

Figure 8.6 and Figure 8.7 show the percent of time that the network is fully connected in the medium size environments for the *LOS* and *RF* model respectively using the three proposed strategies. It is observed that for all the strategies the percentage of time that the network is fully connected decreases as the number of simbots is increased. For the Conservative and Neutral strategies the decrease trend is similar. For the Risky strategy the decrease is much larger as the number of simbots increases than for the other two strategies. We attribute this to the fact that larger networks have more communication constraints making more difficult the task of keeping the network fully connected. Extensive simulations to analyze this situation are presented in Chapter 9.

By comparing Figure 8.6 and Figure 8.7 it is observed that communication networks based on *RF* technologies are easier to keep fully connected than networks based

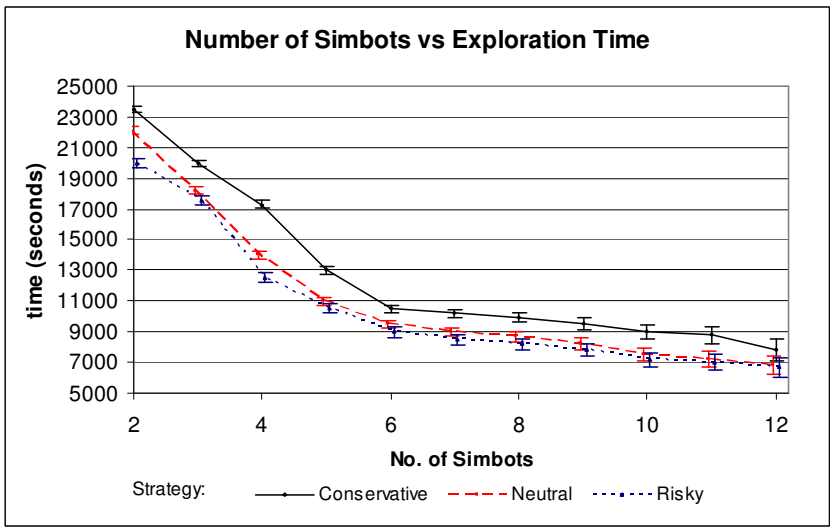


Figure 8.5: Exploration time in the medium environment for the *LOS* model using three strategies: Conservative, Neutral and Risky. The vertical lines show the standard deviation. This deviation is much larger for larger teams compared to that of smaller teams.

on *LOS* technologies. This is attributed to the on/off nature of the *LOS* technologies.

It is concluded that there is a trade-off between exploration time and the percentage of time that the network is fully connected. The Conservative strategy maintains the simbot network fully connected most of the time (95% in average). The Risky strategy is more efficient in terms of the exploration time. Based on the two metrics it is concluded that the Neutral strategy is the best strategy because:

- Exploration time: The Neutral strategy takes only 3% more time in average to build the complete map than the Risky strategy. Moreover, for more than eight simbots (in the medium size environment) the performance is similar.

- Percent of time fully connected: The Neutral strategy has a similar trend to that of the Conservative strategy.

Small adjustments to the parameters of the Neutral strategy do not improve the performance meaningfully. The Neutral strategy will serve as the basis for the rest of the simulations presented in this thesis.

It is observed that above a certain number of simbots (8 simbots for the medium size environment) the exploration time does not tend to decrease as fast as for small number of simbots. Previous research (Burgard et al., 2006) in simulated multi-robot exploration has found a logarithmic decrease in the exploration time as the number

of robots used in the exploration is increased. Our simulations do show this type of decrease. We observed that there was a much larger variance in the results with large teams ($n > 8$ for the medium size environments). Since most often this larger variance was due to outliers with unusually poor performance we suspect that the performance should be better for these large simbot teams. Most of the trials where the performance was poor were trials with small local network sizes ($k < 4$).

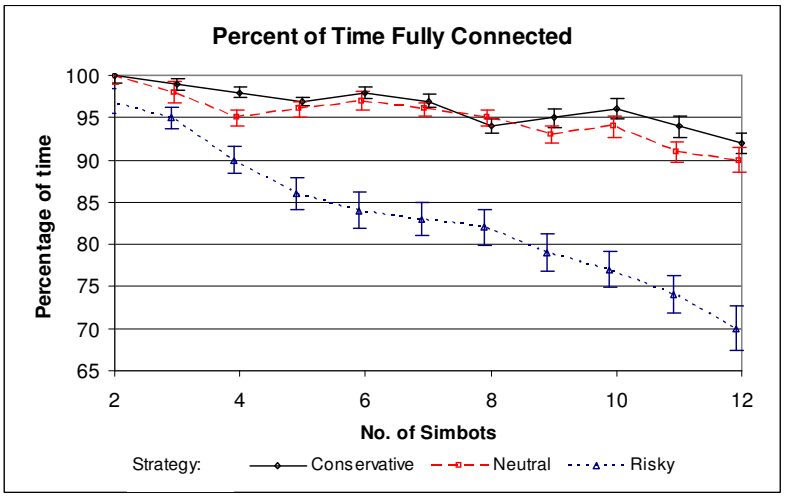


Figure 8.6: Percentage of time fully connected in the medium environment for the *LOS* model using three strategies: Conservative, Neutral and Risky. The vertical lines show the standard deviation. The risky strategy has much lower percentages compared to the other two strategies.

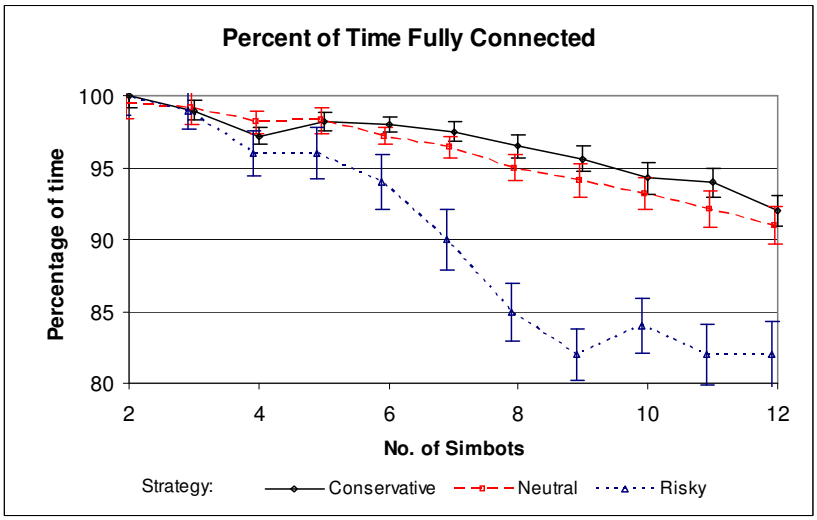


Figure 8.7: Percentage of time fully connected in the medium environment for the *RF* model using three strategies: Conservative, Neutral and Risky. The vertical lines show the standard deviation. The risky strategy has much lower percentages compared to the other two strategies.

Figures 8.8 to 8.9 present the graphs of the exploration time for different team sizes with different local network sizes. The data from the graphs is the average exploration time of the simulation trials for the three proposed strategies. In the small size environment (Figure 8.8 for *LOS* and Figure 8.9 for *RF* communication) for team sizes of more than five simbots and small local networks ($k < 3$) the exploration time is the same if not larger than that of smaller team sizes. In the medium size environment for more than seven simbots and small local networks ($k < 4$) the exploration time is similar if not larger than that of smaller team sizes. From figure 8.10 it is observed that the exploration time for a certain simbot team size n is the same for local network of size $k > 0.8n$. This is due to the fact that during most of the exploration for these sizes of local network the local network is the same as the global network. The simbot network typically forms a tree-like structure with multiple branches.

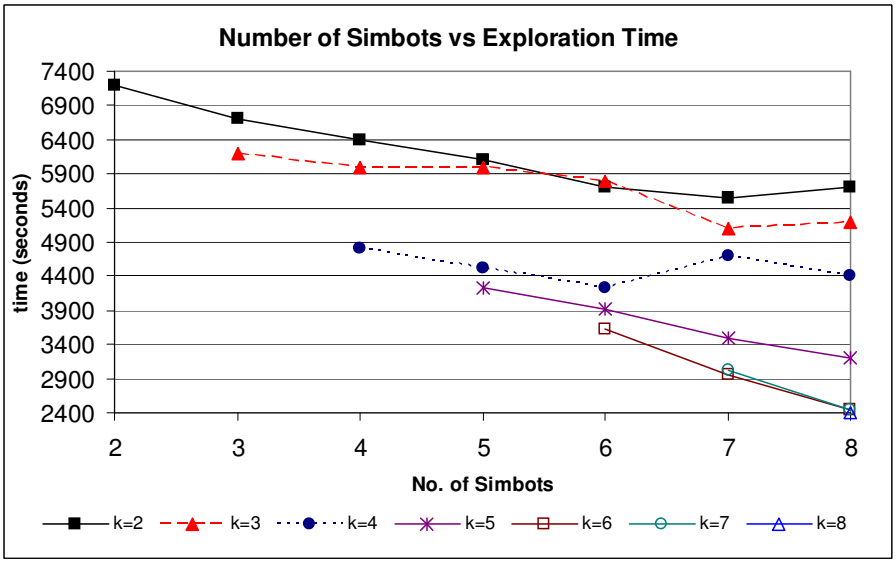


Figure 8.8: Experimental results for the small environment using the *LOS* communication model for local network sizes $k = 2, 3, \ldots, n$, where n is the number of simbots. The exploration time decreases when the size of the local network is increased.

Analysis of the trials revealed that for medium team sizes ($n > 8$) and small local network sizes ($k < 4$) the simbot network tended to have either one or no Explorer simbots for some periods of time. The simbots have a similar behaviour to that of previous architectures (Sweeney, 2002) based in *leader-follower* relationships. A simbot explores the environment while the rest of the simbots maintain the network connected. When there are no Explorer simbots in the simbot network the network tends to contract and eventually becomes static. Eventually a Pusher simbot transitions to

the Explorer role when its expiration time has passed and the exploration continues but time is lost (Section 4.5, page 77). The expiration time is a minimum role time during which the transition to the Explorer role is inhibited for a Pusher simbot.

The following section presents the description of the scenarios under which BERODE was found to achieve suboptimal solutions and proposes solutions to improve the performance in these scenarios.

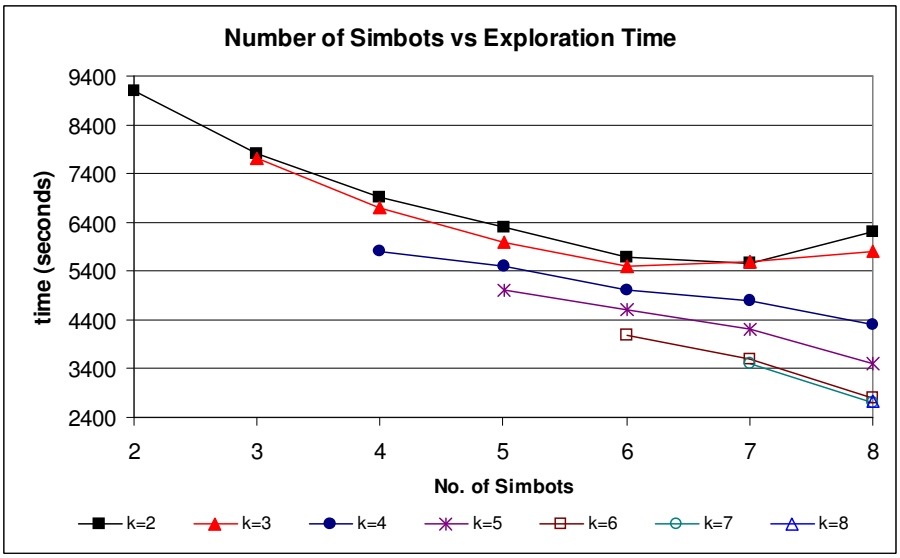


Figure 8.9: Experimental results for the small environment for the *RF* communication model for local network sizes $k = 2,3,\ldots,n$, where n is the number of simbots.

8.7 Enhancements to BERODE

The BERODE architecture was proposed as a decentralized solution for the problem of exploring an environment while keeping connected a communication network. In the initial simulation a trade-off between the exploration time and the size of the local network was found. This simulation revealed that the performance of the architecture in general was as expected: more simbots built the complete map in less time for a certain local network size; however, the performance was poor in some trials for small local network sizes ($k < 4$) and medium team sizes ($n > 7$). The addition of more robots did not provide much if any benefit. The time to build the complete map was just slightly smaller and in some occasions worse. The main causes of the poor performance in these trials were:

1. The lack of an Explorer behavioural role in the simbot network.

2. Inefficient exploration by an Explorer simbot.

3. The MST control network could not be calculated because a simbot was in the Recoverer behavioural role.

The lack of an Explorer behavioural role in the simbot network is caused when the simbots give up the exploration process due to the lack of unexplored areas where the predicted safety level is safe. When the last two Explorer simbots in the network give up the exploration process at almost the same time the simbots transition to the Pusher behavioural role. These two simbots have outdated information about each other based on which they determine that there is an Explorer simbot left in the network. Information is outdated because of the delays in the retransmission process in the *ad hoc* network. This can be prevented using semaphores but this requires either closely synchronized clocks or a special semaphore owner. This is contrary to the decentralized philosophy of our approach and would make the system more fragile.

When there are no Explorers left the network tends to contract and eventually becomes static. The time that the network takes to become static depends on the topology of the network and the environment configuration. Most of the time the simbot network tends to move away from the unexplored areas. Eventually a Pusher simbot transitions to the Explorer role when its expiration time (Section 5.4, page 98) has passed and the exploration continues but time is lost.

Figure 8.11 presents an example of the lack of Explorer simbots in the simbot network. In Figure 8.11(a) simbots $R_{0,E}$ and $R_{2,E}$ move towards the frontiers f_0 and f_2 respectively while $R_{1,M}$ maintains the network. In Figure 8.11(b) simbots $R_{0,P}$ and $R_{2,P}$ transition to the Pusher behavioural role because there are no exploration areas where the communication is predicted to be safe. It is observed that the simbot network tends to shorten. In Figure 8.11 (c) it is observed that the simbots tend to move away from the unexplored areas. This is caused because of the repulsive forces exerted on R_1 by simbots R_0 and R_2.

Solutions such as detecting that the simbot network has become static or detecting that a Pusher simbot is being pushed back by the network are unsuitable. The first one because of the long time that it may take to the network to become static. The second one because a Pusher simbot is almost never pushed back by the network movement. This is illustrated in the example from Figure 8.11(c).

The second cause of poor performance occurs when an Explorer simbot moves towards a *frontier* that is closer to other simbots. Figure 8.12 presents an example of this

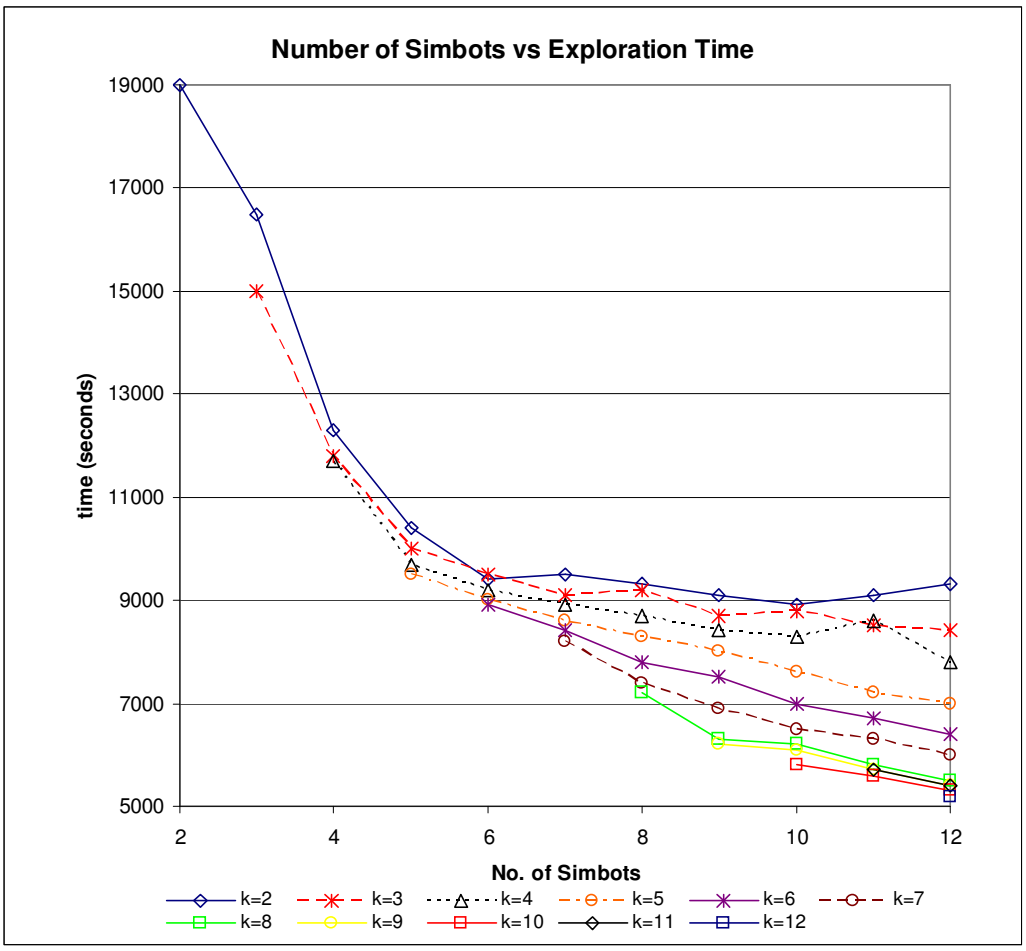


Figure 8.10: Experimental results for the medium environment using the *RF* communication model for local network sizes $k = 2, 3, \ldots, n$, where n is the number of simbots. The exploration time decreases when the size of the local network is increased.

situation where the Explorer simbots $R_{0,E}$ and $R_{2,E}$ move towards the *frontiers* f_0 and f_2 respectively. After some time has passed (Figure 8.12(b)) simbot $R_{2,P}$ transitions to the Pusher behavioural role, $R_{0,E}$ finishes exploring f_0 and selects f_2 as the next *frontier* to explore. It is observed that the closest simbot to the frontier is $R_{2,P}$, which transitioned at a previous instant to the Pusher behavioural role. $R_{2,P}$ is the simbot that could reach the *frontier* more quickly. Figure 8.12(c) presents the common scenario in this type of situation where the simbot network slowly moves in the appropriate direction. Simbot $R_{0,E}$ moves towards the unexplored area while the Pusher simbots induce movement and the simbot network tends to rotate.

In some occasions the advance towards this type of unexplored areas is compromised because the repulsive forces generate local minima for the simbots. Figure 8.12(c') presents this scenario in which the simbot $R_{1,M}$ tends to remain in the up-

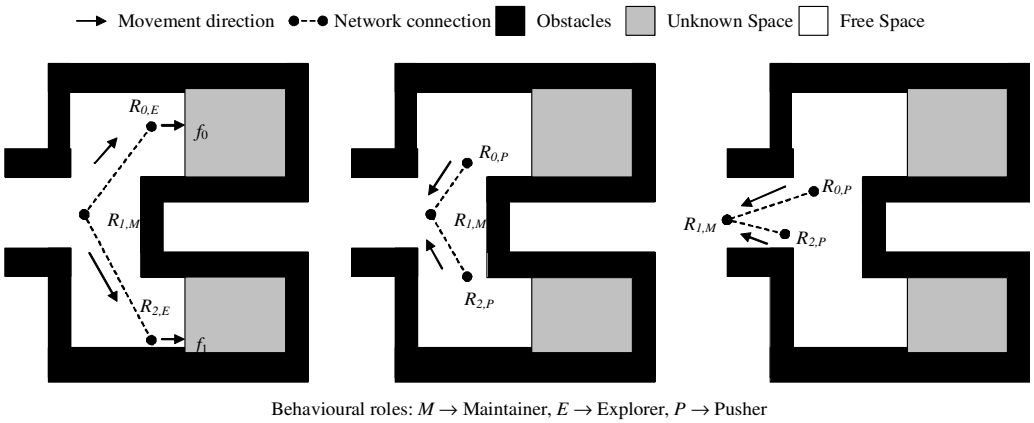


Figure 8.11: An example of the lack of a Explorer in the simbot network. a) Simbots $R_{0,E}$ and $R_{2,E}$ move towards the frontiers f_0 and f_2 respectively while $R_{1,M}$ maintains the network, b) Simbots $R_{0,P}$ and $R_{2,P}$ transition to Pusher because the exploration areas are not communication safe, c) The simbot network tends to shorten and the Pusher simbots move away from the unexplored areas.

per corner of the environment because the repulsive forces generate a local minimum at this place. Eventually the simbot $R_{1,M}$ will move away from the corner and tend to move closer to $R_{0,E}$ but the exploration time increases. In the worst case $R_{0,E}$ transitions to the Pusher behavioural role because the signal quality is worst as the simbot moves away from $R_{0,E}$. The situation could be described as a scenario in which the Explorer simbot tends to move in counterflow with respect to the rest of the network and although eventually the network is pulled by the Explorer simbot the process is either slowed down or abandoned by the Explorer simbot.

The third cause of the poor performance occurs when the simbots give up the recalculation of the MST control network because there is a simbot in the Recoverer behavioural role. Recalculating the network with Recoverer simbots is unsafe because the communication is either marginal or has been lost for the Recoverer simbots.

To address the poor performance we tried mechanisms that detect situations where the exploration is halted (no Explorer left in the network) or inefficient. Exploration is inefficient when there is a more suitable simbot to explore an area than the one that is currently trying to explore such area. These situations were the main source of poor performance and by avoiding them the performance was improved. As previously discussed the use of semaphores to prevent the halting of the exploration requires either

closely synchronized clocks or a special semaphore owner. This is contrary to the decentralized philosophy of our approach and would make the system more fragile. Our strategy to solve this problem is then to detect the deadlocks rather than avoid them.

The mechanisms that we tried addressed the poor performance in a decentralized fashion. These mechanisms were: Role Exchange mechanism, random time MST checking mechanism and tendering mechanism. The role exchange mechanism is a mechanism in which an Explorer simbot exchanges its role with a Pusher role. In BERODE the simbots have knowledge about their local network. They also know the estimated positions of the simbots within this network. An Explorer simbot can then determine if there is a Pusher simbot within its local network that is closer to its currently selected *frontier.* In this case the simbot exchanges its role with this simbot and the exploration process can be speeded up; moreover due to the exchange in roles the problem illustrated in Figure 8.12(c') is avoided because the mentioned counterflow problem is no longer present.

When a simbot transitions from the Explorer role it checks that there is at least one Explorer in the network. Sometimes this information is outdated when the transition occurs due to communication delays. The checking process fails and the simbot network has no simbots in the Explorer role. To address this problem we proposed that the simbots check at random intervals if there are any Explorer simbots left in the network. The random interval is a time that has passed since the last MST control network was calculated. This time is a user defined time plus a random interval. This greatly reduces the situation in which two simbots simultaneously recalculate the MST control network. Provided that this time has passed and there are no Explorer simbots in the local network the simbot starts the recalculation process. The criterion used to transmit the MST control network is different to the one used in Section 5.10.1 (page 132). In Section 5.10.1 the recalculated MST control network is transmitted only when it is different from the previous one. Depending on the existence or absence of Explorer simbots the following actions are taken:

- Explorer simbots in the MST control network: The same MST control network without recalculation is transmitted. This causes the updating of the random interval time for the rest of the simbots. Otherwise their random intervals are not updated and the MST control network is calculated very frequently.

- No Explorer simbots in the MST control network: The simbot recalculates the

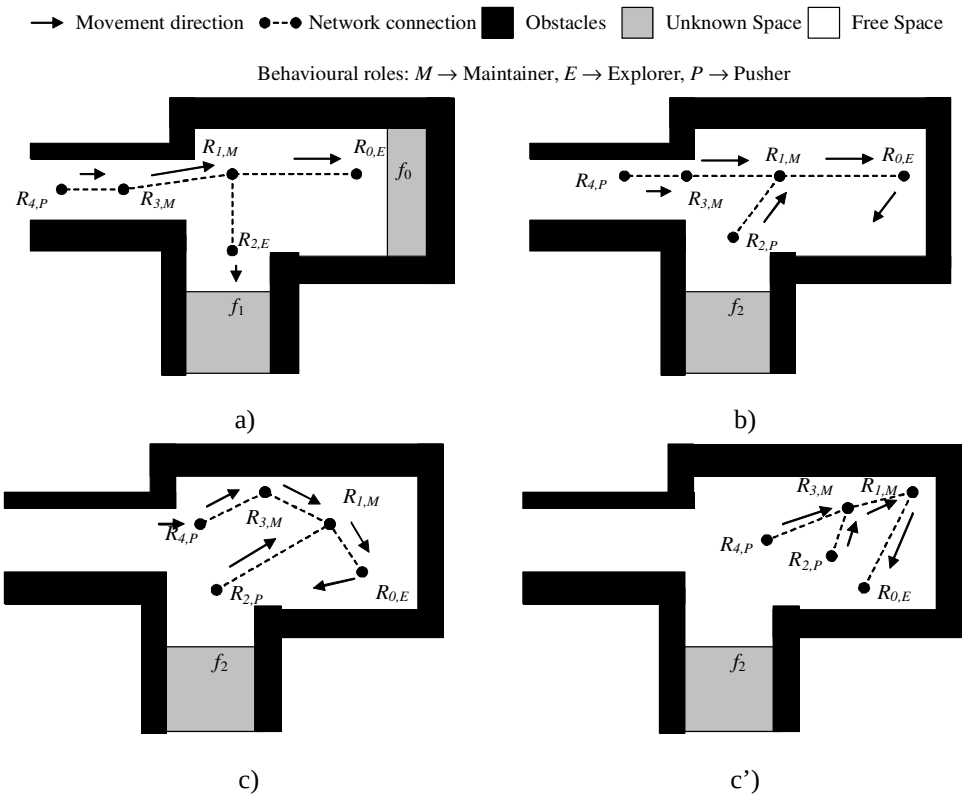


Figure 8.12: An example of inefficient exploration for a common and a worst case scenario. a) Simbots $R_{0,E}$ and $R_{2,E}$ move towards the frontiers f_0 and f_2 respectively. b) Simbot $R_{2,p}$ transitions to the Pusher behavioural role, $R_{0,E}$ finished exploring f_0 and selected f_2 as the next frontier to explore. c) The common case of exploration where the simbot network moves in the appropriate direction. c') The worst case scenario where $R_{1,M}$ tends to remain in the corner.

MST control network and starts a tendering mechanism to determine an Explorer simbot for the updated MST control network. This mechanism is described in the following subsection.

The random time MST checking mechanism addresses the problem of the lack of Explorer simbots in the network and the situation in which the MST control network could not be calculated because there was a simbot in the Recoverer behavioural role. These mechanisms are expected to enhance the performance of BERODE. The version of BERODE that incorporate these mechanisms will be referred as BERODE-2. These additional mechanisms are executed by the simbots' Communication Manager (Section 5.3, page 93).

8.7.1 The Tendering Mechanism

The tendering mechanism is a recovery mechanism implemented to speed up the exploration process once it is discovered that there are no Explorer simbots left in the simbot network. In this mechanism simbots in the Pusher role select a frontier and offer a tender to explore it. The tendering mechanism selects the best tender. The simbot with the best tender transitions to the Explorer role. Although it is possible to select more than one simbot as Explorer the multiple selection frequently leads to recurrent conflicts. Figure 8.13 presents an example of a typical scenario for the tendering mechanism. Multiple selections of Explorer simbots in this scenario is inadequate. The Explorer simbots try to explore areas that are difficult to explore simultaneously. The simbot network is stretched in multiple directions. In the worst case scenario (Figure 8.13(b)) all the Explorer simbots give up the exploration and the tendering mechanism is called again. The selection of only one Explorer simbot avoids this recurrent situation. Eventually the Pusher simbots that are not selected as the Explorer simbots transition to the Explorer role and collaborate in the exploration task.

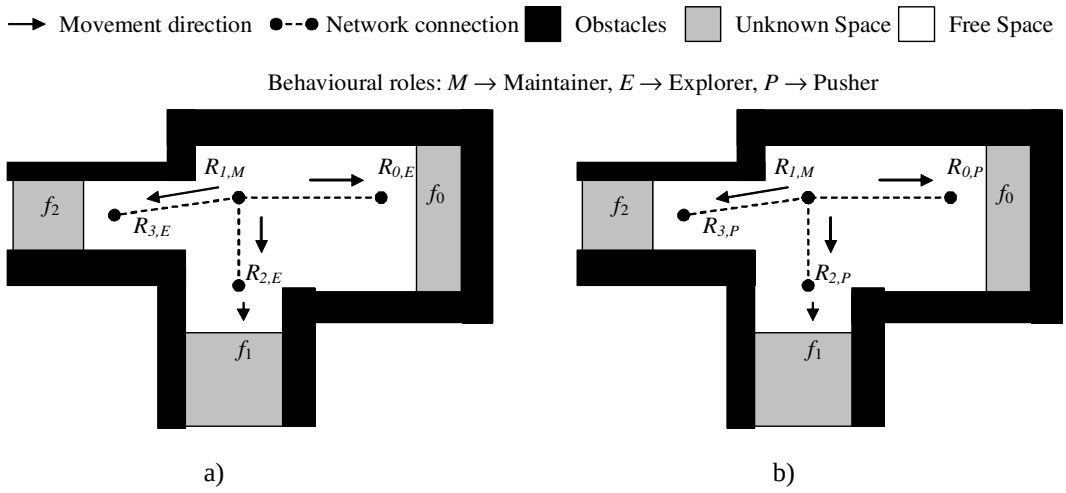


Figure 8.13: A difficult scenario for the tendering scenario. A) Simbots $R_{0,E}$, $R_{2,E}$ and $R_{3,E}$ try to move towards frontiers f_0, f_1 and f_2 respectively. b) Simbot $R_{2,E}$ fails to explore the environment and transition to the Pusher behavioural role.

The tendering mechanism has two stages: selling and tendering. In the selling stage the simbot that discovered that there were no Explorers left in the simbot network recalculates the MST control network and transmits a message. This simbot is referred to as the buyer. The message includes the recalculated MST control network and a

tendering query. Once the simbots have received this message they remain static until the tendering process finishes for communication safety. All the simbots with one control connection in the recalculated MST control network transition to the Explorer behavioural role (Section 5.4, page 98). In the tendering stage the Explorer simbots transmit their selected frontier.These frontiers are the tenders from each simbot. The buyer simbot waits until it has received the tenders from all the simbots in the Explorer behavioural role or a certain time has passed (to avoid the lock up of the process). After this the buyer simbot selects the best tender and declares this simbot as the winner. The best tender is the *frontier* with the largest utility in the smallest hierarchy level (Section 5.6.2.1, page 114). The winner simbot remains as an Explorer simbot while the rest of the simbots that submitted a tender transition to the Pusher behavioural role.

8.7.2 Message Types for BERODE-2

In addition to the messages proposed in Section 5.10.1 (page 132) BERODE-2 implements two additional messages. These messages are presented in Table 7.3.

The role exchange message is used by the Explorer simbots to exchange their roles with a Pusher simbot that is closer to the unexplored area than the Explorer simbot. Two acknowledged message types are used. The first one by the Explorer simbot to exchange the roles and the second one used by the Pusher simbot to confirm the exchange of roles.

The tendering mechanism message is used by the simbot that carries out the tendering process. A request message is used by this simbot to query the simbots about their tenders. The simbots send their tenders to this simbot. The second message (acknowledge message) is sent once the Explorer simbot has been determined.

Content	M_T	**Message Type**	**Message Purpose**	**Frequency of Transmission**	**Description**
Tendering Mechanism	*N*	Request Acknowledged	Control	Event Based	Recovering mechanism to speed up the exploration process
Role Exchange	*K*	2 Acknowledged	Control	Event Based	Exchange of behavioural roles between pairs of simbots

Table 8.3: Additional types of messages used in the BERODE-2 architecture.

8.8 Comparing BERODE-2 with BERODE

BERODE-2 incurs larger computational and communication costs over the initial approach. It is important to compare BERODE-2 against the initial version to assess the enhancements. The comparison is based on the map exploration time. The architectures are compared using the Neutral strategy from Section 8.7. The architectures were compared for the LOS and RF models. The settings for these simulations are the same as in the previous simulations. The three heuristics are used to calculate the MST control network. Each combination of team size and local network was run for 10 trials. It is expected that the additional mechanisms implemented for BERODE-2 will allow the simbots to explore the environment faster. A reasonable increase in the communication cost is expected because the additional mechanisms are not required frequently. The rest of the communication between the simbots is frequently transmitted therefore the additional communication cost is expected to be low.

Figure 8.14 shows the results for the *LOS* communication model in the medium environment. The graph shows the results for local network sizes of $k = \{2,4,6\}$. It is observed that BERODE-2 outperforms BERODE for each team size for the same local network size. Moreover for team sizes larger than seven simbots BERODE-2 with a local network k=2 has smaller exploration times than BERODE with a local network k=4. This is also observed when the local network k=4 for BERODE-2 and BERODE has a local network k=6.

Figure 8.15 shows the results for the *RF* communication model in the medium size environment. As in the case of *LOS* the BERODE-2 outperforms BERODE for each team size for the same local network size for the *RF* model. It is also observed that for team sizes ($n > 7$) BERODE-2 with a smaller local network has the same if not smaller exploration times than BERODE.

By comparing Figure 8.14 and Figure 8.15 it is observed that the simbots that use the *RF* model have smaller exploration times than the simbots using the *LOS* model. This is to be expected because the *LOS* communications restricts the exploration more. The percentage of time that the network is fully connected is smaller (by 5%-10% on average) for the *LOS* model than for the *RF* model. The simbots spend more time fixing and maintaining the network in the *LOS* model. Similar results were found for the small environment for the *LOS* and *RF* models. The communication range for the *RF* model is 5m in free space. Although in the simulated *LOS* model the range is theoretically infinite in practice only in a few places the *LOS* signals have large ranges

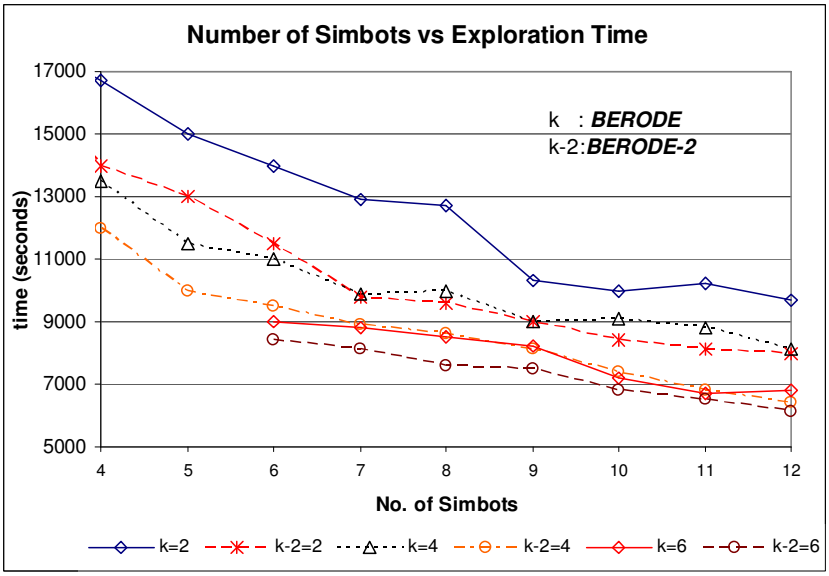


Figure 8.14: Comparison between BERODE and BERODE-2 using the *LOS* communication model in the medium environment for local network sizes $k = 2, 4, 6$. BERODE-2 explores the environment faster than BERODE. The trends for BERODE-2 are shown as dotted trends.

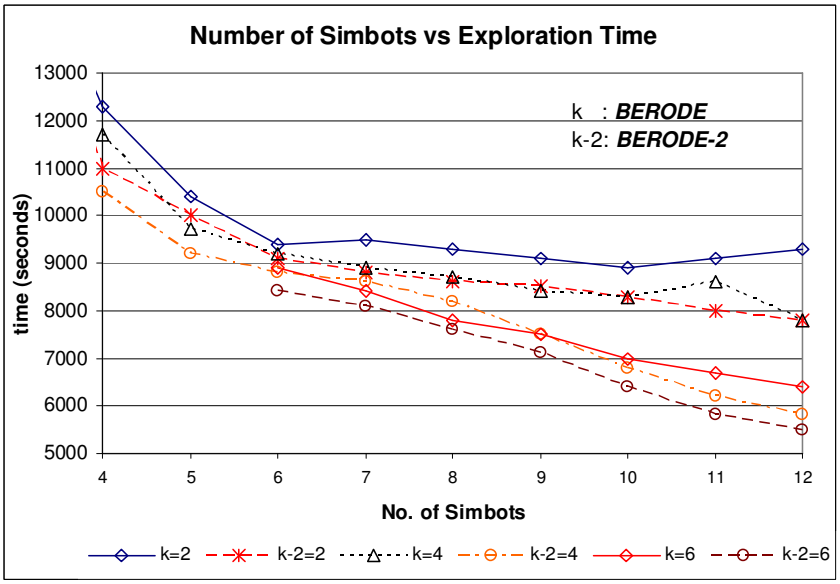


Figure 8.15: Comparison between BERODE and BERODE-2 using the *RF* communication model in the medium environment for local network sizes $k = 2, 4, 6$. BERODE-2 explores the environment faster than BERODE. The trends for BERODE-2 are shown as dotted trends.

(above 5m) for the proposed simulated environments. This is the case in typical indoor environments. In outdoor environments *LOS* may have a better performance because these environments are less cluttered.

Figure 8.16 presents the percentage of additional required bandwidth for the *RF* model. This percentage is the additional bandwidth used by the simbots using BERODE-2 compared to BERODE. The local network sizes used in the comparison are $k = \{2,4,6\}$. The percentage is based on the bandwidth required by the simbots (in bytes/sec.). This calculation is based on the transmitted messages using the proposed application level protocol. The details of the calculation are described in Appendix C.

From Figure 8.16 it is observed that the additional bandwidth is larger when the local network is smaller for most team sizes. For a local network of size *k*=2 as the number of simbots increases the additional bandwidth also increases. For a local network of size *k*=4 the additional bandwidth required remains around 5.5% for all the simbot team sizes. For a local network of size *k*=6 the additional bandwidth tends to slowly decrease as the number of simbots increases.

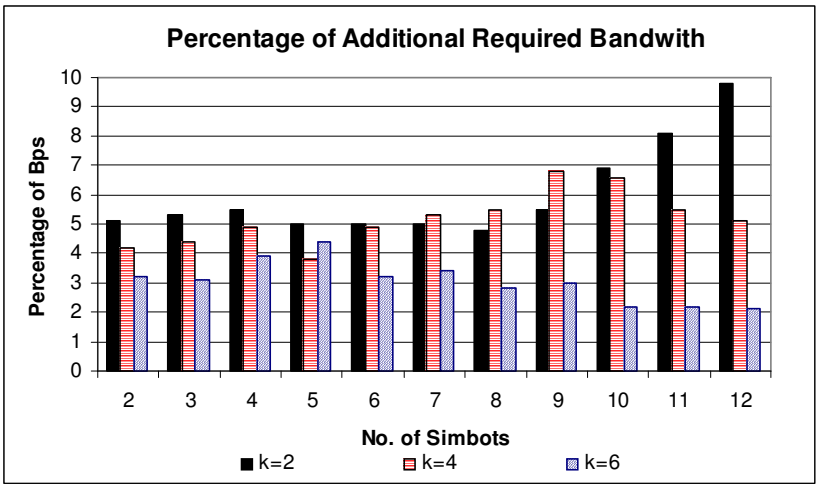


Figure 8.16: Percentage of additional required bandwidth (Bps) by BERODE-2 compared to BERODE using the *RF* model in the medium environment for local network sizes $k = 2,4,6$.

It was found that the simbots with the smallest local network size (*k*=2) more frequently execute the mechanism to detect if there are Explorer simbots left in the network. In the absence of Explorer simbots the tendering mechanism is started. The checking and tendering mechanisms are expensive in terms of communication costs. All the simbots in the network are queried and information is retrieved acknowledging the reception of information. This suggests that small local network sizes ($k < 4$) may not be suitable for BERODE and BERODE-2. This issue is addressed in the following

chapter. For the larger network sizes (k=4, 6) typically the local network contains at least one Explorer simbot. The checking mechanism is then executed less frequently in these larger networks.

Similar trends to those shown in Figure 8.16 were obtained for the *LOS* model. For the *LOS* model the additional required bandwidth was smaller ≅10% compared to that of the *RF* model. This is attributed to the fact that in the *LOS* model a simbot has in average fewer simbots within communication range.

It is concluded that BERODE-2 has a better performance than BERODE. The exploration times are smaller for BERODE-2 while the additional required bandwidth is small. Moreover for medium sizes of the local network the additional bandwidth is the same if not decreased as the size of the local network is increased. The following chapter presents simulations that analyze the trade-off between the size of the local network and the required bandwidth.

8.9 Conclusions

This chapter presented the implementation of the BERODE (Behavioural Role Decentralized) Architecture for the *LOS* and *RF* communication models. An initial simulation to compare performance of BERODE for different strategies was conducted. The strategies reflect a desire to maintain a certain signal quality for the simbots. Three strategies were proposed: Conservative, Neutral and Risky. Conservative strategies try to maintain higher levels of signal quality than Risky strategies. Risky strategies complete the exploration in less time because the simbots are less constrained to maintain a compact network. The neutral strategy is the best because it combines the benefits of the two other strategies. The neutral strategy requires slightly more time to explore the environment than the Risky strategy but the simbot network remains connected most of the time, as occurs with the Conservative strategy.

In this simulation for all the three strategies it is concluded that:

- The percentage of time that the network remains fully connected decreases as the number of simbots is increased. This is to be expected because larger simbot networks have to maintain more communication constraints to remain fully connected.

- *RF* networks are easier to maintain fully connected than *LOS* networks. We expect this because of the on/off nature of *LOS* communication. In *RF* com-

munication the *RSSL* (Received Signal Strength Level) makes the maintenance task easier because it provides a good measure of the expected behaviour of the signals.

- There is a trade-off between the size of the local network and the exploration time. For a certain number of simbots as the local network size is increased the simbots require less time to build the map. In this chapter the ratio between the local and global updates was a fixed value of 4:1. The following chapter presents simulations that analyze the effect of different ratios and update frequencies for the local and global updates.

- The time to explore the environment decreases as the number of simbots increases. This trend ceases to hold when the number of simbots is larger than a certain size (eight for the medium size environment) for small local network sizes ($k < 4$). This poor performance was caused by inefficient exploration and the lack of simbots in the Explorer role. Inefficient exploration occurs when there is a simbot closer to an unexplored area than the simbot that is currently trying to explore the area. The lack of simbots in the Explorer role was caused by the communication delays which caused simbots to rely on outdated information. The lack of Explorer simbots increases the time that the simbots require to build the map.

An Enhanced version of BERODE to address the poor performance was proposed (BERODE-2). BERODE-2 adds three mechanisms to the original version: role exchange, random time MST checking and tendering. The role exchange mechanism is used by the Explorer simbots. An Explorer simbot exchanges roles with a Pusher simbot when the Pusher simbot is closer to its current exploration area. The random time MST checking mechanism verifies that there is at least one Explorer simbot in the network. If there are no Explorer simbots in the network the tendering mechanism is triggered. The tendering mechanism designates an Explorer simbot. The designated simbot is the simbot with the best value *frontier*.

We presented a simulation that compared the two versions of BERODE. The conclusions from this simulation were:

- *LOS* technologies with unlimited range restrict the exploration task more than short range *RF* technologies (5m range). This was expected because in typical

indoor environments there are only a few places where the *LOS* signals can cover large areas.

- BERODE-2 has a better performance than BERODE. The simbots require less time to build the maps (13% less in the simulations). The additional bandwidth required by BERODE-2 is small, 10% in the worst case.

- The additional bandwidth required in BERODE-2 tends to increase as the number of simbots increases for a local network size k=2. The additional bandwidth required in BERODE-2 tends to decrease as the number of simbots increases for a local network size $k \geq 4$. This suggests that small local network sizes ($k < 4$) may not be suitable for BERODE and BERODE-2.

BERODE-2 implements mechanisms that improved the performance of the BERODE architectures. BERODE-2 is scalable because the additional bandwidth required compared to that of the original BERODE approach is at most the same (with a maximum of 9% in the simulations). BERODE-2 will be used in the following chapters.

The following chapter presents simulations that identify trade-offs between the local network sizes and simbot team sizes when parameters such as the frequency of the beacon signal are varied. Chapter 10 presents an empirical analysis of the consistency in the maps built by the simbots. Chapter 11 presents the conclusions of this thesis.

Chapter 9

Scalability, Robustness and Performance of BERODE-2

9.1 Introduction

This chapter presents simulations to assess the performance of the BERODE-2 architecture using *LOS* (Line of Sight) and *RF* (Radio Frequency) communication models. To improve the realism of our simulations relevant measured aspects of the robot sensors and communication devices have been included in the simulated models. This is explained in detail in Chapter 7. Throughout this chapter we will refer to our simulated robots as simbots.

BERODE-2 implements mechanisms that improve the original BERODE architecture proposed in Chapter 4. These mechanisms addressed the problems of inefficient exploration and temporarily inactive exploration observed in BERODE. The first couple of simulations assess the robustness of BERODE-2 for different parameterizations. Afterwards simulations to analyze the scalability of the approach with respect to large numbers of simbots are presented. The scalability is assessed in terms of the communication costs and the time that the simbots require to build the complete map. In BERODE-2 each robot builds its own feature map of the environment. The robots' maps incorporate observations from all the robots. The exploration process stops once any one robot in the network considers that its map is complete[1]. A final simulation to compare BERODE-2 with other approaches is presented at the end of this chapter.

[1] A map is considered to be complete once it is projected into a probabilistic grid map and the size of the portions of the environments for which there is no evidence is below a used defined threshold (Section 8.6, page 239).

The simbots update their feature map using an Extended Kalman Filter (*EKF*) (Smith et al., 1988). The *EKF* produces an estimate (along with its uncertainty) of the position of the features and the robot. The positions of the simbot and the features are referred to the datum (origin) of a global Cartesian system, which is the initial position of the robot with the smallest ID[2] number.

BERODE-2 is based on behavioural roles. These roles reactively adapt to the dynamic conditions of the communication network formed by the simbots as they explore an environment. The communication network is maintained as a fully connected network by creating and updating an MST (Minimum Spanning Tree) control network. The MST control network contains only the minimum links required to have a fully connected network. The MST control network is calculated at the start of the exploration based on the communication network topology and an abstraction heuristic. The simbots retain knowledge of their local network. The local network for a simbot is the network that contains all the simbots within a *k-hop* distance. The MST control network is modified either partially (local network) or completely by the simbots during the exploration process to improve the communication quality. Section 9.2 compares the proposed four heuristics for network modification: connectivity, position, pos. predictive and QS predictive.

The roles generate reactive plans that maintain the simbot's direct connections in the MST control network. The reactive plans try to improve the signal quality for the communication constraints of the simbot. The communication constraints are the connections that a simbot has to keep within communication range. The reactive plans use a predictive model that predicts the signal quality for the communication constraints of a simbot for positions near to the simbot. In BERODE-2 simbots send beacon signals periodically containing information about their estimated position and its uncertainty. The predictions are based on these positions. The predictive model can be either reactive or plan focused. In the reactive model the positions tested are closer to the simbot position than in the predictive model; thus shorter plans are generated more frequently. Section 9.3 presents a set of simulations to analyze the performance of BERODE-2 for different degrees of reactivity. Several beacon signal frequencies are used. The performance is measured in terms of the movement that the simbots have to make to keep the network connected. The performance is considered better when the simbots have to move less because this is more energy efficient. The simulations show that as the frequency of the beacon signal is decreased the degree of reactivity has to be decreased

[2] The robots have an ID number that allows them to identify each other in the network.

in order to get the best performance. This was expected because reactive actions are taken once new beacon signals are received; therefore the robots have to rely on longer term plans because the beacon signals are less frequent.

One of the main goals in BERODE-2 is to be scalable with respect to the communication cost and the time required to explore an environment. Previous approaches have failed to achieve both criteria simultaneously. Centralized approaches are not scalable because a central agent coordinates and generates plans for all the simbots. This agent has to gather and distribute information frequently. As more simbots are added to the network the required communication bandwidth increases exponentially (Scott and Yasinsac, 2004). Centralized approaches are not scalable with respect to the communication cost. Decentralized approaches do not scale well with respect to the time required to explore the environment because of the lack of coordination. Simbots frequently explore areas that have been already explored by other simbots. Having more than a few simbots often does not decrease the time required to explore an environment.

BERODE-2 implements a hierarchical approach to get a good trade-off between co-ordination and communication costs. The hierarchical approach has two levels: local and global. At the local level information is distributed within the local network. At the global level information is distributed to all the simbots in the network. Information is shared frequently at the local level, while at the global level information is shared less frequently. Most of the communication is retransmitted within a small neighbourhood avoiding scalability problems with respect to the number of simbots. Section 9.4 presents simulations to analyze the performance of BERODE-2 with respect to the size of the local network. The performance is analyzed in terms of the exploration time and the communication cost. The exploration time is the time that the robots require to build the complete map. The simulations show that there is a trade-off between the efficiency of the exploration and the knowledge obtained from the local network. Access to more knowledge from the local network decreases the exploration time.

Although having simbots with an effectively unlimited communication range may seem in principle attractive, it is undesirable for the purposes of scalability. As more simbots are added to the simbot network the required communication bandwidth increases. Moreover, there is more interference between the communication signals. Section 9.5 presents simulations to analyze the effect of the communication range on the performance of BERODE-2. The simulations show that there is a minimum value

for the communication range above which the time that the simbots require to build a map is the same.

In BERODE-2 the simbots recalculate the MST control network either partially or globally. It is argued that the recalculation of the network aids the exploration process because the signal quality of the MST connections is improved. Section 9.6 presents simulations to determine the effectiveness of adaptability in the MST control network. In the simulations BERODE-2 is compared with several fixed networks. In the simulations BERODE-2 builds complete maps in less time than the fixed networks. Moreover because of its adaptability BERODE-2 maintains the network fully connected for more time than the fixed networks. Section 9.7 presents the summary of the conclusions from the simulations.

9.2 Comparison of Heuristics for the MST Control Network

This section presents simulations to compare the performance in BERODE-2 of using the different heuristics proposed in Section 4.4.1 (page 70) to calculate the MST control network. The MST control network is calculated at the start of the exploration based on the communication network topology and a heuristic. The MST control network is modified either partially or completely by the simbots during the exploration process to improve the communication quality. The four proposed heuristics are: connectivity, position, pos. predictive and QS predictive.

The connectivity heuristic tries to divide equitably the communication constraints imposed to the simbots. The Position heuristic imposes communication constraints between simbots that have close positions. The pos. predictive heuristic imposes communication constraints between simbots that have small values of discomfort distance. The discomfort distance is the difference between a desired signal quality and the current signal quality for the communication constraint. These values are calculated based on the Cartesian positions of the simbots. The heuristic estimates the required movement to have a zero value for the discomfort distance. The QS predictive heuristic uses the same principle as the pos. predictive heuristic. The difference is that the values of the discomfort distances are calculated using the signal qualities obtained from the *RSSL* (Received Signal Strength Level) instead of the Cartesian positions of the simbots. The *RSSL* is an available value in *RF* technologies. *LOS* technologies do not

provide this value. The QS predictive heuristic is therefore only applicable for *RF* technologies.

We propose two metrics to compare the performance of the heuristics: map exploration time and the path speed per simbot. The exploration time is the time required by the simbots to build the complete map. As previously explained the exploration stops once any one robot considers the map to be complete. The path speed (m/min.) per simbot is the distance that the simbots travelled during the exploration divided by the exploration time and the number of simbots. The path speed is useful to estimate the amount of movement required to maintain the network fully connected. In the simulations when the simbots move they move with a fixed speed. Simbots that have as their main task the maintenance of the network remain static for periods of time. These simbots remain static because they are in the best position to keep their communication constraints. We expect that the four heuristics will have similar exploration times. It is expected that the path speed for the predictive heuristics will be smaller than for the other heuristics because the movements of other simbots are taken in account therefore the simbots have to adjust their positions keep the network fully connected. The predictive heuristics should tend to minimize the amount of movement necessary to maintain the simbot network.

The settings for the simulations are the same as those of the simulations in Section 8.6 (page 239). The heuristics were tested in the small and medium environment from Figure 7.1 (page 164) using the *RF* and *LOS* models. The size of the team of simbots was $n = 2,...8$ and $n = 4,..,15$ in the small and medium environments respectively. The local network sizes were $k = 1,...,n$. Each heuristic was run for 10 trials for each combination of simbot and local network size.

Figure 9.1 and Figure 9.2 show the comparison of heuristics with respect to the exploration time (min.) for the *RF* and *LOS* communication models respectively. For clarity, these figures illustrate the results obtained for a small local network size (k=3) and a large local network size (k=6). From these figures it is observed that:

- Based on the exploration time the heuristics have the following order in terms of decreasing performance: QS predictive, pos. predictive, connectivity and position.

- The trend for the QS predictive and pos. predictive heuristics tends to be linear with respect to the number of simbots. The connectivity heuristic has a logarith-

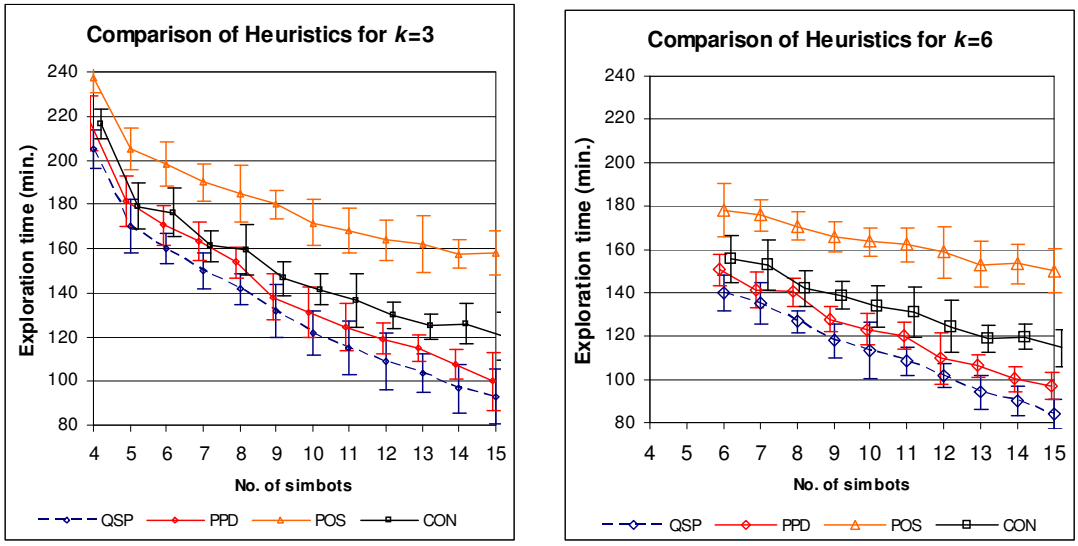


Figure 9.1: Comparison of the exploration time the four heuristics: QS Predictive (QSP), Pos. Predictive (PPD), Position (POS), and Connectivity (CON) for the *RF* model for local networks sizes k=3,6 in the medium environment. The vertical lines show the standard deviation.

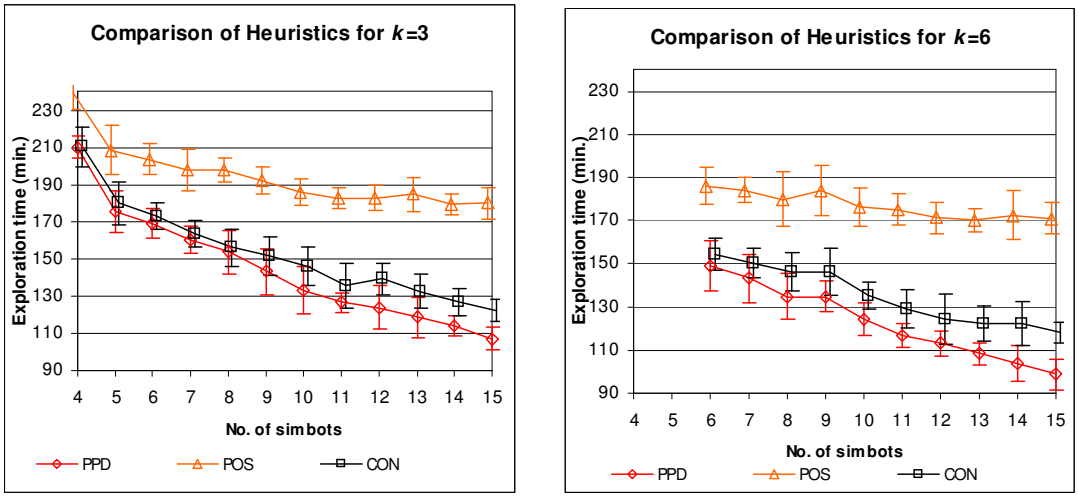


Figure 9.2: Comparison of the exploration time for the three heuristics: Pos. Predictive (PPD), Position (POS), and Connectivity (CON) for the *LOS* model for local networks sizes k=3,6 in the medium environment. The vertical lines show the standard deviation.

mic trend for the *LOS* model. The position heuristic has a logarithmic trend for both communication models.

- In all the heuristics there is a trade-off between the exploration time and the size of the local network. The trade-off is approximately proportional for local network sizes $k \leq n/2$. When $k > n/2$ the exploration time decreases less. For these local network sizes the local network for the simbots is the same as the complete MST control network most of the time. The MST control network can be visualized as a tree structure where most of the time the depth is smaller than $n/2$.

- For the position heuristic for team sizes larger than a certain number of simbots the exploration time is similar. The use of additional simbots does not decrease the exploration time much. For instance for the position heuristic with a local network of size k=6 in the *LOS* model for team sizes bigger than 10 simbots the exploration time is very similar ($\pm$2min.). The same characteristic is observed for the connectivity heuristic when the *RF* model is used.

The previous observations are general observations with respect to the local network size. Similar findings were found in the small environment.

Figure 9.3 and Figure 9.4 show the comparison of heuristics with respect to the path speed (m/min.) for the *RF* and *LOS* communication models respectively. From these figures it is observed that:

- Based on the path speed the heuristics have the following order in terms of decreasing performance: QS predictive, pos. predictive, connectivity and position.

- In the QS predictive and pos. predictive heuristics as more simbots are added to team the path speed decreases. For the connectivity heuristic a similar trend is observed when the number of simbots is below a certain size. For instance in the *RF* model for simbot networks smaller than ten simbots the path speed decreases in an approximately linear fashion.

- In the position heuristic for the *RF* model there is a logarithmic decrease. Moreover for the small local network (k=3) the path speed tends to increase as the size of the team increases. In the LOS model this increment was found for all the local network sizes.

- In the QS predictive heuristic the path speed is nearly invariant with respect to the local network size.

Based on the two metrics the following conclusions are drawn:

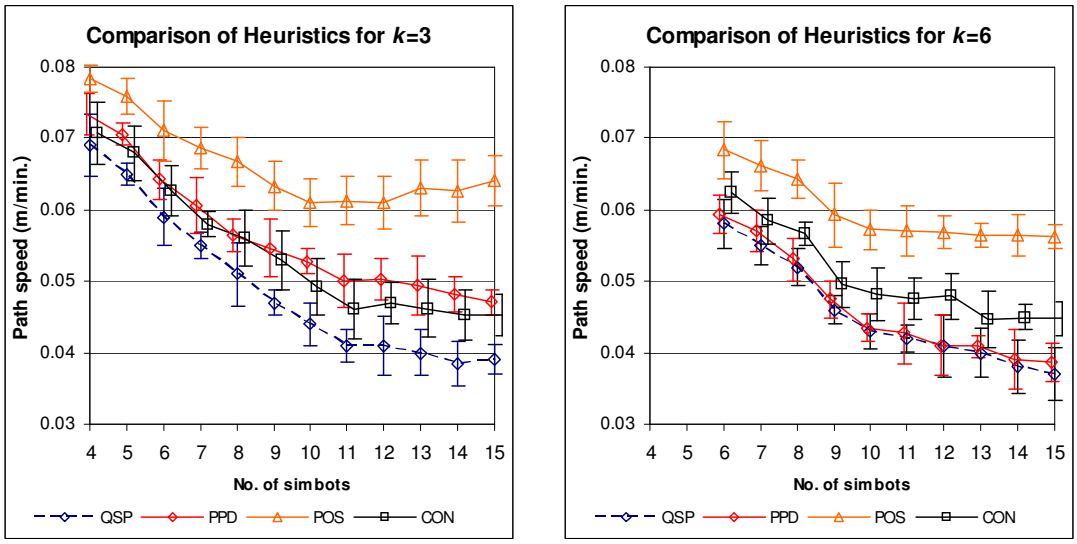


Figure 9.3: Comparison of the path speed for the four heuristics: QS Predictive (QSP), Pos. Predictive (PPD), Position (POS), and Connectivity (CON) for the *RF* model for local networks sizes k=3,6 in the medium environment. The vertical lines show the standard deviation.

- The QS predictive heuristic is the best heuristic among the proposed heuristics because it has smaller exploration times and the simbots have to move less than with the rest of the heuristics. The movement required by the simbots in this heuristic is invariant with respect to the local network size. This is a desirable aspect in the implementation with real robots when power consumption is a constraint in the system because the robots spend less energy. Movement is the main source of power consumption for a mobile robot. However more simulations are required to validate this finding when the simbots communicate less frequently within the local and global levels. These simulations are presented in Section 9.4.

- The pos. predictive heuristic is the second best heuristic and best for the LOS model. The heuristic is suitable for scalability purposes because it has a decreasing trend in the path speed and the exploration time as the size of the team

increases. This heuristic has two disadvantages with respect to the QS predictive heuristic: it has a slightly worse performance with respect to the exploration time (4.6% on average) and the increment in the path speed with respect to the size of the local network.

- The connectivity heuristic is suitable only for small team sizes. For larger team sizes increasing team size does not decrease much the exploration time and the path speed. Moreover, in the *LOS* model the path speed is the same if not worse.

- The position heuristic is not useful for scalability purposes. The exploration time has a logarithmic trend and the path speed tends to increase as the size of the simbot team increases.

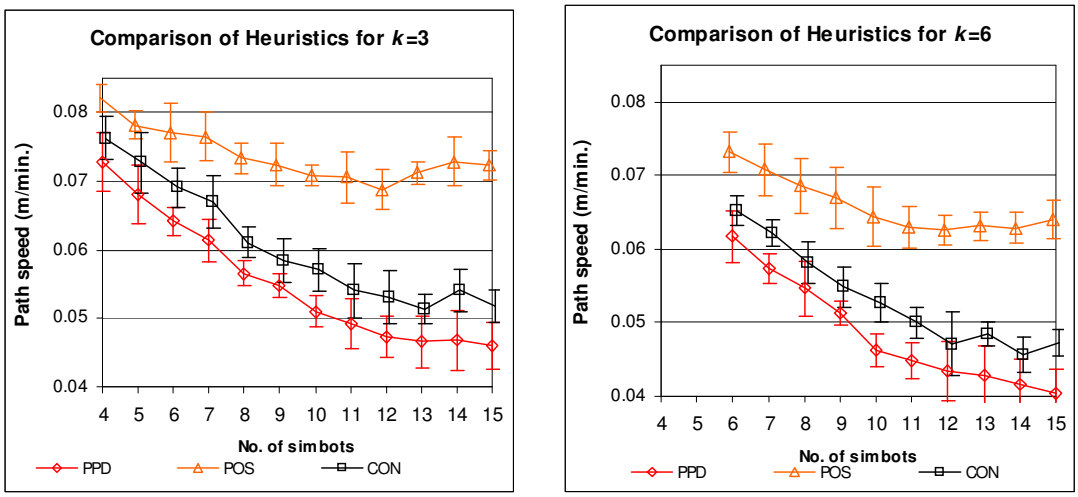


Figure 9.4: Comparison of the path speed for the three heuristics: Pos. Predictive (PPD), Position (POS), and Connectivity (CON) for the *LOS* model for local networks sizes k=3,6 in the medium environment. The vertical lines show the standard deviation.

It is concluded that as expected the predictive heuristics have a better performance than the other heuristics because the movements of other simbots are taken in account therefore the simbots have to adjust their positions less to keep the network fully connected. The predictive heuristics minimize the amount of movement necessary to maintain the simbot network.

It is also concluded that regardless of the heuristic used there is a trade-off between the exploration time and the size of the local network. The trade-off is approximately proportional when the ratio between the sizes of the local network and the simbot

network is less than one half. For larger ratios there is only a small decrease in the exploration time. This is expected because the simbots tend to create MST control networks where most of the simbots are within an (n/2) *hop* connectivity distance; therefore for large ratios the local network is frequently the same as the MST control network.

9.3 The Predictive Model: Long Term Plans vs. Reactive Plans

A predictive model is used by the simbots in the Planning Module (Section 5.6, page 102). The model predicts the quality of the communication signal for nearby positions to the simbot position. The model estimates the attenuation due to the obstacles in the direct path of the signal. In the predictive model for *LOS* technologies any obstacle in the direct path blocks completely the signal. In *RF* technologies the model estimates the loss of power (in dB.) based on recent signals received by the simbots.

The Planning Module is used by the simbots to determine the position where the signal quality is the best for all its communication constraints. This module has three stages: projection, sampling and planning. At the projection stage a subset of features from the feature map is projected into a local grid map. At the sampling stage the current simbot position and several positions inside the local grid map are tested to determine the position with the best OSQ (overall signal quality). At the planning stage a plan to move to the position with the best OSQ is generated. The predictive model can be either reactive or plan focused. In the reactive model the positions tested are closer than in the predictive model. Thus, shorter plans are generated more frequently.

The tested positions are randomly sampled. Sampling a position is an expensive operation in terms of computation. A ray tracing algorithm is used to predict the signal quality for each communication constraint.

In BERODE-2 simbots send beacon signals containing information about their estimated position. The predictions are based on these positions. The beacon signal is sent frequently. The period of time between beacon signals is referred as update time.

It is important to analyse the performance of BERODE-2 when the period of the update time is increased in order to determine if BERODE-2 is capable of keeping the full connectivity of the network despite this increase. This is a desirable characteristic because of its benefits in terms of communication (e.g. reducing the bandwidth

required and power consumption). In this section we test several update times.

The simulations presented in this section compare three levels of reactivity: $S1$, $S2$ and $S3$, where $S3$ is the most reactive and $S1$ is the least reactive. The reactiveness of the predictive model depends on the closeness of the tested positions. A scale factor η determines the closeness of the sampled positions with respect to the size of the projected local grid map. Approaches that are more reactive generate plans more frequently because they move towards closer positions. If the same number of samples is used for the different levels of reactivity the computational cost is bigger for the more reactive approaches because they generate plans more frequently. To have a fair assessment of the level of reactiveness, the computational cost of the different levels of reactivity must be the same. The parameters for the different levels of reactivity are calculated assuming that the best test position is in a corner of the projected grid map. In preliminary simulations the CPU time was measured for the three reactive levels. The CPU time was similar (within 3.5%) for the three reactive levels, therefore we argue that this assumption is a useful basis for fairly assessing the reactive levels.

The more reactive approaches use fewer samples because they sample the positions more frequently than less reactive approaches. Preliminary simulations revealed that 200 samples and a scale factor η=1 obtained $97.5 \pm 0.2\%$ of the time the best OSQ possible for the projected grid map. Based on these values the scale factor and the number of samples for the reactive levels are shown in Table 9.1.

Reactiveness level	**Scale factor**	**Number of Samples**
$S1$	1.0	200
$S2$	0.75	110
$S3$	0.50	70

Table 9.1: Parameters for the Planning Module for the three reactive levels: $S1$, $S2$ and $S3$.

As explained previously in Section 5.6.1 (page 105) is important to assess the level of reactivity for different update times to determine the suitability of the approach for technologies with low bandwidths or when power consumption is a constraint in the system. The reactive levels are compared for the update times T = 1,5,10 sec. These times were selected based on the fact that the process of sensing an area takes 1.8 sec.

We propose two metrics to measure performance: The percent of time fully connected and the path speed per simbot. The percent of time fully connected is the percentage of the time that the simbot network remains fully connected. The path speed per simbot (m/min.) is the travelled distance by the simbots during the exploration

divided by the exploration time and the number of simbots. The exploration time was not used as a metric because the main interest of this section is to assess the effect of the reactivity on the connectivity of the network.

The settings for the simulations are the same as those of Section 8.6 (page 239). The reactive levels were tested in the medium environment from Figure 7.1 (page 164) using the *RF* and *LOS* models. The size of the team of simbots was $n = 4,..,15$ in the medium environment with local network sizes $k = 1,...,n/2$ for each team size. The previous experiment revealed that for $k > n/2$ the local network tends to be the same as the global network most of the time. The performance for these local network sizes tends to be the same. Each heuristic was run for 10 trials for each combination of simbot and local network size. It was expected that the reactivity level would be inversely related to the update time. Approaches that are more reactive should perform better when information is updated more frequently.

Figure 9.5 presents the comparison of the percentage of time fully connected for updates times T=1, 5, 10 sec. using two local network sizes k=3, 6. The graph shows the results in the medium size environment for the *RF* communication model. From these figures it is observed that:

- For all the reactive levels the percentage of time fully connected decreases when the frequency of the update time decreases. This is expected because the simbots move for longer periods of time between the beacon signal updates.

- For all the update times as the number of simbot increases the percentage of time fully connected decreases until a certain minimum value is reached. The minimum value tends to be smaller for small local network sizes (k=3). This seems to be due to the less knowledge obtained from a small local network. Further simulations in Section 9.4 analyze this in detail.

- The percentages of time fully connected are smaller for the local network size k=3 compared to k=6.

- For the update time T=1 sec. the more reactive the approach the larger the percentage of time fully connected. This is observed for all the simbot network and local network sizes.

Similar results were obtained for the *LOS* model. In the *LOS* model the percentages of time fully connected are smaller with respect to those in *RF* model. This is expected because of the increased difficulty in maintaining line of sight communication.

Figure 9.6 presents the graphs of the path speed for the update times T=1, 5, 10 sec. From these graphs it is observed that:

- The path speed decreases as the number of simbots increases for all the reactivity levels. This was expected because the QS predictive heuristic was used to calculate the MST control network. When more simbots are added the simbots have to travel less to build the complete map. The previous section discussed this finding in detail.

- The increment in the path speed as the update time increases is considerably larger for small local networks compared to the increment in larger local networks. As observed in Figure 9.7, for a local network of size k=6 the surface is much flatter than for the local network of size k=3.

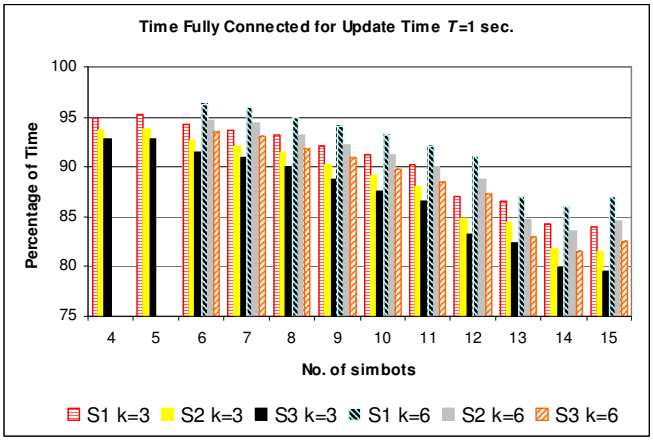
- For an update time T=1 sec. the path speed is smaller as the module is more reactive. Analysis of the recorded information of the trials revealed that the less reactive level generated longer term plans than the more reactive levels. These longer plans were dropped when the conditions of the network change (i.e. lost connection). The simbot had to move more to recover from unpredicted situations.

- For an update time T=5 sec the path speed is smaller for the intermediate level of reactiveness. The difference between the path speeds of the three levels of reactiveness is very small.

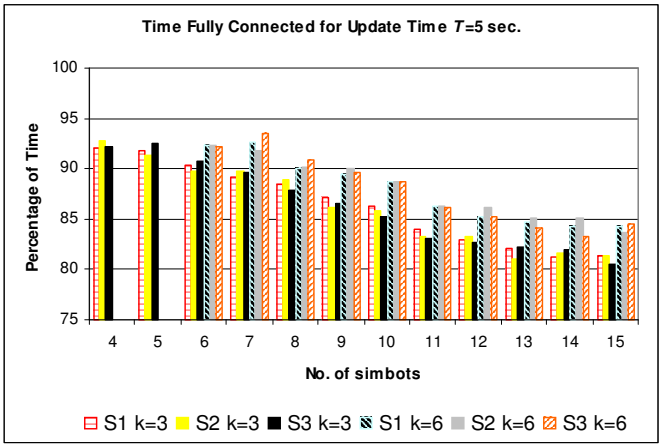
- For an update time T=10 sec. the path speed is smaller as the module is less reactive. In the more reactive approaches the simbot generates many short terms plans based on the same information about the positions of the simbots. The simbot tends to move towards the best possible OSQ in steps. Analyses of the recorded information revealed that the incremental paths frequently include small detours. The less reactive approaches generated shorter paths because the same plan remained for larger periods of time.

Similar findings were observed for the *LOS* model. In the case of the *LOS* model the increment in the path speeds are larger. This confirms our findings from Section 9.2.

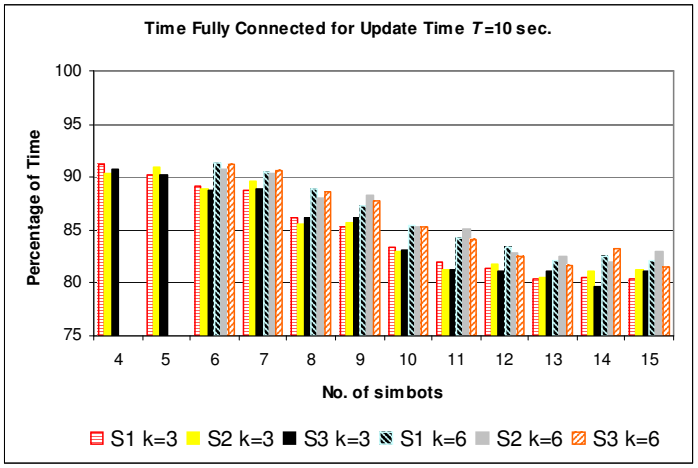
Based on the observations from the metrics it is concluded that:



a) $T = 1$ sec.

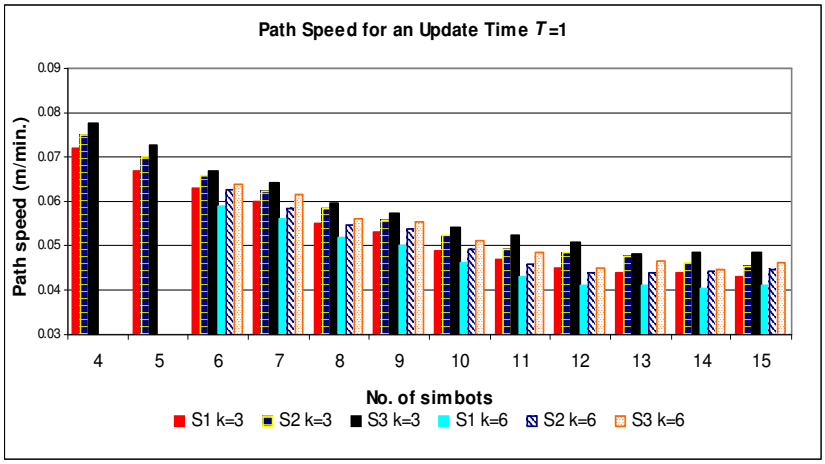


b) $T = 5$ sec.

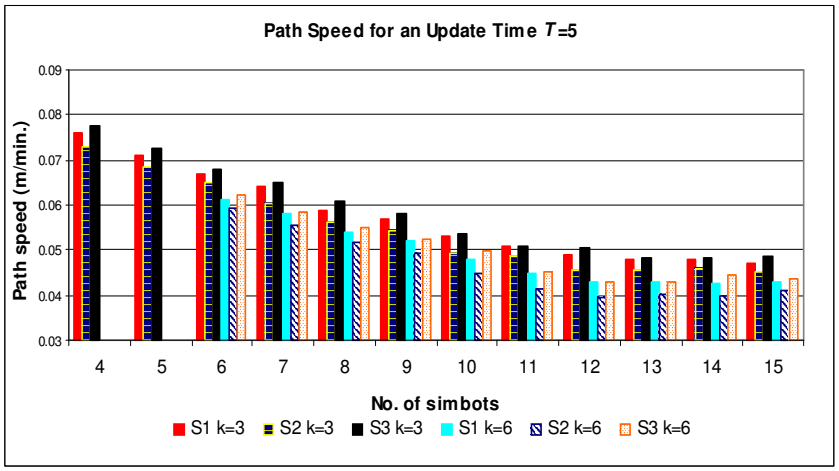


c) $T = 10$ sec.

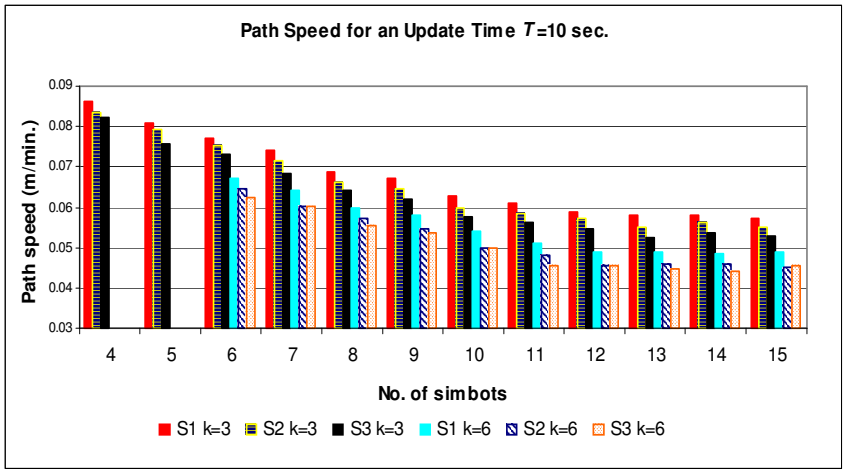
Figure 9.5: Comparison of the time fully connected for an update times T=1,5,10 sec. Three levels of reactivity ($S1$, $S2$, $S3$) for two local network sizes k=3,6 are compared using the *RF* communication model. The simulations use the medium environment.



a) $T = 1$ sec.



b) $T = 5$ sec.



c) $T = 10$ sec.

Figure 9.6: Comparison of the path speed for update times T=1,5,10 sec. Three levels of reactivity ($S1$, $S2$, $S3$) for two local network sizes k=3,6 are compared using the *RF* communication model. The simulations use the medium environment.

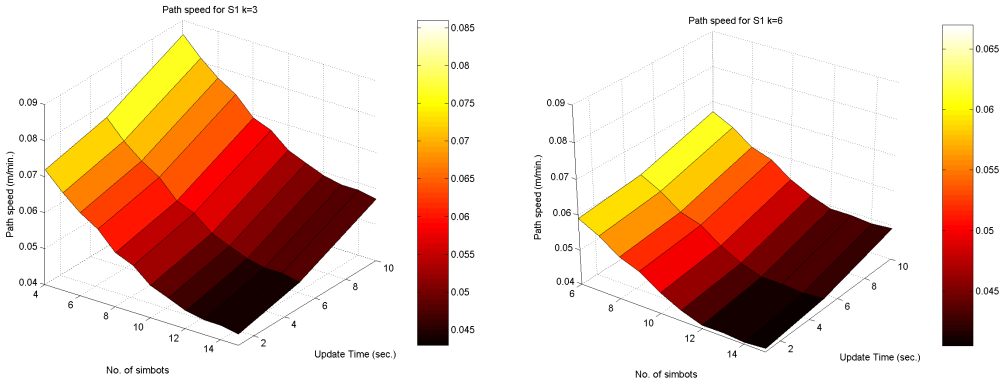


Figure 9.7: Path speed for the reactive level $S1$ for the local network sizes k=3, 6 using the *RF* communication model. The robots have to correct less their position (smaller path speed) when the size of the local network is larger.

- Regardless of the reactivity of the approach when the update time is decreased the percentage of time that the network is fully connected decreases.

- There is a certain minimum value for the percentage of time that the network remains fully connected. This value depends mainly on the size of the local network and to a less extent on the reactiveness of the module. The validation of the existence of a minimum value for more infrequent update times is analyzed in the following section.

- The degree of reactiveness required to achieve the best performance in terms of the path speed is related to update time. As expected reactive approaches are suitable for small update times whereas long term plans perform are suitable for large update times.

The existence of a minimum percentage of time fully connected (80% in the worst case) for the simbot network is very important for scalability purposes. The minimum percentage is slightly lower for local networks of small size compared to larger local networks. It seems reasonable to suppose that this is due to fact that less knowledge can be obtained from a smaller local network. In fact, as shown in the following section, when the simbot has access to less information or information is received less frequently, the minimum percentage decreases.

9.4 The Effect of the Size of the Local Network

One of the main goals in BERODE-2 is scalability. BERODE-2 implements a hierarchical approach to get a good trade-off between coordination and communication costs. The hierarchical approach has two levels: local and global. At the local level information is distributed within the local network. The local network for a simbot is the network that contains all the simbots within a *k-hop* distance. At the start of the exploration the MST control network is built. The simbots retain knowledge of their local network. At the global level information is distributed to all the simbots in the network. Information is shared frequently at the local level, while at the global level information is shared less frequently. Most of the communication is retransmitted within a small neighbourhood thus avoiding scalability problems as the number of simbots in the architecture increases.

In previous simulations a trade-off between the local network size and the exploration time was identified. It was observed that for a certain size of the simbot network as the size of the local network is increased the exploration time decreases. This was expected because the simbots have access to more information from the simbot network. This section presents simulations to analyze this trade-off in more detail.

We expect to identify the range of local network sizes with the best trading between the exploration time and the communication cost. It is expected that the best trading range will be found to depend on the sizes of the local network and the simbot network.

Two metrics are proposed to analyze the trade-off: The percentage of decrease for the exploration time and the percentage of additional required bandwidth. The decrease percentage is the decrease in the exploration time for a local network of size k with respect to a local network of size k-1 expressed as a percentage. The percentage of additional required bandwidth is the increment in the communication bandwidth for a local network of size k with respect to the smallest size of local network (k=1) expressed as a percentage. This percentage is based on the bandwidth required by the simbots and is calculated (in Bps) by the supervisor controller in the simulator. This calculation is based on the transmitted messages using the proposed application level protocol. The details of the calculation are described in Appendix C.

The settings for the simulations are the same to those of Section 8.6 (page 239) using the medium and large environment from Figure 7.1 (page 164) for the *RF* and *LOS* models. The size of the team of simbots was $n = 4,\ldots,16$ and $n = 8,\ldots,20$ in the medium and large size environments respectively. We executed 10 trials for each

combination of simbot and local network size.

Figure 9.8 shows the percentage of decrease for the exploration time in the large environment using the *RF* model for simbot networks of size n= 4, 6, ..., 20. In the figure the x-axis is the ratio between the local network and the simbot network sizes and will be referred as network ratio (n_R). The trends have a similar pattern and can be divided in three sections with respect to the network ratio. In the first section n_{R1}=[0,R_1] the percentage of decrease is similar ($\pm$0.5% std.). In the second section n_{R2}=[R_1, R_2] there is a linear decrease as the network ratio increases. In the third section n_{R3}=[R_2, 1] the percentage is 0.1 ± 0.8%. The thresholds R_1 and R_2 depend on the size of the simbot network. Figure 9.9 presents the percentage of decrease in a 3D perspective for clarity to compare the results between the *RF* and *LOS* models.

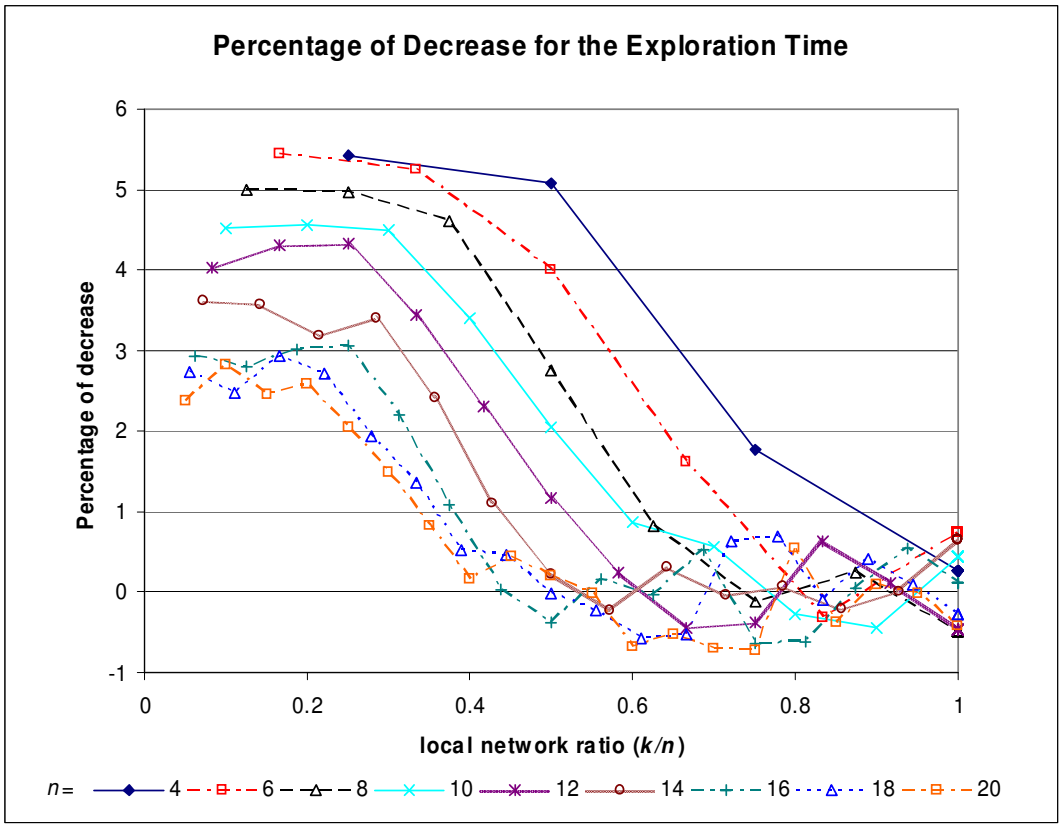


Figure 9.8: Percentage of decrease for the exploration time in the large size environment using the *RF* model for simbot networks of size n=4, 6, ,..,20. The x axis shows the local network in relation to the size of the team, the y axis shows the percentage of decrease.

Further analysis on the data from the *RF* and *LOS* simulations revealed the threshold $R_1 = n^{-0.56}$, which can be roughly approximated $R_1 \cong n^{-0.5}$. The threshold was validated using the data from previous simulations for the small and medium environ-

ments.

The slopes for the second section for the *RF* model are approximately parallel. The slopes are more pronounced for larger sizes of simbot networks for the *LOS* model (Figure 9.9).

It is concluded that:

- The percentage of decrease in time is a fixed percentage for network ratios $n_R \leq n^{-0.5}$. The fixed percentage decreases proportionally as the simbot network size is increased. The fixed percentage is larger for small simbot networks. Based on the results from previous simulations we argue that the larger decrease is caused by the proportionally larger amount of information obtained for small local networks compared to that of large local networks when the size of the local network is increased one level.

- The fixed percentage has a minimum value (around 2.5%) for simbot networks larger than a certain size ($n >$16 in the large environment).

- The benefit of having local networks with $n_R \geq n^{-0.5}$ decreases in a linear fashion. Not much further improvement with respect to the exploration time can be achieved therefore the implementation of local networks with these sizes is not recommendable. Moreover as the local network is increased the communication cost increases.

- As expected the decrease in percentage has an average zero value for network ratios n_{R3}=[R_2, 1]. The analysis revealed that for these network ratios the local network is the same as global network most of the time. There is no difference with respect to the knowledge obtained from the simbot network for these network ratios. The MST control network can be visualized as a tree structure where most of the time the depth is smaller than $n/2$.

- The faster decrease in the *LOS* model for large simbot networks with network ratios n_{R2}=[R_1, R_2] seems to be due to the increased difficulty of maintaining the network fully connected in this model with respect to the *RF* model.

Before we move into the bandwidth metric analysis it is important to mention that the bandwidth involved in the broadcasting of information in mobile *ad hoc* networks increases exponentially with the number of nodes when simple broadcasting[3] is used

[3] In simple broadcasting a node retransmits all the messages that it receives.

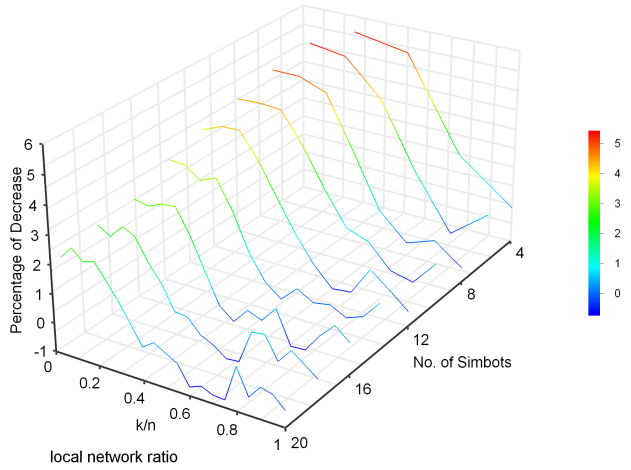
(Scott and Yasinsac, 2004). To reduce this complexity we have used the MST control network to distribute the information efficiently between the simbots. The details of the implementation are shown in Section 5.10 (page 130).

Figure 9.10 presents the percentage of additional bandwidth required in the large environment using the *RF* and *LOS* models. It is observed that the trends have a similar shape and can be divided in three sections with respect to the network ratio. In the first section and third sections the increment is linear. It is observed that for all the trends the increase is linear for network ratios below 2/5 (first section). The linear increase from the third section is observed for network ratios above 3/5. As previously explained for these ratios the local network tends to be the same as the global network. Therefore the increase is linear rather than polynomial as in the second section.

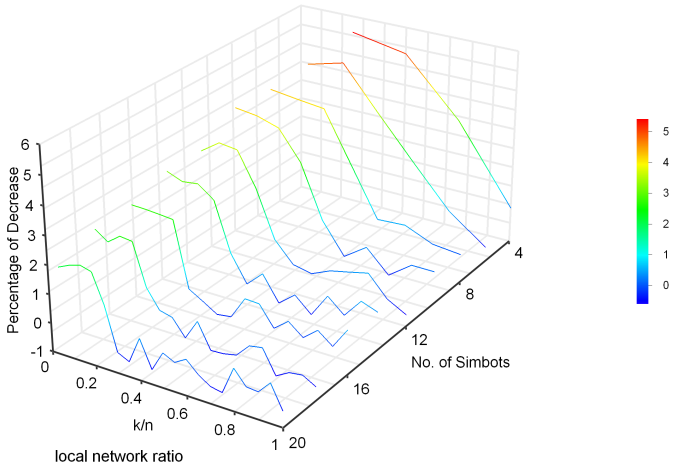
The polynomial increase in the second section is observable for simbot networks with size $n \geq 14$. Polynomial regression analysis on the data from the second section for network sizes $n \geq 14$ revealed that the polynomial order was at least $O(n^2)$. For the available data the maximum value of the polynomial order was $O(n^{4.7})$ for a network size n=20 using the *RF* model. The polynomial order grows with the size of the simbot network. Analysis of the recorded trials revealed that given a certain network ratio in large networks there are more simbots within communication range than in small networks. In large simbot networks a simbot receives more local level messages from simbots that are not part of its local network. These messages only generate traffic because the simbots do not use this information. The polynomial order is larger in the *RF* model (Figure 9.10(a)) compared to the *LOS* model (Figure 9.10(b)). This is expected because the connectivity in the *LOS* model is smaller.

In previous simulations the features extracted from the environment were exchanged between the simbots at the local and global levels every 5 and 20 sensing steps[4] respectively. The ratio between the local and global updates will be referred as update ratio (R_{update}). It is important to validate the observed pattern for the additional required bandwidth when different update ratios are used. For this reason we conducted additional simulations where the global features were exchanged every 10 and 40 sensing steps (update ratios R_{update}=2,8). Figure 9.11 presents the additional required bandwidth for update ratios R_{update}=2,4. It is observed that the percentages are smaller for an update ratio R_{update}=2 compared to R_{update}=4. This is expected because as the update ratio is decreased the information is exchanged only at the

[4] A sensing step is the time that the simbot requires to sense an area using the *low cost* platform. This process takes 1.8 seconds for the proposed *low cost* platform (Section 7.7, page 207).



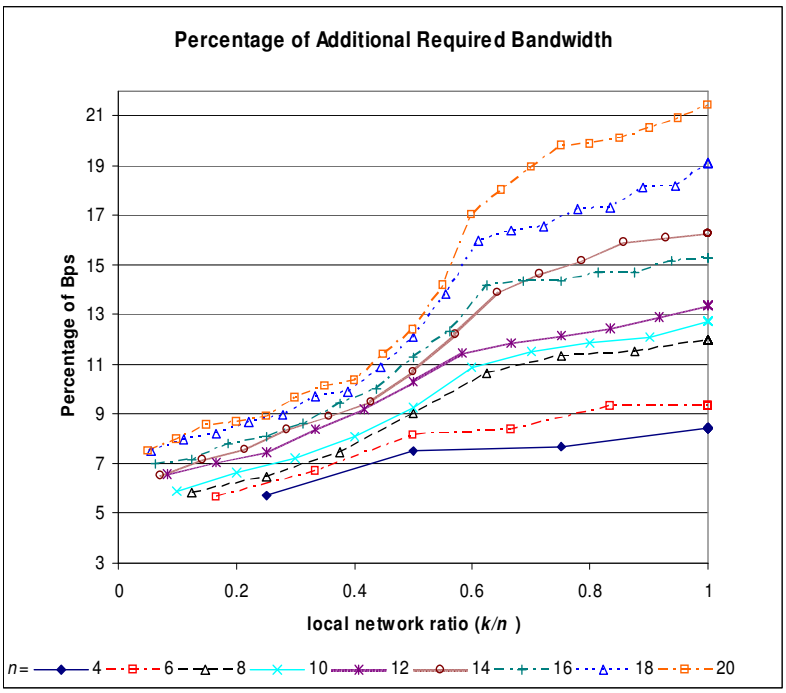
a) Percentage of decrease in the exploration time using the *RF* model



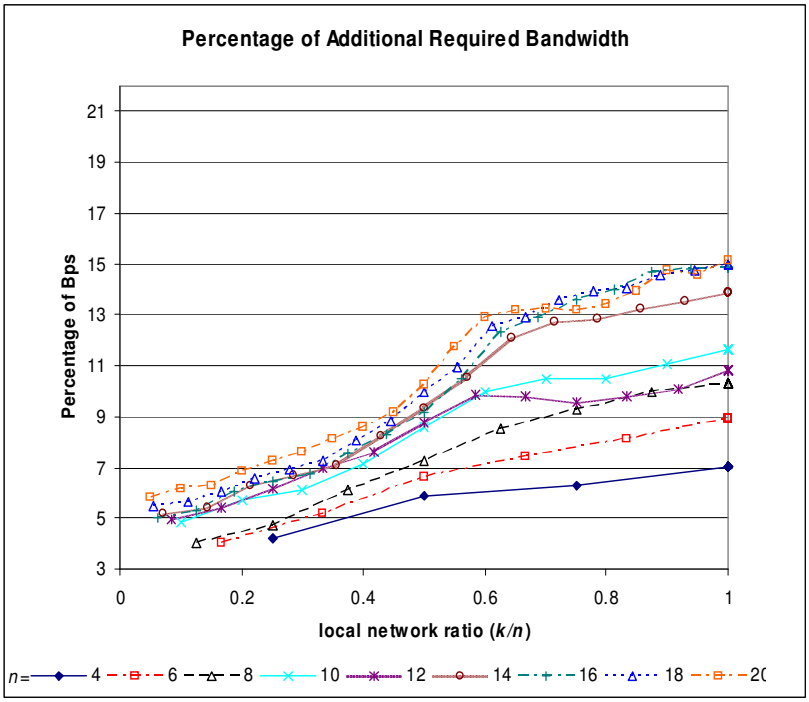
a) Percentage of decrease in the exploration time using the *LOS* model

Figure 9.9: Percentage of decrease for the exploration time in the large environment using the (a) *RF* and the (b) *LOS* model for simbot networks of size $n = 2, 4, ..20$. The local network ratio is the ratio between the sizes of the local network and the simbot network.

global level. Modifying the update ratio does not affect the percent of decrease in the exploration time; however the exploration time tends to increase when the update ratio is increased. Although the increase in exploration time is small (0.7% on average between the values of update ratio R_{update}=4, 8) it was observed that the simbots have a tendency to build maps that contain repeated features. This is caused by the failure in the matching of the external features with the current feature map of the simbots. The computational cost of updating the map is increased because of the repeated features. This issue is addressed in Chapter 10.

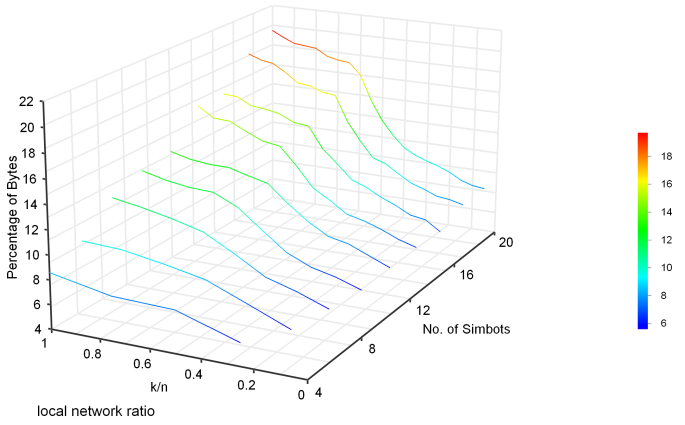


a) *RF* model

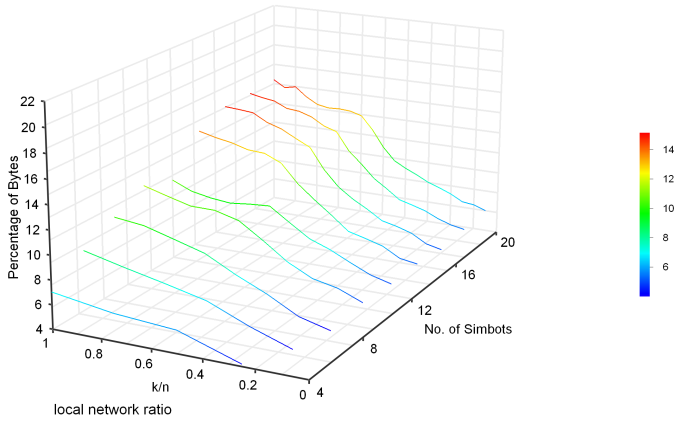


b) *LOS* model

Figure 9.10: Percentage of additional bandwidth required in the large environment using (a) the *RF* model and (b) the *LOS* model for simbot networks of size $n = 4, 6, \ldots, 20$. The x axis shows the local network in relation to the size of the team, the y axis shows the percentage of additional bandwidth required.



a) $R_{update} = 4$



b) $R_{update} = 2$

Figure 9.11: Percentage of additional bandwidth required in the large environment for an update ratio of (a) $R_{update} = 4$ and (b) $R_{update} = 2$ using the *RF* model for simbot networks of size $n = 2, 4, ..20$.

From Figures 9.10 and 9.11 it is observed that as the number of simbots used in the simulations increases the additional bandwidth required is small (with a maximum of 22.3% for 20 simbots using the *RF* model). From a practical point of view given the current trends of wireless technology the additional bandwidth does not represent a problem for the number of simbots used in our simulations. For larger number of simbots the bandwidth might become a problem because the increase is above the linear order with respect to the number of simbots. To verify how fast the bandwidth becomes a problem we conducted additional simulations using simbot team sizes of n=25, 30, 40 in the large environment. From Figure 9.12 it is observed that for flat communication ($n_R = 1$) when the number of simbots increases beyond 30 the increase in bandwidth is large (e.g. 65% for an increase from 30 to 40 simbots). It is clear

that a flat communication approach is not scalable because there is an accelerating increase in the required communication bandwidth as the number of robot increases. This increase will soon surpass the capacity of available communication technologies as the numbers of robots increase. For large teams of simbots $n \geq 40$ the hierarchic approach has practical benefits as it will scale when small network ratios are used $n_R \leq 0.4$. For instance for 40 simbots with a network ratio $n_R = 1$ there is 130% increase in the additional bandwidth, while for a network ratio $n_R = 0.4$ the increase is only 15.3%.

One of the goals of BERODE is to be suitable for large groups of *low cost* small robots. For this type of robots it is very important to manage efficiently the consumption of power everywhere possible, including transmitter power. In the hardware implementation of BERODE the hierarchic communication approach can extend the autonomy of the robots which is always an important consideration for mobile robots. The use of the hierarchic approach not only will allow power consumption saving in the hardware implementation of BERODE but more importantly it will allow the use of BERODE for larger numbers of robots than flat communication ($n_R = 1$) with a given communication bandwidth.

In this section we have analysed the effect of the size of the local network. Based on the two metrics proposed it can be seen that there is a trading between the exploration time and the size of the local network. When the size of the local network is increased the exploration time decreases. We identified the range of local network sizes with the best trading. This range is a function of the network ratio and is n_R=[0, min($n^{-0.5}$, 2/5)]. The threshold ratio for the exploration time is $n^{-0.5}$ and the threshold for the additional bandwidth required is 2/5.

For local networks with ratios n_R =[min($n^{-0.5}$, 2/5),1] the trading between the exploration time and the size of the local network worsens as the size of the local network increases. In this range the percentage of decrease in the exploration time decreases in a linear fashion as the network ratio increases. The percentage of decrease stabilizes at zero. In the range n_R=[min($n^{-0.5}$, 2/5), 3n/5] the additional bandwidth increases in a polynomial fashion with an order of at least O(n^2). The order of the polynomial increases as the size of the network increases. In the range n_R=[3/5,1] the additional bandwidth increases in a linear fashion. In this range the local network is the same as the global network most of the time. Despite the worse trading for these network ratios ($n_R \geq$[min($n^{-0.5}$, 2/5),1]) the space is still explored faster with the largest local network sizes.

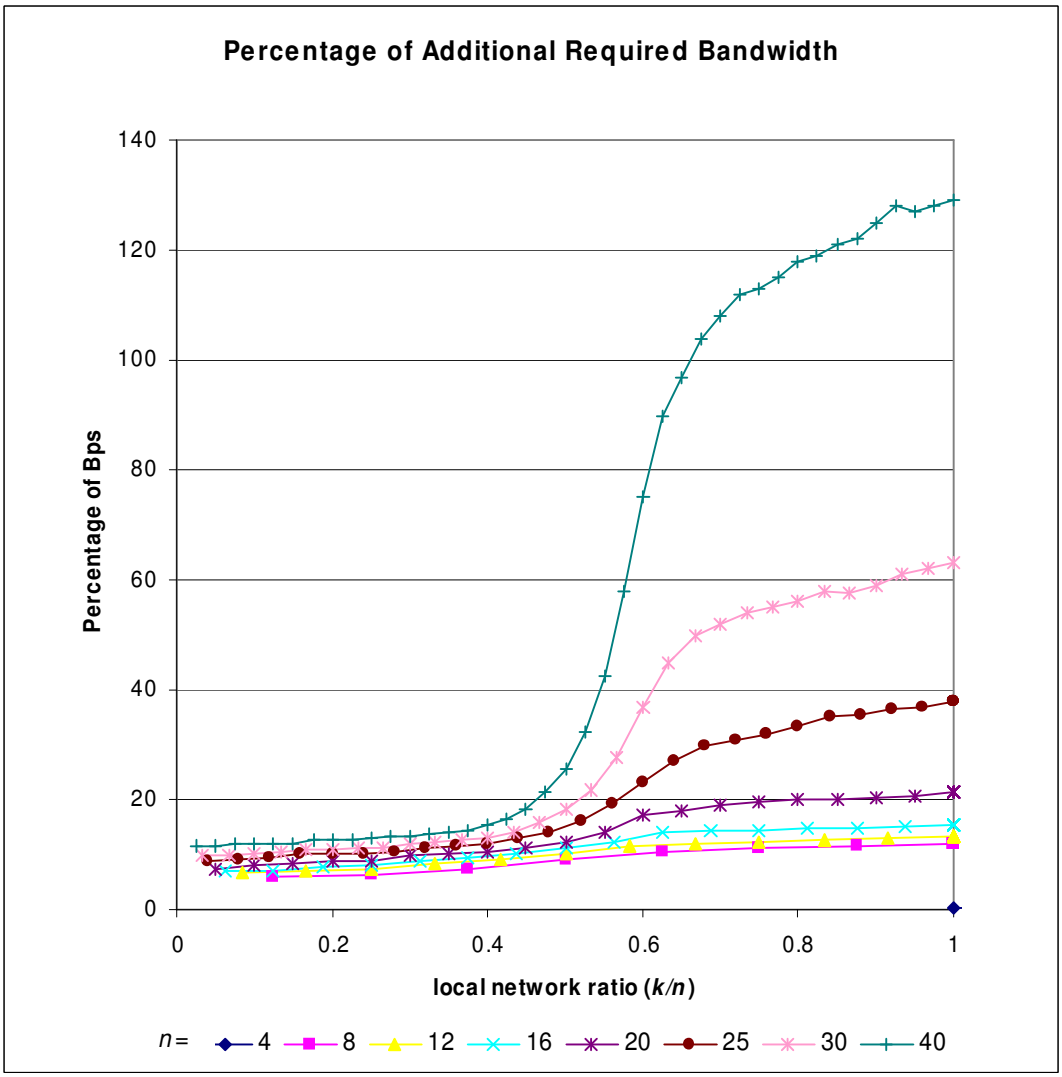


Figure 9.12: Percentage of additional bandwidth required in the large environment using the *RF* model and the *LOS* model for simbot networks of size $n = 4, 8, 12, 16, 20, 25, 30, 40$. The trends show the average for the *RF* and the *LOS* models. The x axis shows the local network in relation to the size of the team, the y axis shows the percentage of additional bandwidth required.

We conclude that the best trading between exploration time and local network size is obtained when the local network ratio is in the range n_R=[0, min($n^{-0.5}$, 2/5)]. It is also concluded that for small team sizes ($n < 40$) it is better to use the largest local network size because the exploration can be achieved faster and the additional bandwidth is not much larger. For larger teams of simbots ($n \geq 40$) it is better to use small local network sizes because for large network sizes ($n_R \geq$[min($n^{-0.5}$, 2/5),1) there is an accelerating increase in the bandwidth required as the number of simbot increases. This accelerating increase can easily surpass the capacity of available communication technologies even for small numbers of simbots.

We suspect that the range where the trading between the exploration time and the communication cost is best will be similar (± 0.05 for the maximum value of the range) in other types of environments such as open spaces with scattered obstacles or highly cluttered environments. In open spaces the simbots have more freedom to move therefore the exploration can be speeded up. In this environment the degree of connectivity of the network is larger than in office environments because there are fewer obstacles. Although the percentage of decrease in the exploration time ratio is likely to be larger due to the freedom of movement, the additional bandwidth is expected to increase faster with respect to the local network ratio because of the higher degree of connectivity, therefore the range in open spaces with scattered obstacles is expected to be similar to that of the office environment. In highly cluttered environments the exploration is slowed down because the simbots are more constrained. In this environment the degree of connectivity of the network is expected to be smaller because the obstacles will frequently block the communication between the simbots. Although the exploration is slowed down because of the clutter, the additional bandwidth is expected to have a slower increase with respect to the local network ratio because of the smaller degree of connectivity; therefore the range with the best trading in cluttered spaces is expected to be similar to that of the office environment.

9.4.1 Minimum Percentage of Time Fully Connected for a Simbot Network

Previous simulations from Section 9.3 revealed that for a certain local network size there is a minimum percentage of time that a simbot network remains fully connected regardless of the size of the network. This minimum percentage is smaller when the updates of the beacon signal are less frequent. The beacon signal is the information

sent by the simbots to their direct connections about their position.

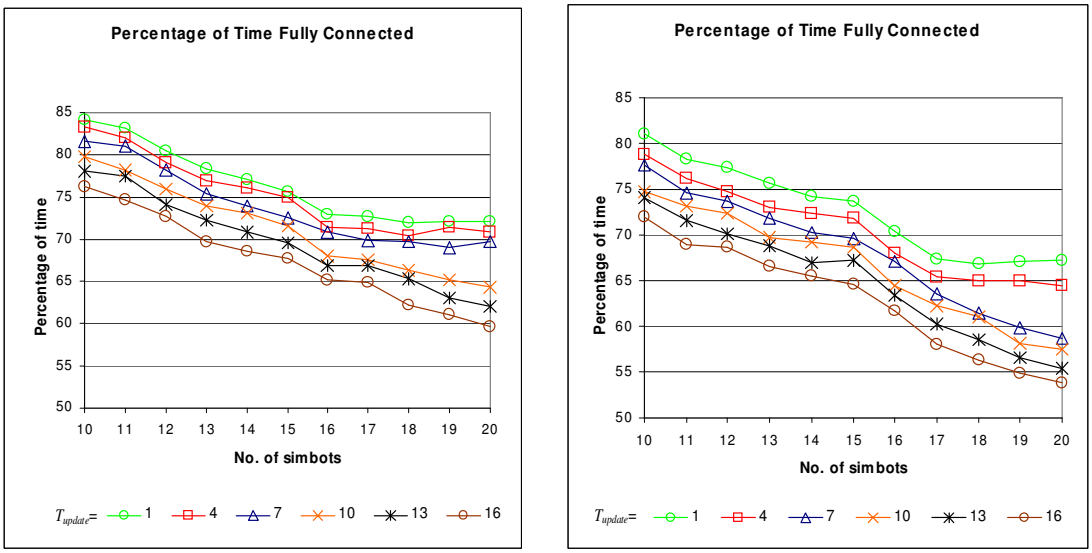
This section presents simulations to validate the existence of a minimum percentage of time fully connected for infrequent local level updates. At the local level the simbots send beacon signals (T_{update}) and updates of their features within the local network frequently. Simbots transmit extracted features at the local and global level every S_L and S_G sensing steps ($S_L < S_G$). A sensing step is the time that the simbot requires to sense an area using the *low cost* platform (Section 7.7, page 207). This process takes 1.8 seconds for the proposed *low cost* platform. The simbots sense the environment once they travelled a minimum distance. Simbots that tend to remain stationary communicate feature updates less frequently. In previous simulations the simbots transmit their features every S_L=5 and S_G=20 sensing steps. It is expected that existence of a minimum percentage of time fully connected can be validated when the beacon signals and local update of features are shared less frequently.

In Section 9.3 various levels of reactivity were compared. In the current simulations the less reactive level from those simulations is used. This level performs better when the update frequency for the beacon signals is decreased. In the simulations the size of the simbot team is $n = 10, 11, \ldots, 20$, the size of the local networks where the trading between exploration and communication cost is best is $k = 1, 2, 3$ for $n = [10, 15]$, and $k = 1, 2, 3, 4$ for $n = [16, 20]$. The frequency of the beacon signals are T_{update}=1, 4, 7, 10, 13, 16. The sensing steps at the local and global level are S_L=5, 10, 15 and S_G=20 respectively. The simulations were realized in the large environment using the *RF* and *LOS* communication models.

Figure 9.13 presents the percentage of time fully connected for local networks of size k=1 with local update S_L=15 using the a) *RF* and b) *LOS* models. This graph presents the extreme case with respect to access to information at the local level. The size of the local network is the minimum and the local features are updated less frequently. It is observed that the simbot networks using the *RF* model are easier to maintain than the networks using the *LOS* model. This confirms the findings from the previous section.

The percent of time fully connected stabilizes at a minimum value when $T_{update} <$10 for the *RF* model and when $T_{update} <$7 for the *LOS* model. For the trends that have a minimum value, the minimum value is 1.2% smaller on average as T_{update} is decreased. Figure 9.14 presents the results obtained for the *RF* model with local networks of sizes k=1, 3 and local feature updates S_L=5, 15. It is observed that the existence of a minimum percentage and its value is mainly determined by the beacon update rate T_{update}.

Decreasing the frequency of local feature updates diminishes the minimum percentage of time fully connected, but it does not affect the existence of a minimum percentage. This is expected because the main cause of disconnections in the simbot network is outdated information about the positions of the direct connections for a simbot. Less frequent local feature updates do not cause disconnections because the simbots whose main task is network maintenance tend to remain stationary when there is no updated feature information. It was observed that less frequent local feature updates increased the exploration time and caused a decrease in the consistency of the maps. The exploration time increases because the simbots tend to remain stationary for periods of time. The maps tend to be less consistent due to the delays in the integration of features received from the simbot network to the map built locally by the simbot. This problem is analyzed in Chapter 8.



a) *RF* model b) *LOS* model

Figure 9.13: Percentage of time fully connected for local networks of size $k = 1$ with local update $S_L = 15$ (sensing steps) using the a) *RF* and b) *LOS* models for update times $T_{update} = 1, 4, 7, 10, 13, 16$.

In the simulation the simbot explores the environment travelling 0.1m and stopping to sense the environment. The simbot travels at a speed of 0.15 m/s. and the sensing process takes 1.8 seconds. The average speed of the exploration process is $V_{av} \cong 0.04$ m/s. Based on this average speed we can estimate a maximum distance that the simbots can move guaranteeing that the minimum percentage is kept. The update times for the

RF and *LOS* models are T_{update} = 7s and T_{update} = 10s respectively. Based on these times and the average speed V_{av} it is determined that the maximum distances that the simbots in BERODE-2 can move guaranteeing that the minimum percentage is kept are d_{travel} = 0.28m and d_{travel} = 0.16m for the *RF* and *LOS* models. In scenarios where the maintenance of connectivity is vital or the available bandwidth is limited the designer may decide to slow down the exploration guaranteeing the maintenance of the network. This experimental simulation provides the basis for determining the desired exploration speed.

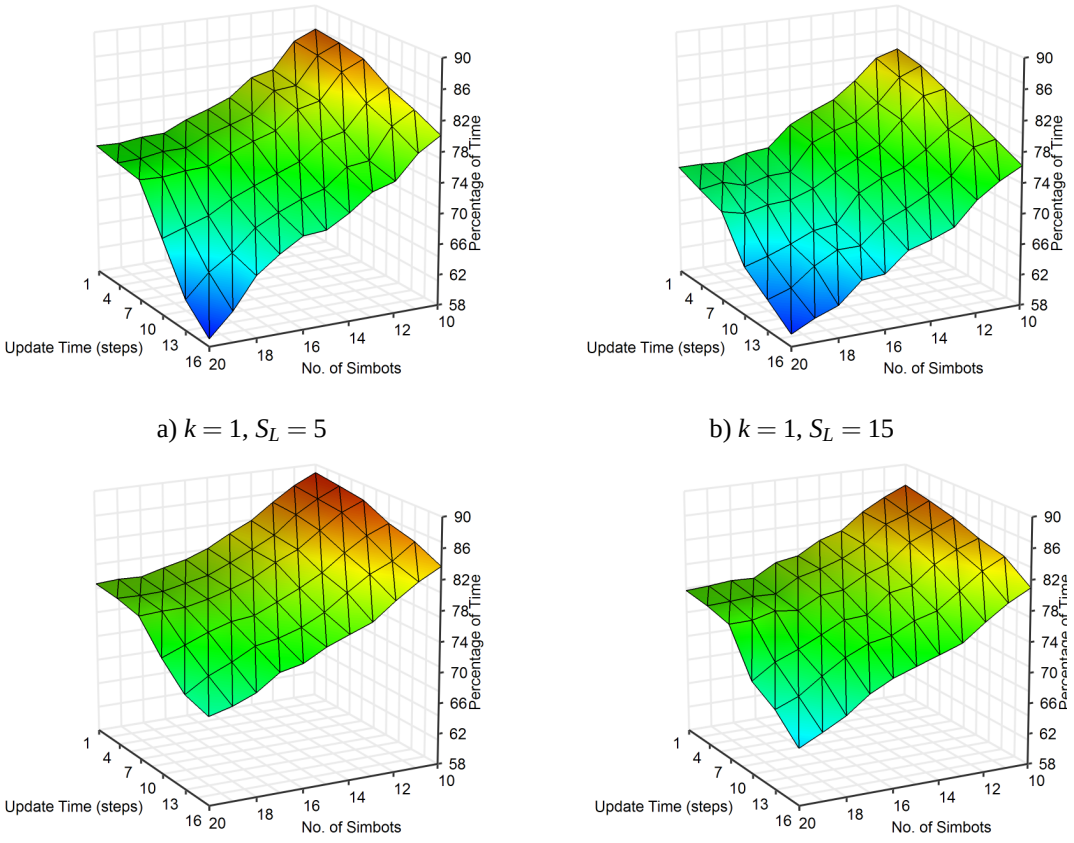
We conclude that BERODE-2 will rarely fail to achieve a minimum percentage of time as a fully connected network when the distance that the simbots move between updates is small. This applies to all the local network sizes because local networks outside the optimal range have larger sizes of local networks therefore the amount of knowledge obtained from the network is larger. We argue that the existence of the minimum percentage is strengthened when more knowledge is obtained. Moreover, this argument is supported by the simulations from Section 9.3.

9.5 The Effect of the Communication Range in the RF Model

Previous simulations have shown that as expected *LOS* communication restricts the exploration more than *RF* communication. In *LOS* communication the simbots have to spend more time maintaining and fixing the control network.

In previous simulations the communication range has been assumed to have a fixed value. The communication range is an important feature in a multirobot system. Although having robots with an effectively unlimited communication range may seem in principle attractive it is undesirable for the purposes of scalability. The bandwidth required increases as more robots are added to the network. Moreover when power consumption is a constraint in the system it is desirable to restrict communication to short distances because the amount of power required for transmitting a signal in free space is a function of the square of the distance. In indoor environments the situation is much worse because of the structure and the obstacles. It is therefore important to assess the effect of the communication range in the performance of the system.

In this section we present simulations to assess the effect of the communication range on the performance of BERODE-2. We expect to identify a trade-off between



a) $k = 1$, $S_L = 5$ b) $k = 1$, $S_L = 15$

c) $k = 3$, $S_L = 5$ d) $k = 3$, $S_L = 15$

Figure 9.14: Percentage of time fully connected for local networks of size $k = 1,3$ with local update $S_L = 5,15$ (sensing steps) using the *RF* model. For update times $T_{update} < 10$ the minimum value stabilizes with respect to the number of simbots.

the communication range and the exploration time for the simbot networks. We expect to identify a minimum value for the communication range for which the exploration time is the same as that of larger communication ranges.

Previous research (Burgard et al., 2006) has addressed the effect of the communication range in multirobot exploration for a centralized architecture. The robots were not constrained to remain within communication range. The robots formed communication clusters and one agent decided which areas to explore for all the robots in the cluster. The experiments were carried out in an office environment with three real robots and in simulation with teams from 1 to 5 robots. The research revealed that the exploration time is the same when the communication range is above 30% of the diameter of the environment for all the team sizes used in their simulations (the maximum team size was five robots). The communication model assumed no interference

from the obstacles or environmental structure. We argue that the 30% ratio between the communication range and the diameter of the environment model does not apply to teams larger than five robots because as more robots are added to a team the area covered by the signals of the robots increases. The communication range necessary to have the same exploration time is likely to decrease when the size of the team is increased. Moreover, we argue that the communication range is not a good metric to assess the relation between the communication range and the exploration time when a more realistic model for *RF* communication is used. In our *RF* model the interference from the obstacles and the environmental structure is modelled. We propose to analyze the effect of the communication range based on the percentage of covered area. The percentage of covered area is the average of the percentage of the area of the environment that a signal with a certain communication range covers. This percentage is different for each environment because it depends on the degree of clutter present in the environment. For the environments used in our simulations this value was calculated by sampling the grid space of free positions with a resolution of 0.2m.

The communication ranges compared in the medium and large size environments were R=3.5, 5, 6.5, 8, 9.5, 11,∞ and R=3.5, 5, 6.5, 8, 9.5, 11, 12.5, ∞ respectively. Table 9.2 and Table 9.3 present the percentages of covered area for the medium and large size environments. The areas of the medium and large environments are 192 m^2 and 320m^2 respectively. The *RF* area is the area that the signal covers in the free space environment. It is observed that the percentage of area covered is much smaller than the area that the signal could cover if the interference from the obstacles was not modelled.

The settings for the simulations are the same as those used in the simulations of Section 8.6 (page 239). The medium and large environments with the *RF* model were used in these simulations. The size of the simbot team was $n = 4,6,\ldots,16$ and $n = 10,12,\ldots,20$ in the medium and large environments respectively. The size of the local networks was $k = 1,2,3$.

We expected to identify a minimum value for the area covered after which the exploration time would be the same for the simbot network. It was expected that this value would be smaller for larger simbot networks because as more simbots are used in the exploration the total area covered by the simbot network is bigger.

Figure 9.15 presents the exploration time with different network sizes for the medium environment. It is observed that the trends have a logarithmic decrease. The decrease is larger for simbot networks of smaller sizes. It is observed that there is a

***RF* range (m)**	3.5	5	6.5	8	9.5	11	∞
***RF* Area (m^2)**	28.48	78.54	132.73	201.06	283.53	380.13	∞
Area covered (%)	8.23	14.12	24.56	35.94	52.52	65.73	100

Table 9.2: Percentage of covered area in the medium environment for various *RF* communication ranges. The environment is represented as a grid space. The covered area was obtained by sampling the free positions of the environment with a resolution of 0.2m.

RF range (m)	3.5	5	6.5	8	9.5	11	12.5	∞
RF Area (m^2)	38.48	78.54	132.73	201.06	283.52	380.13	490.78	∞
Area covered (%)	4.51	8.22	13.45	20.13	32.13	39.14	43.12	100

Table 9.3: Percentage of covered area in the large environment for various *RF* communication ranges. The environment is represented as a grid space. The covered area was obtained by sampling the free positions of the environment with a resolution of 0.2m.

threshold value (T_{area}) for the percentage of area covered after which the exploration time is the same. T_{area} is smaller for larger simbot networks. For the sizes of the simbot networks shown the threshold is in the range T_{area}=[25%, 40%]. Figure 9.16 presents the exploration time for the large size environment for the simbot networks with local network sizes of k=1, 3. It is observed that the results are very similar. In Figure 9.16 the covered area threshold is in the range T_{area}=[25%, 40%] for a local network size k=1 and T_{area}=[20%, 35%] for a local network size k=3.

From these figures it is observed that the thresholds for the different simbot network sizes have a linear increasing trend with respect to area covered. This linear trend is shown in the figures. We argue that these linear trends have roughly the same slope. To prove this argument we propose a linear fitting for the points in the polynomial trends for which the exploration time stabilizes. The slope for these points is calculated using a simple genetic algorithm. To calculate the slope first a least squares polynomial fitting for the simbot network trends was performed for all the available data. Table 9.4 shows the polynomial fitting for the trends from Figure 9.15. For the medium environment there are 7 polynomials per local network size while for the large size environment there are 8 polynomials per local network size. The total number of polynomial functions is 45.

We used a genetic algorithm because we can easily compute the average slope from the multiple solutions of the linear fitting. In the genetic algorithm the information encoded is the value for the slope. The fitness function is calculated as follows:

1. For a given local network size the linear fitting function is $y = mx + b$. m is the slope value from the genetic algorithm. The value of b is determined by finding the interception between the linear fitting function and each of the polynomial functions for the local network size. There are l=7 and l=8 different solutions for $b = \{b_1, ..., b_l\}$ in the medium and large environments respectively.

2. The fitness value for a given local network size measures the tangent of the polynomials at the interception point between the linear fitted function and the polynomial. The fitness value results from the addition of all the tangents and is defined as:

$$Fitness = \sum_{j=1}^{l} \sum_{i=1}^{l} \left| \frac{f_i(x_{j,i} + \Delta x) - f_i(x_{j,i} - \Delta x)}{2\Delta x} \right| \tag{9.1}$$

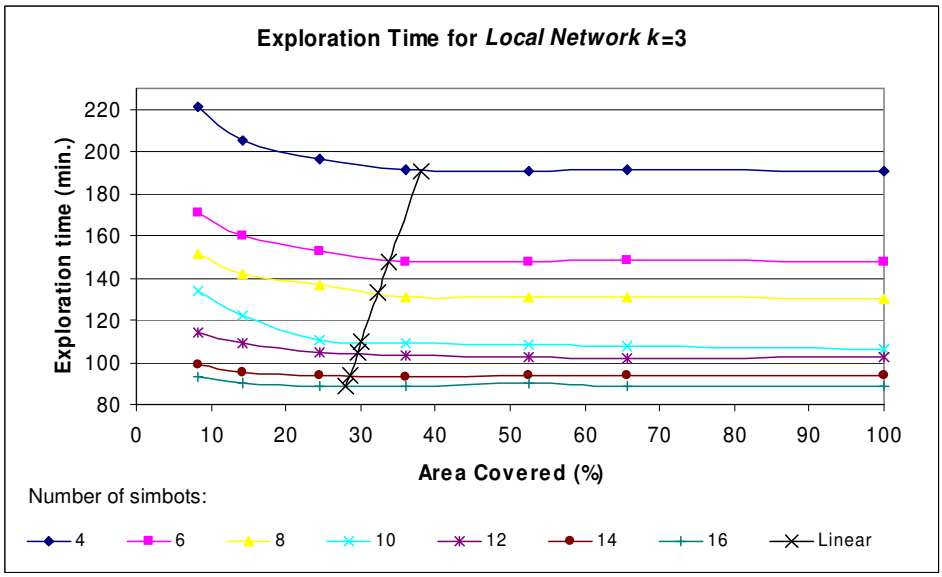


Figure 9.15: Exploration time for simbot networks with different communication ranges using the *RF* model for the medium environment. The x axis shows the percentage of area covered by different communication ranges. Several trends for different sizes of simbot networks are presented. It is observed that the trends have similar shapes. The trend labelled as "linear" shows the threshold for the area covered after which the exploration time is the same for a certain number of simbots.

$f_i(x)$ is the i^{th} polynomial function. The variable $x_{i,j}$ is the value of x (percentage of covered area) at the intersection point between the i^{th} polynomial function and the linear function with slope m and $b = b_j$. Δx is the interval used to estimate the tangent

No of Simbots	Polynomial Regression	Residual R^2
4	$y = 545.11x^4 - 1342.4x^3 + 1173x^2 - 433.83x + 248.58$	0.9891
6	$y = 304.33x^4 - 786.62x^3 + 731.49x^2 - 291.16x + 189.88$	0.9933
8	$y = 146x^4 - 428.96x^3 + 455.01x^2 – 206.09x + 164.55$	0.9866
10	$y = 576.01x^4 - 1382.6x^3 + 1165.6x^2 - 413.42x + 160.88$	0.9843
12	$y = 240.32x^4 - 556.7x^3 + 457.03x^2 - 163.09x + 124.74$	0.9932
14	$y = 179.91x^4 - 419.42x^3 + 339.23x^2 - 111.57x + 105.86$	0.9799
16	$y = 254.07x^4 - 553.56x^3 + 405.1x^2 - 116.78x + 100.18$	0.9815

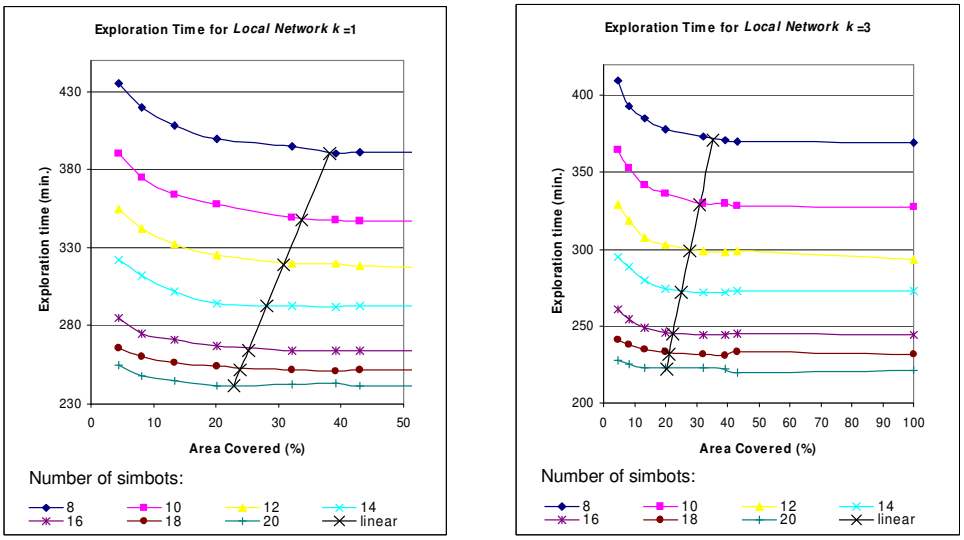
Table 9.4: Polynomial functions for the trends from Figure 7.14. x is the percentage of covered area and y is the exploration time (min.) R^2 is the residual of the least squares polynomial fitting.

for the polynomial function. We use a value of $\Delta x = .01$. The total fitness value is the sum of the fitness values for the different local network and environment sizes.

The value of the slope was 84.41^o with an average tangent value of -0.00872 (with standard deviation of 0.25^o) at the intersection points for the polynomials. The average value is approximately zero (stabilization value for the polynomials) and the standard deviation is small therefore we conclude that the slope is the same for the different local network and environment sizes. Figure 9.15 and Figure 9.16 show the slope from the linear fitting for one of the values of *b*.

It is concluded that the percentage of area covered by a simbot signal is in the range T_{area}=[25,40] where as expected for larger sizes of the simbot network the coverage necessary to have the same exploration time for the signal decreases. This decrease is linear with a slope of 84.41^o. This finding is relevant for applications in which the approximate size of the environment and the degree of clutter is known *a priori*; for instance in an exploration task to update an outdated map. Moreover, the simbots could potentially adjust their communication range dynamically by estimating the percentage of area that their signal covers according to their currently built map. This dynamic adjustment could be useful to improve power consumption while minimizing the interference in communication.

Even though the value for the slope (84.41°) is close to 90° there is a large difference in the percentage of area covered required to have the same exploration time for different simbot network sizes.



a) local network size $k = 1$ b) local network size $k = 3$

Figure 9.16: Exploration time for simbots networks with different communication ranges using the *RF* model for the large environment for local network sizes a) $k = 1$ and b) $k = 3$. The x axis shows the percentage of area covered by different communication ranges. Several trends for different sizes of simbot networks are presented. The trend labelled as "linear" shows the threshold for the area covered after which the exploration time is the same for a certain number of simbots.

9.6 Comparison with Fixed Simbot Networks

Previous approaches (Sweeney et al., 2002; Kannan et al., 2003) in the maintenance of simbot networks rely on a fixed simbot network configuration. Based on this simbot network *leader-follower* relations are imposed. The simbots form a chain of *leader-follower* relations that constitute a column topology. All the simbots with one exception are follower simbots. The simbot without a leader directs the exploration while the rest of the simbots maintain the network and if possible explore the nearby space. These approaches have been applied successfully to small team sizes. We argue that this type of approach relying on fixed simbot configurations can be improved for exploration purposes. We will now compare BERODE-2 against several fixed networks to assess its advantages and disadvantages.

In BERODE-2 the simbots recalculate the MST control network either partially or globally. It is argued that the recalculation of the network aids the exploration process because the signal quality of the MST connections is improved. To determine the effectiveness of this adaptability in the MST control network, we propose to compare

the performance of the adaptive MST control network against a fixed MST control network. The fixed network is calculated at the beginning of the exploration and remains the same through all the exploration process. In the initial configuration all the simbots are within communication range of at least another simbot, however not all the simbots are within range of each other. The implemented fixed networks use the BERODE-2 architecture.

Three types of fixed networks are proposed for the comparison: maximum, minimum and *k*-connections connectivity. The maximum connectivity tries to create star like topologies whereas the minimum connectivity tries to create column like topologies. The *k*-connections connectivity tries to create a network in which all the simbots have *k* connections.

The fixed networks are calculated using Dijkstra's algorithm which builds an MST (Cormen et al., 1990) of the communication network. This algorithm iteratively adds the edge (connection) with the smallest weight that connects a visited node (simbot) with a non-visited node. Every time an edge is added the non-visited node is labelled as visited. The algorithm stops once all the nodes are labelled as visited. In the maximum connectivity type the weight of an edge is the negative value of the number of added connections. In the minimum connectivity type the weight of an edge is the number of added connections. In the *k*-connections type the weight of an edge is the absolute difference between the number of added connections and *k*.

The settings for the simulations are the same as those of Section 8.6 (page 239). The networks were compared using the medium and large environment from Figure 7.1 (page 164) using the *RF* and *LOS* models. The size of the team of simbots was $n = 4,...,16$ and $n = 8,...,20$ in the medium and large environments respectively. The local network sizes were $k = 2,4,...,n/2$ for each team size. Previous simulations revealed that for $k > n/2$ the local network tends to be the same as the global network most of the time. The performance for these local network sizes tends to be the same. Ten trials were run for each combination of simbot and local network size. The QS predictive and the pos. predictive heuristics are used to calculate the MST control networks in the *RF* and *LOS* communication models respectively. For the fixed networks the information is transmitted always at the global level.

Two metrics are proposed for the comparison: the speedup factor and the percentage of time fully connected. The speedup factor is the ratio between the exploration times for a single simbot network and a simbot network of a certain size. The exploration time is the time that the simbots required to build a complete map of the

environment. As explained in Section 9.1 the exploration stops once any one robot considers the map to be complete. The speedup factor therefore describes the scalability of the control approach as the number of simbots in the network is increased. For a simbot network of size n the ideal speedup is n.

It is expected that the simbots that implement BERODE-2 (adaptive network) will have better speedup factors than the simbots using the fixed networks. It is also expected that the percentage of time fully connected for BERODE-2 will be higher than for the fixed networks.

Figure 9.17 – Figure 9.19 present the comparison of the speedup factor for BERODE-2 for different local network sizes (BERODE-2k) against four fixed networks: maximum connectivity (*MAX*), minimum connectivity (*MIN*) and $k-$connections for branching factors k=3,5 (*KC* k=3 and *KC*=5 respectively). The average branching factor (*ABF*) for the fixed networks was: 7.8 for the maximum connectivity type, 1.2 for the minimum connectivity type, 2.9 for the k-connections type for k=3 and 5.1 for the k-connections type for k=5.

In the simulations with the *LOS* model the fixed networks frequently got stuck and failed to finish the exploration. These trials were not considered in the results and the trials were repeated. This situation was even worse in the large size environment where only in a few trials the minimum connectivity type managed to build the map. This is attributed to the larger presence of clutter in the environment in the large environment. This increases the difficulty of the task to maintain line of sight for the communication constraints.

The fixed networks finished the exploration in 95.2% of the trials when the *RF* model was used, while finishing only in 60.2% of the trials when the *LOS* model. The general success rate for the fixed networks was 70.3% BERODE-2 finished the exploration in all the trials when the *RF* model was used and in 92% of the trial when the *LOS* model was used. The general success rate was 95.6%

From Figures 9.17 to 9.19 it is observed that:

- As expected regardless of its local network size BERODE-2 has better speedup factors than the fixed networks. The closest speedup factor for a fixed network was 8.39% worse on average compared to BERODE-2 with k=3. This closest factor was for the minimum connectivity type in the medium environment for the *RF* model.

- Fixed networks with smaller average branching factors (*ABF*=1.2, 2.9) have bet-

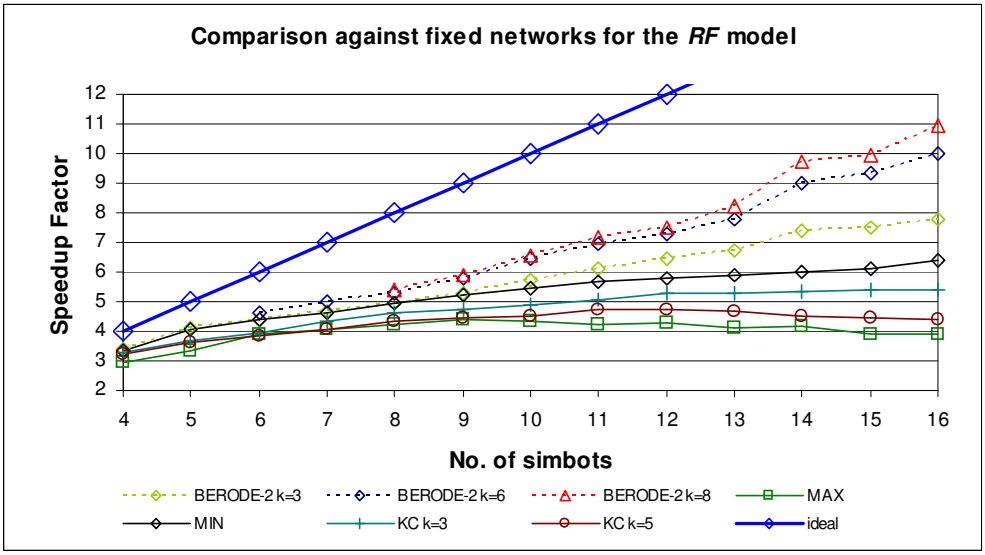


Figure 9.17: Comparison of the speedup factor for BERODE-2 with local network sizes $k = 3, 6, 8$ against fixed networks using the *RF* communication model in the medium environment. The dashed lines are the trends for the BERODE-2 architecture. The maximum possible speed up factor is shown as reference.

ter speedup factors than those with larger branching factors (*ABF*=5.1, 7.8).

- Fixed networks with small branching factors (*ABF*=1.2, 2.9) using the *RF* model have a linear increase in the speedup factor with respect to the number of simbots up to a certain number of simbots. Afterwards there is only a slight increase if not zero increase in the worst case. For instance in Figure 9.18 for the minimum connectivity type for more than 16 simbots the speedup factor is the same.

- In the *LOS* model the speedup factor for the fixed networks with small branching factors tends to worsen for medium size teams ($n > 7$). Adding more simbots increases the time required to build the map instead of diminishing it.

- The speedup factor for fixed networks with large branching factors (*ABF*=5.1, 7.8) decreases as the size of the time increases for teams larger than a certain minimum size. This size seems to depend on the communication model and the environment configuration. In cluttered environments with a *LOS* model the size is likely to be very small. For instance from Figure 9.19 it is observed that for the maximum connectivity type this value is r=4.

Figure 9.20 and Figure 9.21 present the percentage of time fully connected using the *RF* model in the medium and large environments. From these figures it is observed

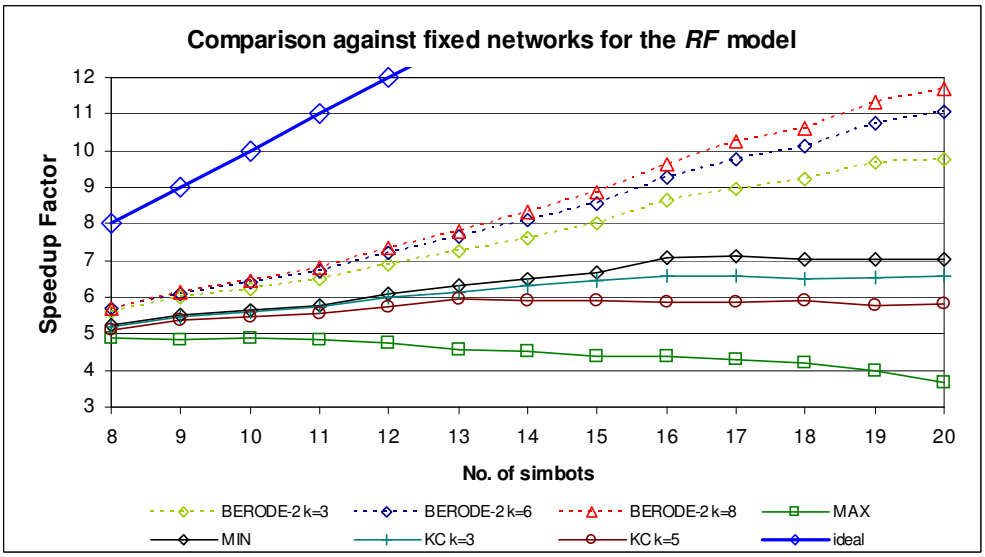


Figure 9.18: Comparison of the speedup factor for BERODE-2 with local network sizes $k = 3, 6, 8$ against fixed networks using the *RF* communication model in the large environment. The dashed lines are the trends for the BERODE-2 architecture. The maximum possible speed up factor is shown as reference.

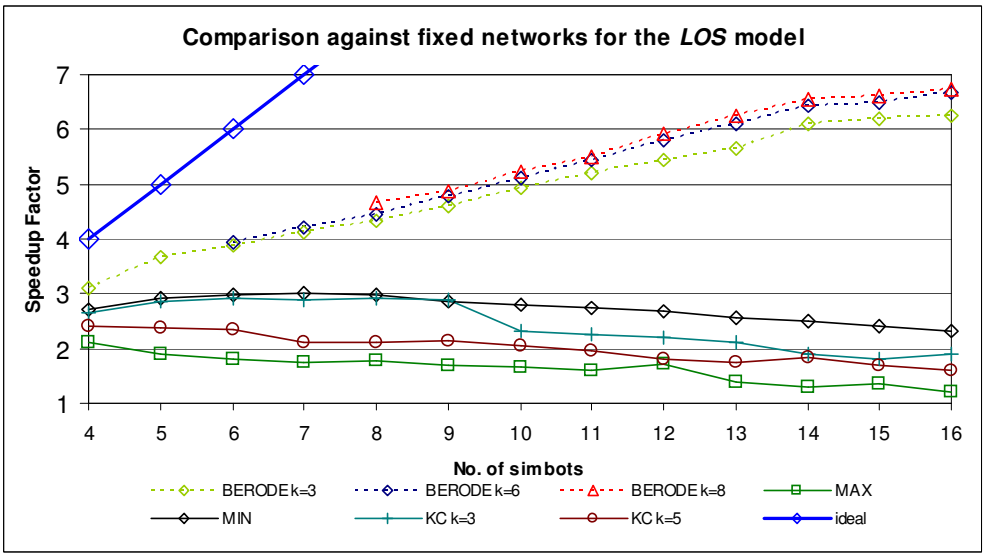


Figure 9.19: Comparison of the speedup factor for BERODE-2 with local network sizes $k = 3, 6, 8$ against fixed networks using the *LOS* communication model in the medium environment. The dashed lines are the trends for the BERODE-2 architecture. The maximum possible speed up factor is shown as reference.

that:

- The percentages are smaller ($5.42 \pm 0.3\%$ on average) for all the compared networks in the large environment with respect to the medium environment. This is attributed to the largest presence of clutter in the large environment. In the presence of clutter the simbots rely more on the predictive model used in the Planning Module (Section 5.6, page 102). The accurate prediction of signals that go through obstacles is more difficult to achieve than the prediction of signals with direct *LOS*.

- Regardless of the local network size in BERODE-2 the percentage of fully connected time slowly decreases in a linear fashion as the number of simbots increases until a certain simbot network size is reached. Afterwards the percentage of time stabilises. This observation confirms the results from the simulations from Sections 9.3 and 9.4.1. In these simulations it was found that the minimum percentage depends on the frequencies of the beacon and local updates (Section 9.4.1). A beacon update is the position update that the simbots send to their direct connections. The local updates are the features transmitted within the local network periodically.

- The minimum connectivity type maintains similar if not better percentages than BERODE-2. In the medium size environment this type has the highest percentages for any team size ($0.78 \pm 0.2\%$ better than BERODE-2 with k=8). In the large environment this type maintains similar percentages to those of BERODE-2.

- For the fixed networks the average branching factor and the percentage of time fully connected are inversely related. This is expected because the risk of becoming disconnected increases when a simbot has to remain connected with more simbots simultaneously.

- For the fixed networks there is no minimum percentage of time fully connected as in the case of BERODE-2. However the decrease tends to be logarithmic.

From these simulations we conclude that:

- As expected BERODE-2 has a better performance than the fixed networks. The improvement is larger in difficult scenarios for communication. Scenarios are

difficult when the communication model is *LOS* constrained or in the presence of significant clutter as in the large size environment. In cluttered scenarios the simbots rely more in the predictive model used in the predictive planning module (Section 5.6, page 102). The accurate prediction of signals that go through obstacles is more difficult to achieve than the prediction of signals with direct *LOS*.

- BERODE-2 has a much higher success rate (95.6% against 70.3%) finishing the exploration task than the fixed networks.
- Fixed networks of the minimum connectivity type have good performance for small team sizes when the *RF* communication model is used. The performance is good because for medium team sizes ($n \leq 12$):
  - The percentage of time fully connected is better than if not the same as for BERODE-2. This result is not surprising because in the fixed networks the simbots maintain all the time the same control network, whereas in BERODE-2 when the network is modified there is a transition period where the network might become disconnected because of information delays.
  - The speedup factor is close to that of BERODE-2 ($8.39 \pm 1.2\%$ smaller on average) in the medium environment. The speedup factor is $14.01 \pm 3.7\%$ smaller on average in the large environment for team sizes $n \leq 16$.

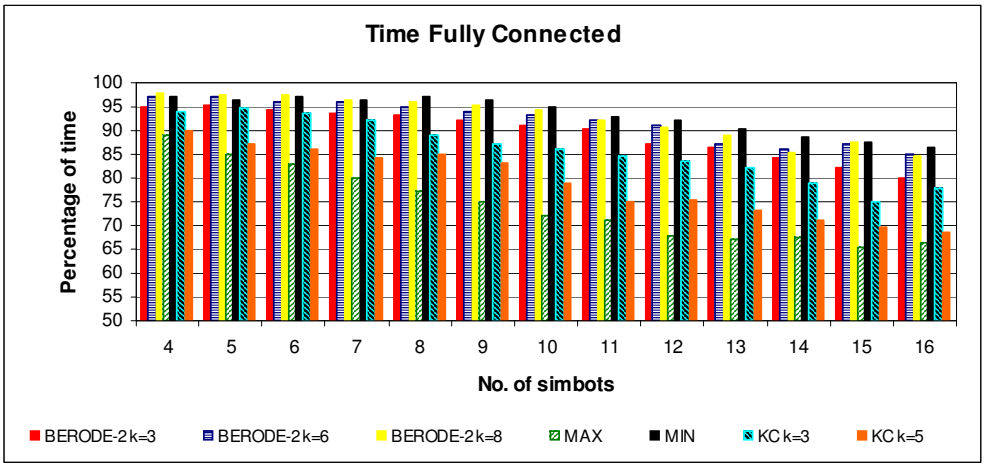


Figure 9.20: Comparison of the percentage of time fully connected for BERODE-2 with local network sizes $k = 3, 6, 8$ against fixed networks using the *RF* communication model in the medium environment.

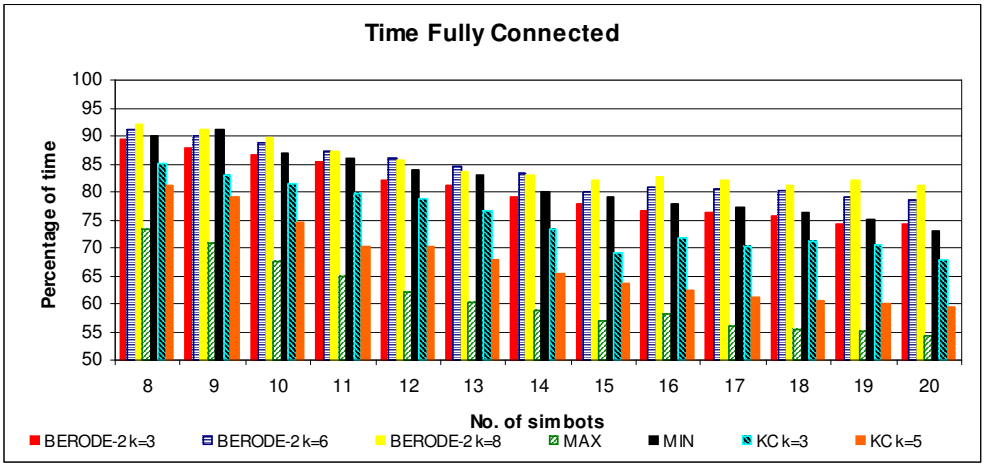


Figure 9.21: Comparison of the percentage of time fully connected for BERODE-2 with local network sizes $k = 3, 6, 8$ against fixed networks using the *RF* communication model in the large environment.

Although not as efficient as BERODE's adaptive networks, fixed networks with column like control formations are a good solution for simbot networks of medium sizes using RF technologies. These networks are suitable for indoor environments with small clutter and might be preferred over BERODE because of their simplicity. Our findings suggest that these networks are suitable for outdoor environments and environments with scattered obstacles where the communication conditions are more predictable.

9.7 Conclusions

This chapter has demonstrated in simulation the robustness and scalability properties of BERODE-2 using the *LOS* and *RF* communication models. We cannot guarantee that these properties will transfer to a real world implementation but our simulations have been validated in relevant aspects for the robot sensors and the communication models. The experiments of Chapter 7 show that our simulations are reasonable and conservative approximations to the experimental data.

The simulations show that BERODE-2 is robust with respect to infrequent updates at the local level. BERODE-2 is robust because the time that the simbots remain as a fully connected network slowly decreases as the simbots receive less frequent updates. Moreover, when the frequency of the updates is above a certain value there is a minimum percentage of time that a simbot network of any size is guaranteed to re-

main fully connected. This minimum depends on the size of the local network and the frequency of the local updates. Chapter 8 presents simulations that demonstrate that the frequency of the updates at the global level does not affect the robustness of BERODE-2.

The simulations show that BERODE-2 is scalable because as more simbots are added to the network the exploration time decreases up to a certain number of simbots. The decrease is logarithmic at worst and confirms the findings from previous simulations (Burgard et al., 2006) that suggests that for a certain size environment there is a maximum number of simbots that can efficiently explore the environment.

BERODE-2 is scalable in terms of communication cost because there is an inversely proportional trade-off between the time required to build the complete map and the network ratio. The network ratio is the ratio between the sizes of the local network and the simbot network. For small network ratios there is a linear increase in the bandwidth required with respect to the number of simbots. For small network ratios the trade-off between the time required to build the map and the communication cost is the best.

For small numbers of simbots (in our simulations $n < 40$) it is better to use the largest local network size because the exploration can be achieved faster and the increase in the additional bandwidth is small. For larger teams of simbots (in our simulations $n \geq 40$) it is better to use small local network sizes because there is an accelerating increase in the bandwidth required as the number of simbots increases which can easily surpass the capacity of available communication technologies even for small numbers of simbots.

Previous research (Sweeney et al., 2002; Kannan et al., 2003) has implemented fixed control networks to maintain the robot network fully connected. These networks have been applied successfully in the exploration of environments using small robot teams. Our simulations show that BERODE-2 has a better performance than these fixed control networks because it has better speedup[5] factors than the fixed networks. The closest speedup factor for a fixed network was $8.39 \pm 1.2\%$ worse on average compared to BERODE-2 with a local network size k=3. Generally speaking BERODE-2 maintains the network fully connected more time than the fixed networks. 5.7% on average with respect to the best fixed network which is the minimum connectivity type. In this fixed network the simbots tend to form column like topologies. BERODE-2

[5] The speedup factor is the ratio between the exploration times for a single simbot network and a simbot network of a certain size.

has a much higher success rate (95.6% against 70.3%) at finishing the exploration task than the fixed networks.

The simulations from Section 9.2 revealed that although there is only a slight increase in the time required to build a complete map when the updates at the local level are less frequent, there is a tendency to build maps with repeated features. This is caused by failures in the matching of the external features with the current feature map of the simbots. The computational cost of updating the map is increased because of the repeated features. This issue is analyzed in following chapter. The following subsection presents a summary of the main contributions of this chapter.

9.7.1 Contributions

The main contributions from this chapter are:

- The determination of the best heuristic from several proposed heuristics to calculate the MST control network using the *RF* and *LOS* models. The QS predictive and the Pos. predictive are the best heuristics for the *RF* and *LOS* models respectively. These heuristics are suitable for scalability purposes because as more simbots are added to network the time required for building the complete map decreases. The QS heuristic has as its main quality that the movement required by the simbots to keep the network connected is invariant with respect to the size of the local network. This is a desirable quality when power consumption is a constraint in the system because the simbots spend less energy. The Pos. predictive heuristic is the best applicable heuristic for the LOS model. This heuristic is sensitive to the local network size. The movement required by the simbots to maintain the network increases when the size of the local network is decreased.

- The determination of a range of values for the size of the local network where the trading between communication cost and exploration time is the best. This range is n_R=[0, min($n^{-0.5}$, 2/5)] and is defined as a function of the network ratio. In this range the exploration time and the additional bandwidth are inversely proportional. The additional bandwidth is the increment in bandwidth with respect to the bandwidth for the minimum size of local network. Outside the best trading the additional bandwidth increases in a polynomial fashion with an order of at least O(n^2) and the exploration time tends to be the same as the size of the local network is increased. For small numbers of simbots ($n < 40$) it is better to use

the largest local network size to achieve a faster exploration. For large numbers of simbots it is better to use small local network sizes due to the fast increase in the bandwidth required when the size of the local network is large.

- The experimental proof of the existence of a minimum percentage of time as a fully connected network for a simbot network of any size given a certain local network size. The minimum will rarely fail to be achieved when the distance that the simbots move between beacon updates is small ($<$0.30m) with simbots that move at an average speed of 0.15 m/s in office like environments.

- The identification of a communication range threshold whose signal covers a minimum area in the environment for the *RF* model. The area covered by a signal with a certain range depends on the clutter of the environment. Communication ranges above the threshold require the same time to build a complete map of the environment. Based on this threshold the simbots could potentially adapt their communication range on the run to have an efficient management of the power consumption and avoid interference between communication signals. Cheap communication technologies such as Bluetooth already include power consumption management.

- The demonstration of the advantages of adaptability on the MST control network over fixed control networks. The adaptive networks have much better speedup factors than the fixed networks and keep the network fully connected for more time than the fixed networks. Moreover, they have higher success rates at finishing the exploration task.

Chapter 10

Map Consistency in BERODE-2

10.1 Introduction

A central requirement of the exploration task is that the maps generated by the robots are consistent. This chapter presents simulations to analyze the consistency of the maps built by the simulated robots in the BERODE-2 (BEhavioural ROle DEcentralized) architecture. The purpose of the analysis is to determine if BERODE is scalable with respect to map consistency. We use the term consistency to refer to the amount of similarity between a pair of maps. If the robots have a large number of inconsistencies in their maps they are less likely to achieve decentralized coordination, which is the main goal in BERODE-2. Therefore BERODE-2 can be regarded as scalable with respect to map consistency if the maps have the same if not better consistency as the number of robots is increased. If map consistency imposed a scalability limit on BERODE-2 then additional mechanisms to improve consistency, such as further exploration, would be required to allow the decentralized coordination of the robots.

To have a more meaningful consistency analysis relevant measured aspects of the robot sensors and communication devices have been included in the simulated models (Chapter 7). Throughout this chapter we will refer to our simulated robots as simbots.

In BERODE-2 all the simbots remain within communication range of at least one simbot. The simbots form a communication network that is kept as a fully connected network by creating and updating an MST (Minimum Spanning Tree) control network. The MST control network contains only the minimum links necessary to have a fully connected network.

In BERODE-2 each simbot builds its own feature based map of the environment. An *EKF* (Extended Kalman Filter) is used to update the estimates and uncertainty of

the robot and features locations. The simbots start off at known locations and share a common global coordinate system, which is the initial position of the simbot with the smallest ID number. Having a common frame of reference allows the exchange of extracted features (Section 5.8, page 119).

Depending on their origin the features are either internal or external. Internal features are extracted by the simbot from its raw sensor data. External features are features that a simbot receives from other simbots in the simbot network. These features are handled by the Map Interface Module (Section 5.8, page 119). Two types of features are received: Local and Global. Local features are used to aid navigation but they are not integrated in the local feature map. Local features are transmitted within the local network. The local network for a simbot is the network that contains all the simbots within a *k-hop* distance on the MST control network. The Global features are integrated in the feature map and are transmitted to all the simbots.

The simbots are based on the *low cost* platform proposed in Chapter 7. The platform uses inexpensive sonar and infrared sensors to map the environment. The simbots build a feature based representation of the environment. The Map Interface Module incorporates *a priori* structural knowledge (e.g. parallel walls, perpendicularity, etc.). The maps generated by the simbots frequently contain small differences. The exploration is not impaired by small differences in the simbots' maps. Large differences may create inconsistencies in the simbots' maps that cause the simbots to take different decisions. This chapter presents simulations to determine the consistency of the maps built by different simbot network sizes. To be able to identify the types of inconsistencies that cause poor performance we propose to compare the feature maps of the simbots directly and through their projected grid and topologic representations. We expect to have consistent maps at the topologic level and small differences at the grid level. Section 10.2 presents the metrics used in the three comparisons.

Section 10.3 presents an analysis of the scalability of BERODE-2 with respect to map consistency. It is shown that BERODE-2 is scalable because as more simbots are added to the simbot network the maps become more consistent. The maps are more consistent at all the levels (feature, metric and topologic) of the map. In the feature comparison it is observed that there are a few repeated features in the maps. The number of repeated features increases as the number of simbots increases. It is found that these repeated features appear in the maps because there are delays in the integration of the external features.

This problem is analyzed in Section 10.3.1. The problem is found to be specific to

the current implementation in which the features transmitted by the simbots are stored in the Map Interface Module (Section 5.8, page 119). This module stores, matches and updates the recent observations of features since the last time the features were transmitted globally. The module does not incorporate the later corrections realized by the *EKF* to these features. This causes failures in the matching process. Unmatched features are added to the map as new features, generating a surplus in the number of features.

In Section 10.3.2 the use of an *EKF* based on local maps is proposed as a solution to this problem of repeated features. The computational and communication costs of this solution are analyzed in this section. The implementation of these local maps is beyond the scope of this thesis and remains as a future area of research for BERODE-2.

The problem of repeated features is analyzed in Section 10.4 to determine the usefulness of the *EKF* global implementation when the global features updates are less frequent. It is concluded that given its ease of implementation and its low additional cost in computational terms the current global *EKF* implementation is suitable for medium simbot network sizes ($n \leq 14$)[1]. Nonetheless the implementation is not robust to infrequent updates because the number of repeated features increases when the updates are less frequent. This is unlikely to be a problem in practice unless a low bandwidth communication technology is chosen for the implementation of BERODE-2.

The estimated features frequently contain temporary errors (e.g. inaccurate estimation of the extremes of the lines) that are removed with map management mechanisms (Section 8.2, page 227). These temporary errors generate inconsistencies that cause problems for the planning modules. These errors are caused by the uncertainty in the sensor measurements. Section 10.5 presents simulations to assess the effect of this kind of uncertainty in BERODE-2. It is concluded that the uncertainty in the positions of the simbots increases the difficulty of the task of network maintenance. In the presence of uncertainty the network is more likely to become disconnected. This is reflected in the longer time required for building a map where most of the additional time is used by the simbots in moving to positions that restore full connectivity of the network. Nevertheless the full connectivity of the network was maintained most of the time ($\cong$85%). Section 10.6 presents a summary of the conclusions from the simulations.

[1] In our office test environments. This number will reduce as map features increase, and vice versa.

10.2 Metrics for Map Comparison

The exploration task requires consistency in the maps generated by the simbots. In centralized architectures the central agent merges the partial maps generated by the simbots therefore only a global representation of the environment is generated. In the case of decentralized architectures each simbot maintains its own representation. Depending on the approach simbots exchange their representations either periodically or when they come within range of each other. In BERODE-2 the simbots periodically exchange their recently observed features. This type of exchange guarantees that the observed features are used only once to update the feature based maps of the simbots. The maps generated by the simbots are likely to contain small differences. Nevertheless, so long as these differences do not create inconsistencies, the exploration process will not be impaired. Since the approach adopted here will not produce completely consistent maps, for the assessment of system performance it is important to identify and quantify the differences in the maps generated by the simbots. Moreover it is important to assess these differences for simbot teams of different sizes because as the team size increases more inconsistencies may occur.

We propose to compare the final feature maps both directly and through their projected grid and topologic representations. The purpose of having three metrics for the comparison is to identify situations where the inconsistencies in the maps cause poor performance in BERODE. It is expected that topological inconsistencies will have a greater impact on the performance of BERODE than small metric inconsistencies. In the direct comparison the features from pairs of maps are matched. The percentage of matched features is used as the measure to determine the similarity between the pairs of maps. In the projected grid comparison the feature maps are first projected into grid maps and then their similarity is measured by comparing their grid cells. In the topologic comparison a topologic representation is obtained from the projected grid map. The topologic map is a representation where a finite set of places are connected by paths. Places are positions in the environment where qualitative changes occur. Two metrics are then applied to compare these places.

10.2.1 Feature Comparison

The feature comparison is based on the analysis of the feature maps and the matching of maps in pairs. By analysing the feature maps we can determine the relationship between the uncertainty of the features in the maps and the number of simbots. This

allows us to determine if there is a benefit with respect to map accuracy in the use of more simbots. By matching pairs of maps we can determine how similar the maps built by a team of simbots are. Ideally the maps built by the team of robots should be identical. In practice this is not the case, however BERODE-2 can be regarded as scalable with respect to map consistency if the maps have a high degree of similarity (percentage of matched features) and the degree of similarity is the same if not better when the number of simbots is increased.

The analysis of the maps consists of the calculation of the average of the uncertainties for the feature parameters and the average number of observations per feature. Four averaged uncertainties are obtained for the current feature models (two parameters for the line feature and two parameters for the point feature). The purpose of the analysis is to determine the trends in the uncertainty parameters as the size of the simbot team increases. The number of observations per feature is expected to increase as the number of simbots increases. For clarity these values will be presented as ratios with respect to the values found for the minimum size of the simbot network (1 simbot).

For the matched maps the values calculated are the percentage of matched features and the feature surplus ratio. The percentage of matched features measures the similarity between the maps generated by a team of simbots. The feature surplus ratio compares the number of features for simbot teams of different sizes.

The percentage of matched features is the average of the percentages for all the combinations of pairs of maps. Matching more than two maps by sequentially matching pairs of maps is inadequate because the result is different for each sequence (the matching process is not commutative). The result differs because the matching test based on Mahalanobis distance uses a distance threshold (Chi-Square test) where the matches close to the threshold return different results (Section 6.3.8, page 155). As more maps are sequentially matched and averaged, the outcome of the matches diverges more for different matching sequences.

Features are matched using the same criteria and thresholds as in the matching algorithm described in Section 6.3.8 (page 155). This algorithm is used to match the extracted features from the sensors to the previously mapped features. The matching process allows multiple matches for a feature. For instance a large line from a corridor in one map may appear as two slightly overlapped small lines in the other map. Figure 10.1 presents an example of the results of the matching process for a pair of maps generated by two simbots. In c) the matched and averaged map is shown while in d)

the unmatched features from both maps are shown.

The feature surplus ratio compares the number of features for the maps generated by simbot teams of different sizes. As the number of simbots increases the features are observed more frequently. If these additional features fail to be matched against previously mapped features they are incorporated in the map as new features. These failures generate a surplus in the size of the feature map. Larger feature maps are more computationally expensive (O(N^2) where N is the number of features in the maps). It is therefore important to identify if there is a trend as the size of the simbot team increases.

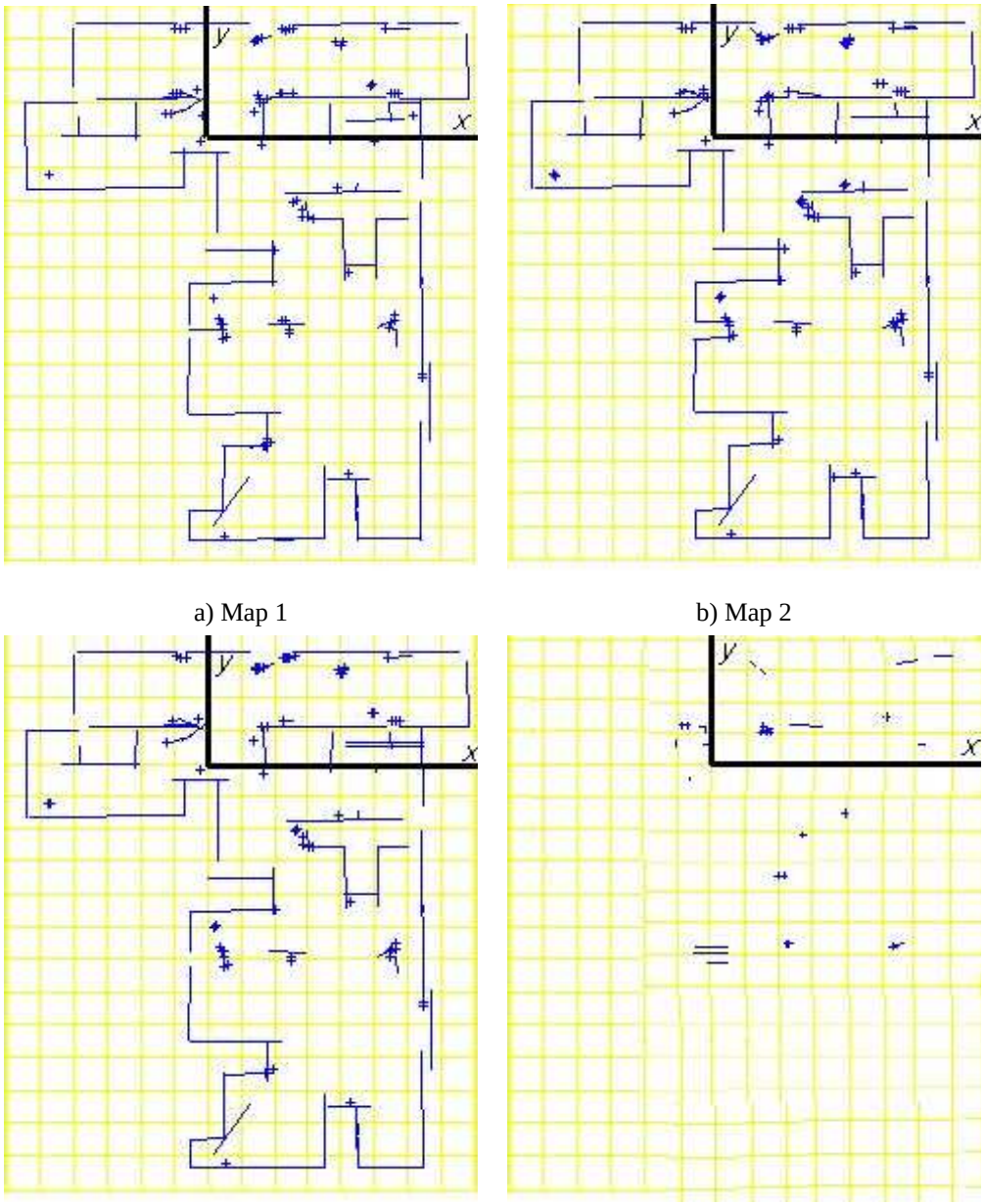
The feature comparison is used to determine the similarity of pairs of feature maps based on the number of matched features. The comparison also analyzes the uncertainty of the feature parameters as a function of the simbot network size.

10.2.2 Topological Comparison

A topological map is a representation of the environment where a finite set of places are connected by paths. In our topologic map places are positions in the environment where qualitative changes occur. Typically these places are positions where two or more paths intersect. Dead ends are also such places. Based on these places it can be easily determined if a place is a room, corridor etc. however this is beyond the scope of this comparison. Topologic maps are usually represented with Voronoi[2] graphs (Choset and Nagatani, 2001). Topologic maps are useful to compare maps at a high abstraction level. In our approach the simbots have to be able to generate consistent maps at the topologic level otherwise the coordination between them fails. For instance when one simbot identifies a corridor and another simbot identifies a dead end the simbots are not able to coordinate and explore the corridor.

The topologic comparison has the following steps: projection, topologic map generation and place measurement. In the projection the feature maps are projected into their probabilistic grid maps (Appendix C). Then the probabilistic grid map is converted to a binary grid map. The binary map is the thresholded probabilistic map where cells are either occupied or free space. The topologic map is extracted from the binary grid map. To extract the topologic map the thinning algorithm proposed by Wang and Zhang (1989) is used. The thinning algorithm returns a skeleton of the grid map. The skeleton grid map is formed by the set of positions equidistant to the

[2] A Voronoi graph is a set of positions equidistant to the N_o closest obstacles.



a) Map 1 b) Map 2

c) Average map d) Unmatched features

Figure 10.1: An example of the matching process for two maps a) map 1 and b) map 2, c) shows the matched and averaged features from the matching process, d) shows the unmatched features from both maps. In this map it is observed that some short line segments could not be matched. These small line segments are difficult to observe in the original maps because they are very close to larger line segments. The size of the reference grid (yellow lines) is 1m x 1m.

n closest obstacles. Based on the skeleton map the places for the topologic map are extracted. A place is a position on the skeleton map where more than or less than two branches fork. Figure 10.2 presents an example of the topologic map generated from the probabilistic grid map. The topologic places are the blue circles and the red lines are the paths that connect these places (skeleton of the map).

In the place measurement step the maps are compared using two metrics: the average distance between places and the Hausdorff distance. The average distance is the average metric distance between places in two maps. This distance is calculated by matching all the places in a map to their closest place in the other map. The Hausdorff distance is the maximum closest distance between two sets of places. More formally, the Hausdorff distance between the sets of places A and B is defined to be

$$H(A,B) = \max(M(A,B), M(B,A)) \tag{10.1}$$

$$M(A,B) = \max_{a \in A}\{\min_{b \in B}\{d(a,b)\}\} \tag{10.2}$$

where $d(a,b)$ is any metric between these places.

The Hausdorff distance is widely used in computer vision for measuring shape similarity (Lam et al., 1992). The Hausdorff distance has been used for matching purposes in robot exploration based in topologic maps (Munoz-Gomez et al., 2004). In our case we are only interested in using this distance as a value to measure the difference between topologic maps. As explained in the previous section matching more than two maps by sequentially matching pairs of maps is inadequate because the result is different for each sequence (the matching process is not commutative). For this reason the metrics for a team of simbots in the topologic comparison are the average values for all the possible combinations of pairs of maps.

The topologic comparison is used to determine the consistency of the maps at a higher abstraction level than the grid and feature comparison. The comparison measures the similarity between significant places in the environment.

10.2.3 Grid Comparison

In the grid comparison the feature maps are first projected into grid maps and then their similarity is measured by using a grid metric. The grid metric is a value that determines the similarity between the feature maps built by the simbots. The value is calculated by projecting the feature map from each simbot into a consistency grid map. The feature map is projected first into a probabilistic grid map (Appendix C).

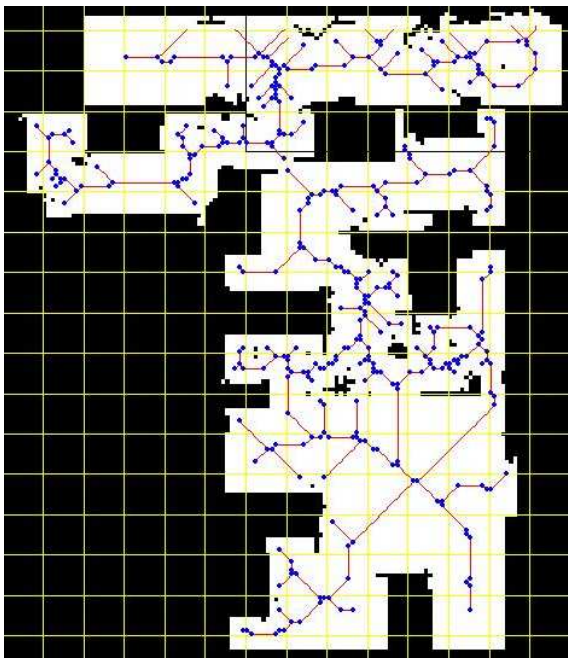


Figure 10.2: Topologic map of the feature map built by a simbot. The feature map is projected into a grid map from which the topologic map is obtained using a thinning algorithm. The blue circles represent the topologic places and the red paths are the paths that connect these places (skeleton of the map). The size of the reference grid (yellow lines) is 1mx1m.

Then the probabilistic grid map is converted to a binary grid map. The binary map is the thresholded probabilistic map where cells are either occupied or free space. A cell that has a value of zero is free and a value of one if is occupied. The binary grids from the n maps (from the n simbots) are added in a conjunctive grid map g. A cell $g(x,y)$ in this map has then a value $[0,n]$ that represents the number of maps in which the cell was labelled as occupied. The value $v(x,y)$ of the grid cells in the consistency grid map are calculated as

$$v(x,y) = \begin{cases} 2*(n-g(x,y))/n & (n/2) \leq g(x,y) \\ 2*g(x,y)/n & (n/2) > g(x,y) \end{cases} \tag{10.3}$$

where $v(x,y)$ has a value $[0,1]$ that represents the degree of consistency for the cell. A value of 0 implies that in all the maps the cell has the same label whereas a value of 1 implies the largest inconsistency when half of the simbots disagree with the other half of the simbot team. The grid metric is the averaged value for all the cells. The grid metric is then a normalized value with respect to the number of maps and can be used to compare the degree of consistency of the maps generated with different numbers of simbots. Figure 10.3 presents an example of the conjunctive and consistency grid maps generated by a group of ten simbots for the environment from Figure 7.1 (page

161) using a grid cell resolution of 0.1m for the projection of the features. In a) the percentage represents the portion of the simbots that labelled a cell as occupied space while in b) the percentage represents the degree of consistency for the cells.

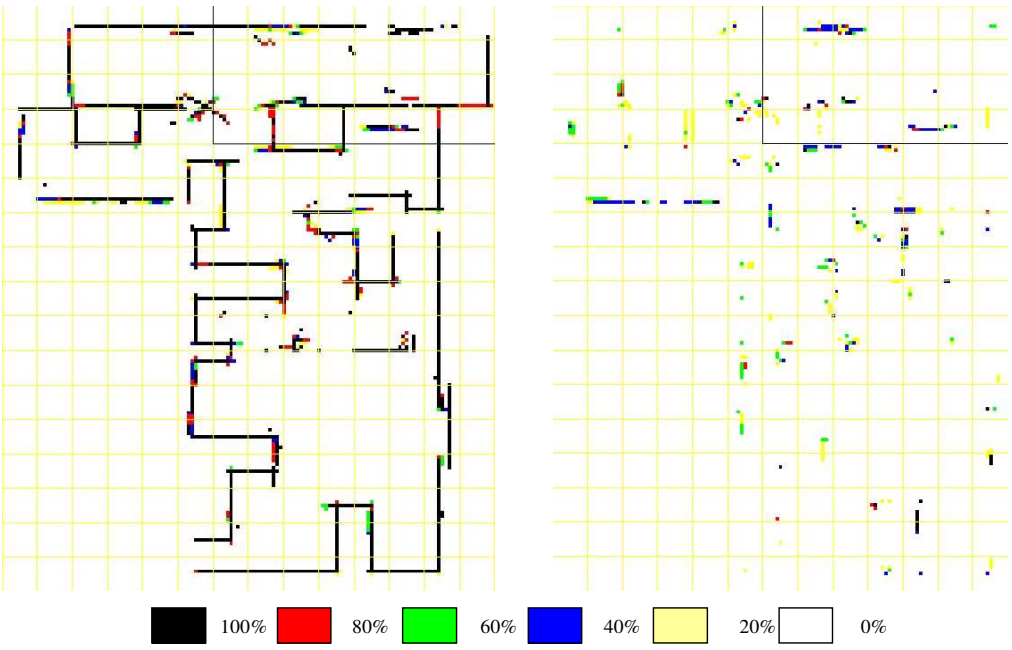
The grid metric compares the consistency based on the projection of feature maps into grid maps. It is then important to verify that the metric is relatively invariant with respect to the size of the grid cells (grid resolution). To verify this the value of the metric was calculated using four values of resolution (0.025m, 0.05m, 0.1m and 0.2m) for the simulations from Section 9.4 (page 277). Figure 10.4 presents the grid metric values and their trend lines for the simulations that used the *LOS* model. For the simulations with the *RF* model similar results were obtained. It can be observed that the trend lines are almost parallel and that as the resolution decreases the trend lines tend to be closer to zero. It is concluded that the grid metric is a good metric because the resolution does not affect the trends with respect to the number of simbots. The tendency to be closer to zero is because of the coarser resolution.

We have proposed and defined three types of comparison: grid, feature and topologic for the maps built by the simbots in BERODE-2. By having three levels of comparison the maps can be compared with different degrees of abstraction. It is then easier to identify the type of inconsistencies that cause poor performance in BERODE-2.

The grid comparison measures the similarity at a coarse level (grid cells). In this comparison the similarity between the extremes of the feature lines can be quantified. The accurate detection of the extremes has proven to be important in the performance of BERODE-2. The feature comparison measures the uncertainty in the map as a function of the simbot network size. The topological comparison measures the similarity between significant places in the environment. We hoped that the maps would be as similar if not more similar when the degree of abstraction for the comparisons was increased. This was the case in our simulations.

10.3 Map Consistency for Different Simbot Network Sizes

In BERODE-2 each simbot builds its own feature based representation of the environment. In this section we present a simulation to analyse the consistency of the maps built by simbot networks of different sizes. The analysis is carried out using the pro-



a) Conjunctive map b) Consistency grid map

Figure 10.3: An example of the a) conjunctive and b) consistency grid maps build by a group of ten simbots for the environment from Figure 7.1. In the conjunctive map the percentage represents the number of simbots that detected the grid cells as occupied. In the consistency grid map the percentage represents the number of robots that agreed that a grid cell was either occupied or free. The size of the reference grid (yellow lines) is 1mx1m.

posed types of comparison: feature based, metric and topologic. It is expected that the maps built by bigger simbot networks will contain more repeated features because more observations are made by these networks. The simbots incorporate more information from different sources of uncertainty. For the metric comparison it is expected that the maps built by bigger simbot networks will be more consistent because as more observations are made the projected grid maps tend to be more accurate. For the topologic comparison it is expected that the consistency of the maps will be the same for the different simbot network sizes because based on the visual inspection of the maps built by different simbot they seem to contain only small metric differences (Figure 10.1).

The settings for the simulation were the same as those of the simulation from Section 8.6 (page 239). The *RF* and *LOS* models were used in the medium and large simulated environments (Figure 7.1, page 161). The size of the simbot team was n=4, 6, . . . , 16 in the medium environment. The size of the local networks was k=min ($n^{0.5}$,

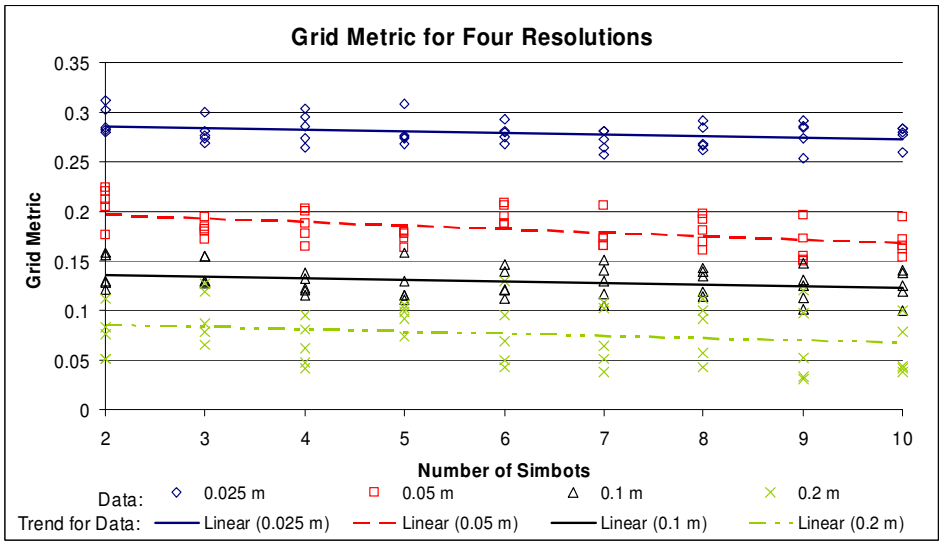


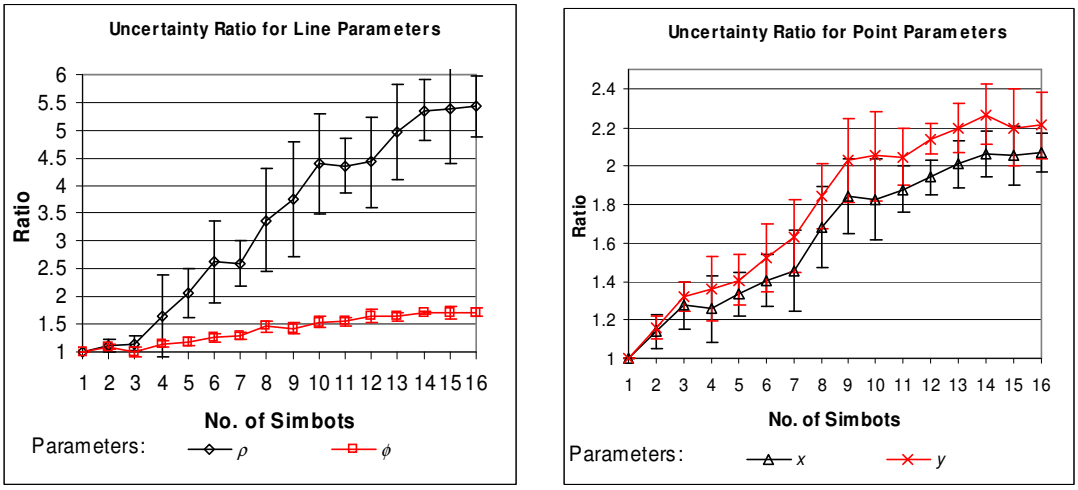
Figure 10.4: Grid metric for four grid resolutions. The grid metric is a value that quantifies the similarity between the grid maps of the simbots. The data from ten trials for each combination of resolution and simbot network size (n=1, 2, . . . , 10) is shown. The linear trends are the least square linear regression for the data from the different resolutions.

2n/5), which is the biggest size for the local network in the best trading range (Section 9.4, page 277). Preliminary simulations revealed that for a simbot network of a certain size the consistency is the same for different local network sizes. This was expected because the local features are used only as a temporary aid for navigation and they are not incorporated into the representation. The number of trials was 10 for each simbot network size. The results from the figures of this section are the average results for the *RF* and *LOS* models because there was no difference in the results with respect to the communication model.

Figure 10.5 presents the uncertainty for the line $L(\varphi,\phi)$ and point $P(x,y)$ features for simbot networks in the medium environment. The uncertainty of the parameters is shown as a ratio of the average uncertainty for the feature maps for a network of n simbots with respect to the minimum simbot network size (1 simbot). It is observed that for all the parameters as more simbots are added to the simbot network the uncertainty is smaller. This is expected because the uncertainty in the features decreases monotonically as more observations are made. Larger simbot networks incorporate more observations than smaller simbot networks.

From Figure 10.7(a) it is observed that the observations per feature increases with the size of the simbot network. The graph shows the trends for the total of the features as well as the trends for the line and point feature. Newman (1999) proved that in the

limit for an infinite number of observations the uncertainty of the features is reduced to the uncertainty in the initial position of the robot. In Fenwick et al.'s (2002) research it was shown that the reduction in uncertainty is accelerated when more robots are added to the network. This is due to the combined information from multiple robots. Our results confirm this finding.



a) Line features b) Point features

Figure 10.5: Uncertainty ratio for the parameters of the a) line $L(\varphi,\phi)$ and b) point $P(x,y)$ features of the maps built by the simbot network of different sizes. The graphs show the average results from the *RF* and *LOS* models for the medium environment. The vertical lines show the standard deviation of the averaged parameters.

From Figure 10.5(a) it is observed that the ratio is bigger for the line features compared to the point features. This is to be expected because a line represents a larger portion of the environment and can be extracted from more positions than a point.

From Figure 10.5 it is observed that the decrease in the uncertainty is much bigger for the distance to the origin parameter ρ for the line feature than for other feature parameters. This result was unexpected but is caused by the larger observation ratio for the line features compared to the point features. The ratio for the orientation parameter ϕ is smaller than the ratio for the distance to the origin parameter ρ because when a line is extracted from raw data the uncertainty is proportionally larger for the orientation parameter. The convergence for the orientation ϕ is slower.

Figure 10.6 presents the percentage of matched features for different simbot network sizes. It is observed that the percentages are similar with an average value of 93.65% and a standard deviation 0.56%. The map built by each simbot is updated

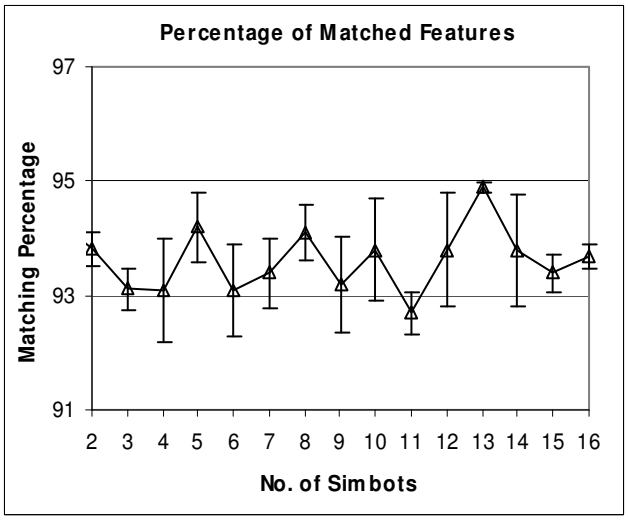
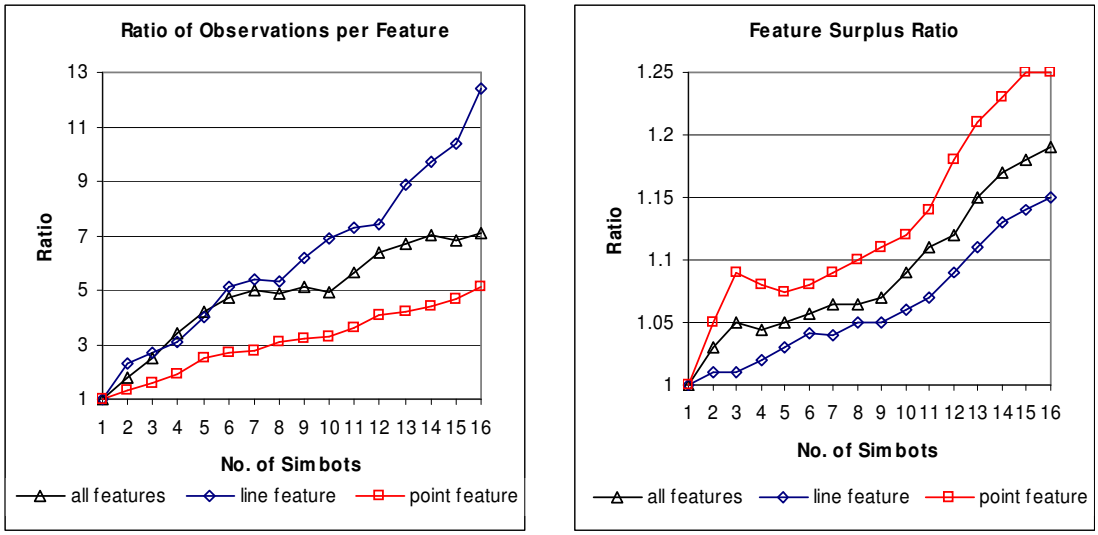


Figure 10.6: Percentage of matched features for different simbot network sizes using the *RF* model in the medium environment. The vertical lines show the standard deviation in the trials. It is observed that the percentage is stable within a range of 2%.



a) Observations per feature b) Feature surplus ratio

Figure 10.7: Ratios of a) observations per feature and b) feature surplus for different simbot network sizes using the *RF* and *LOS* models in the medium environment. The trends show the ratio for all the features (observations of line and point features), and for line and point features alone.

using the observations from all the simbots. Numeric differences in the feature maps from the simbots can cause matching failures when their features are matched.

From Figure 10.7(b) it is observed that the feature surplus ratio is larger for large simbot networks compared to small simbot networks. This is caused because in larger simbot networks the map of a simbot contains proportionally less local observations of the features. The integration of features from the simbot network is delayed because the simbots transmit their extracted features periodically. These delays in the integration of the features cause failures in the matching of the external features that have a correspondence in the current map of the simbot. These external features are added to the map as new features. This causes the feature surplus because of the repeated features. A solution to this problem is proposed in the next section.

Figure 10.8 presents the grid metric for different simbot network sizes in the medium environment. It is observed that as the number of simbots increases the value for the grid metric is smaller. This was expected because as more observations are made the projected grid maps tend to be more accurate.

In the topologic comparison the topologic places of pairs of maps are compared using the average and the Hausdorff distance (Section 10.3.2). The topologic map is the skeleton of the projected grid map. The skeleton grid map is the set of positions equidistant to the N_o closest obstacles. The topologic places are positions on the skeleton map where more than or less than two branches fork. The average distance is the average metric distance between places in two maps. This distance is calculated by matching all the places in a map to their closest place in the other map. The Hausdorff distance is the maximum closest distance between two sets of places. Figure 10.9 presents the average and Hausdorff distances for different network sizes using the *RF* model in the medium environment. The average and Hausdorff distances decrease as the size of the simbot network increases. This was unexpected because although the maps may contain small metric differences these differences should not generate differences at the topologic level. It was found that frequently one or two maps built by a team of simbots are different at the topologic level to the rest of the maps. The distances are calculated by averaging the results for all the combinations of pairs of maps. The distances between these maps and the rest of the maps are much larger than the average values for the distances. In small simbot networks these larger distances contribute more to the average value. This is observable in Figure 10.9 where the variance for the average values is larger for small networks compared to that of large networks.

As previously explained, to achieve decentralized coordination between the sim-

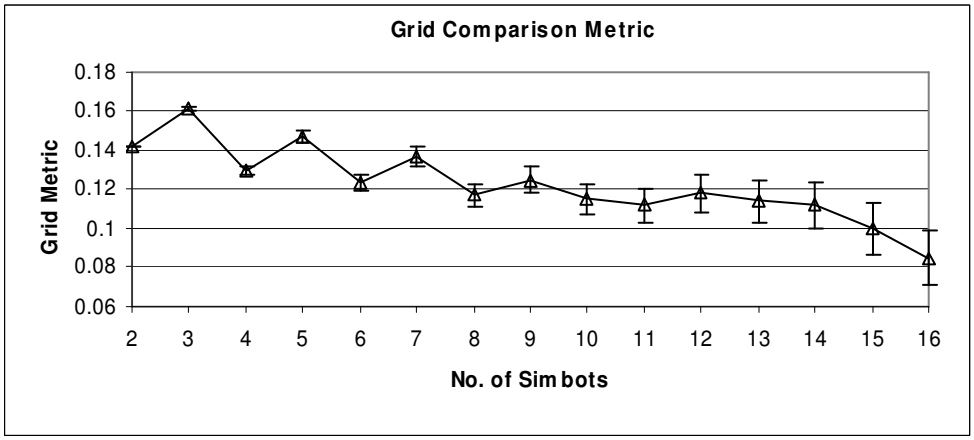


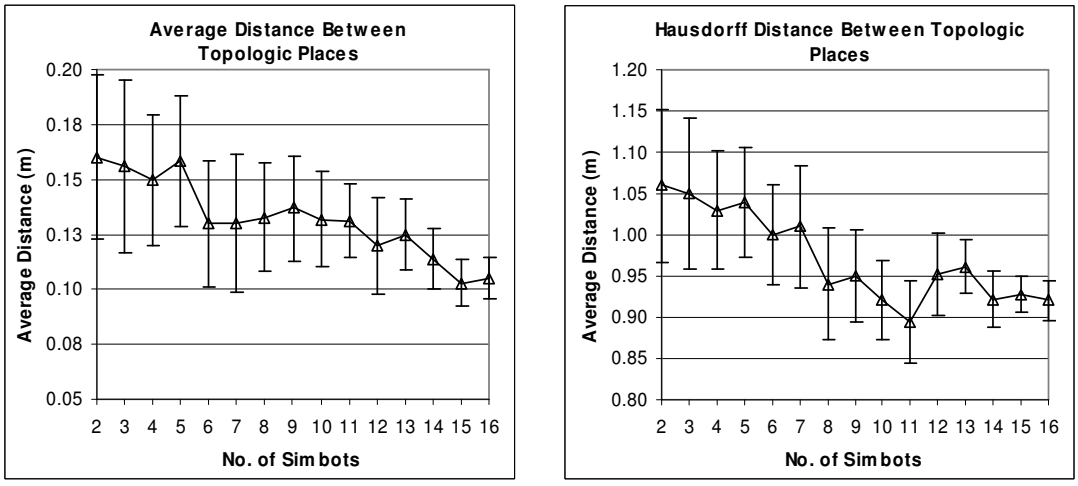
Figure 10.8: Grid comparison metric for different simbot network sizes using the *RF* and *LOS* models in the medium environment. Smaller values mean that the projected grid maps are more consistent. The vertical lines show the standard deviation in the trials.

bots their maps have to be consistent enough to take congruent decisions. Therefore, if the consistency of the maps is at least the same when more simbots are used in BERODE-2 we can expect that the consistency of the maps will not be a limit for the scalability of BERODE-2.

We conclude that BERODE-2 is scalable with respect to map consistency because the maps are more consistent when the size of the simbot network is larger. The maps are more consistent at the metric and topologic levels of the map. From the feature comparison it was observed that: the uncertainty in the features decreases as the number of simbots increases; the percentage of matched features is similar for all the simbot network sizes; and the number of repeated features (feature surplus ratio) increases as the number of simbots increases. Based on the first two observations for the feature comparison we argue that the degree of consistency at the feature level is at least the same if not better as the number of simbots increases. As explained before the surplus is generated by failures in the matching process. These failures are caused by the delay in the integration of the information. This problem is specific to the current implementation of the Map Interface Module and is discussed in detail in the following subsection.

10.3.1 Analysis of the Current Implementation

In BERODE the simbots periodically transmit their extracted features. This causes a delay in the integration of these external features into the local map of a simbot. In the



a) Average distance b) Hausdorff distance

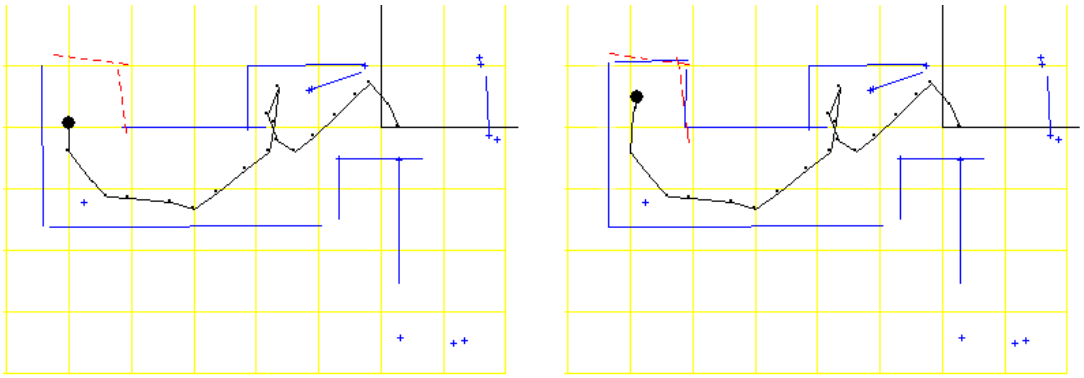
Figure 10.9: a) Average and b) Hausdorff distances between the topologic places from the topologic comparison of the maps built by different simbot network sizes using the *RF* and *LOS* models in the medium environment. Smaller values for the distances mean that the topologic maps are more consistent.

current implementation the external features do not incorporate the later corrections realized by the *EKF*. We expected that the omission of these corrections would not cause failures in the matching of the features. Only small differences in the parameters of the features and their uncertainty were expected.

Figure 10.10 presents a typical example of the map building process for a simbot. At time t_1 the simbot stores two extracted lines (dotted red lines) for transmission. b) At time t_2 the features are re-observed and updated by the *EKF* in the map. In the storage module the re-observed features are matched and averaged but they do not incorporate the later correction (dotted red lines) from the *EKF*. The difference between the stored lines and the corresponding lines in the local map is observed in Figure 10.10(b).

In the previous section we found a large amount of repeated features in the maps generated by the simbots. Although the Map Interface Module (Section 8.2, page 227) implements a process that periodically identifies and removes repeated features, the process fails to remove all the repeated features because there is not enough similarity. The features are not similar enough because as more observations are integrated the uncertainty of the features decreases monotonically. In the current matching process based on the Chi-Squared criterion and the Mahalanobis distance (Section 6.3.8, page 155) when the features are more certain the difference between the parameters of the

features needs to be smaller so that they can be matched. In the Map Interface Module the line features are ordered with respect to their distance from the initial position. This is used to speed up the periodic check of repeated features; however this process is still computationally expensive because the *EKF* has to be updated. Moreover, the more frequent execution of this process does not much improve the removal of the repeated features and incurs a high computational cost.



a) t_1 b) t_2

Figure 10.10: An example of the map building process. a) At time t_1 the simbot stores two extracted lines (dotted red lines) for transmission. b) At time t_2 the features are re-observed and updated in the map. It is observed that the stored features do not incorporate the later correction realized by the *EKF*.

We observed in the previous section that the number of repeated features was larger for larger simbot networks. This generated a surplus in the number of features with respect to the necessary number of features to represent the environment. We hypothesised that the surplus generated by the delay in the information is specific to the current implementation of the Map Interface Module and not associated with BERODE's general architecture.

To validate this hypothesis a simulation where the simbots transmit their extracted features without delay (t=0) was conducted using the same settings from the previous section. Figure 10.11 presents the graphs for the percentage of matched features and the feature surplus ratio for the Map Interface Module with no delay (t=0) and with a global update frequency (t=20). It is observed that when there is no delay (t=0) the percentage of matched features is larger than when there is a delay (t=20). When there is no delay the percentage of matched features is stable (97.3$\pm$0.5%) with respect to the number of simbots. As observed in Figure 10.11(b) the feature surplus ratio

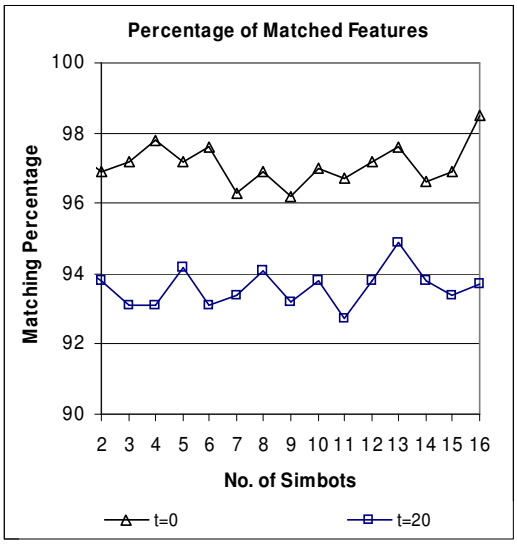
increases as the number of simbots increases. This increase is much smaller when there is no delay (t=0) than when there is a delay (t=20). When there is no delay there is only a slight increase in the surplus ratio with respect to the number of simbots. This confirms that the delay in the integration of the information generates the failure in the matching process and causes the surplus in features. The use of local maps is suggested as a solution to this problem. The next subsection discusses the implications of this use of local maps in BERODE.

10.3.2 The Use of Local Maps

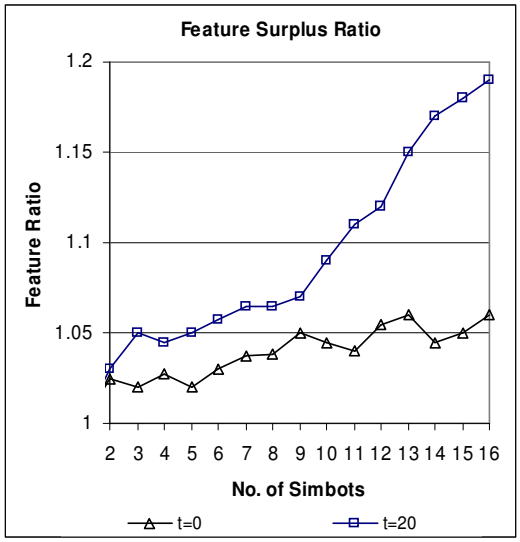
We argue that the implementation of an *EKF* based on local maps for BERODE-2 will solve the inconsistency problem when the updates are less frequent. Previous research (e.g., Tardos et al., 2002a; Rodriguez-Losada, 2004) has implemented *EKFs* based on local feature maps to reduce the computational cost of the *EKF* (Section 3.4.3, page 42). In these approaches the simbots build a local map for a subset of features, and accumulate the pending correction, so that it can be transferred to the complete map when required. The computational cost of the *EKF* is then a function of the number of features in the local map. These techniques fuse independent local maps using common references. Repeated features are identified and used to improve the map estimates. Afterwards these features are removed from the map. The decision to fuse a local map to the complete map is frequently based on a maximum number of objects allowed for the local maps.

Losada conducted experiments with multiple real robots using *EKFs* based on local maps (Rodriguez-Losada, 2004). Losada discovered that when *a priori* knowledge was not used and a large correction was done by the *EKF* the maps tended to deform locally. The robots used *a priori* structural knowledge (e.g. parallelism, perpendicularity) in the local maps. The experiments showed that the robots generated maps as consistent as the maps generated with a global *EKF*.

In previous research the robots created local maps (Tardos et al., 2002a; Rodriguez-Losada, 2004) and delayed their integration to the global map based on a size criterion. We argue that the use of an *EKF* based on local maps will eliminate the feature surplus problem of our current implementation because it has been shown that delays in the integration of local maps to the global map do not cause feature surplus problems. The simbots could start a new local map after the last transmission of features at the global level. Instead of transmitting only the parameters of the features and their covariance



a) Percentage of matched features



b) Feature surplus ratio

Figure 10.11: a) Percent of matched features and b) feature surplus ratio for different global update times t=0, 20 steps using the *RF* and *LOS* models in the medium environment.

the simbots should transmit the features and the covariance matrix of the local map. The covariance matrix contains the information about the correlation between the features in the local map.

A simbot that has observed and stored k features will transmit 4*k covariance values in the current implementation. If a local map approach is used the covariance matrix will contain $(4*k)^2$ covariance values. The increase in terms of communication is then a constant square factor. We estimate that this will increase the required bandwidth about 10% with respect to the current bandwidth required in BERODE-2. We don't expect this to be a problem based on the bandwidths available in current technologies.

As explained in the previous section the more frequent checking for repeated features increases the computational cost and does not much diminish the number of repeated features.

In this research a global map was used because of its ease of implementation. This implementation however has been found to have this drawback of repeated features. Repeated features cause an increase of $(z/N)^2$ in the computational cost of the *EKF*, where z is the number of repeated features and N is the number of features. The computational cost of the global *EKF* is $O(N^2)$ where N is the number of features in the map. In the simulations the features have been transmitted at the global level every 50

seconds (20 sensing steps * 2.5 per sensing step). This transmission rate is very low and should not be a problem in terms of bandwidth for current communication technologies. In the previous section it was observed that the surplus of features was below 15% for network sizes with $n < 14$.We argue that due to its ease of implementation a global *EKF* is adequate for medium simbot team sizes ($n < 14$) in environments similar to our office test environments. This will vary depending on the number of features in the mapped environment. In cluttered environments the limit on the team size will be smaller than ($n < 14$) because more features are extracted.

The use of local maps for map building is an ongoing research area and has proven to be a time consuming task. For this reason the implementation of local maps is beyond the scope of this thesis and remains as a future area of research for BERODE-2. The next section analyzes the repetition problem for less frequent updates to determine the usefulness of the current implementation.

10.4 Map Consistency for Different Global Update Frequencies

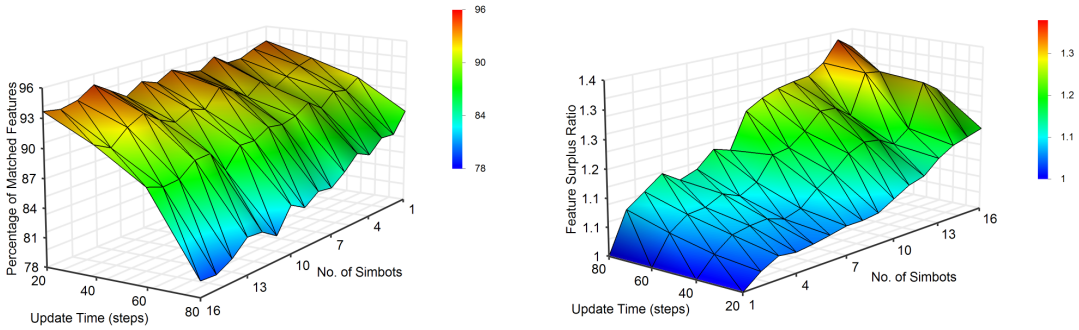
Previous simulations from Section 9.4 (page 277) revealed that when the global features are shared less frequently the maps built by the simbots have a tendency to contain repeated features. This is caused by the failure in the matching of the external features with the current feature map of the simbots. Moreover, the simulations from the previous section revealed that the maps built by large simbot networks contain more features compared to those built by small simbot networks. These additional features are often repeated features.

This section presents a detailed analysis of the consistency of the maps when the features are shared less frequently at the global level to determine the usefulness of the current implementation. Based on previous simulations it is expected that the number of repeated features will increase as the number of simbots increases and the feature updates are less frequent. It is expected that the degree of consistency for the maps will slowly degrade as the updates become less frequent. We expect to be able to determine a range of network sizes and update frequencies for which the current implementation is suitable given its ease of implementation.

The settings for the simulation are the same as those of the simulations from Section 8.6 (page 239). The simulations used the medium environment for the *RF* and

LOS model. The size of the simbot team was $n = 2,3,..,16$. The size of the local networks was k=min ($n^{0.5}$, $2n/5$), which is the biggest size for the local network in the best trading range (Section 9.4, page 277). The frequency of the feature updates at local and global level were S_L=5 and S_G=10, 20, 30, 40 sensing steps respectively. A sensing step is the time that the simbot requires to sense an area using the *low cost* platform. This process takes 1.8 seconds for the proposed *low cost* platform (Section 7.7, page 207).

Figure 10.12 shows the percentage of matched features and the feature surplus ratio. As expected the surplus ratio increases as the number of simbot increases and the feature updates are less frequent. It is observed that as the updates become less frequent the increase in the surplus ratio is much bigger for large simbot networks compared to that of small simbot networks.



a) Percentage of matched features b) Feature surplus ratio

Figure 10.12: a) Percentage of matched features and b) feature surplus ratio for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

From Figure 10.12 (a) it is observed that the percentage of matched features decreases as the feature updates become less frequent. This was expected because the corrections realized by the *EKF* are not applied to the features that the simbots shared periodically causing failures in the matching process (Section 10.3.1). The amount of failures increases when more simbots are added to the network because the simbots incorporate proportionally more external features. This is observed in Figure 10.12(a) where the decrease is larger for large simbot networks compared to that of smaller simbot networks; as a consequence the surplus ratio for large network increases faster than for small networks when the updates are less frequent. It is observed that when the network size is smaller than 14 simbots and the global update time $S_G \leq 20$ the surplus ratio is at most 1.15. We argue that the current implementation is a good im-

plementation in these cases ($n \leq 14$ and $S_G \leq 20$) for our test environments because of its ease of implementation and its low additional computational cost. As discussed in the previous section, this number will reduce as map features increase, and vice versa.

Figure 10.13 shows the comparison based on the grid maps. It is observed that the consistency of the maps decreases faster for large simbot networks than for small networks. Similar tendencies are observed for the metrics used in the topologic comparison (Figure 10.14). It is observed that for the higher update frequency (S_G=10) the maps are more consistent (smaller values for the grid metric) for large simbot networks compared to smaller networks, whereas for the smallest update frequency (S_G=40) the consistency is only slightly larger. It is observed that the slope of increase in the grid metric for small simbot networks is very small compared to the slope of large simbot networks. The decrease in the consistency at the metric and topologic levels is attributed to the integration of the uncorrected features. The faster decrease in the consistency for large networks is caused by the integration of a larger proportion of uncorrected features.

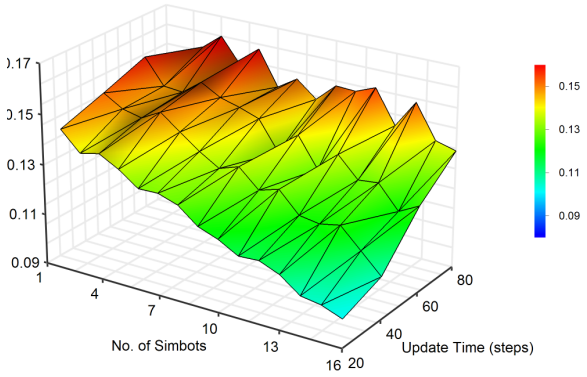
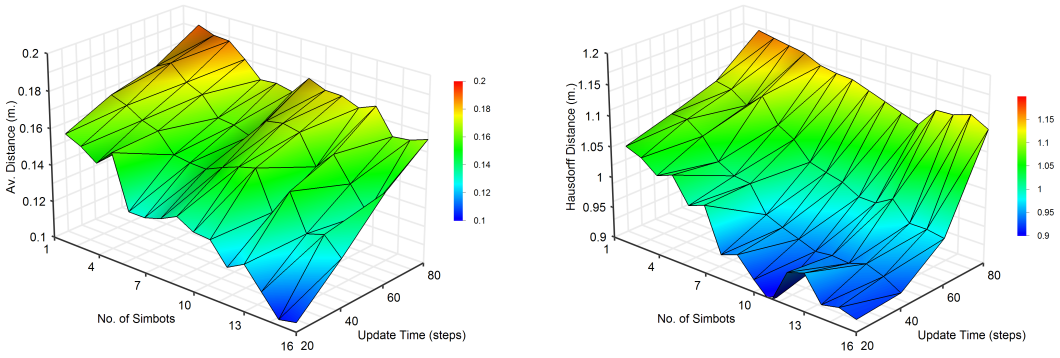


Figure 10.13: Grid metric for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

The implementation of a Map Interface Module based on local maps has been discussed in the previous section. It is believed that this implementation will greatly reduce if not eliminate the problem of repeated features.

We conclude that the current implementation is suitable for simbot networks $n \leq 14$ and global update times $S_G \leq 20$ (sensing steps) given its ease of implementation and its low additional computational cost. The limit on the size of the simbot network will vary depending on the amount of map features. In cluttered environments the limit will be smaller because the cost of the global *EKF* is $O(N^2)$ where N is the number of features in the map.



a) Average distance b) Hausdorff distance

Figure 10.14: a) Average and b) Hausdorff distances for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

The current implementation is not robust to infrequent updates because the number of repeated features increases when the updates are less frequent. This increment is bigger for large simbot networks. In Section 10.3.2 we discussed the implementation of a Map Interface Module based on local feature maps. We argue that this implementation will eliminate the inconsistency problem arising when the updates are less frequent.

10.5 Comparison with Ideal Sensors

This section analyzes the effect of the uncertainty in the sensors on the performance of BERODE-2. The simbots are based on the *low cost* platform described in Section 7.7 (page 207). The platform uses inexpensive sensors to build the environment: sonar and infrared. The simbots build a feature based representation of the environment. The estimated features frequently contain temporary errors that are removed by map management mechanisms (Section 8.2, page 227). For instance the estimation of the limits of the line features is difficult to achieve with sonar sensors. Frequently the line has an estimated length longer than the real length. These longer lengths generate problems for the planning modules.

For instance Figure 10.16 presents a typical example of these problems for a simbot network of three simbots. The Figure shows the map built by simbot R_0 from the environment from Figure 10.15 at times t_1 and t_2. Between times t_1 and t_2 the map management process is executed (Section 8.2, page 227). This process finds and removes repeated lines and updates the extremes of perpendicular lines with extremes in

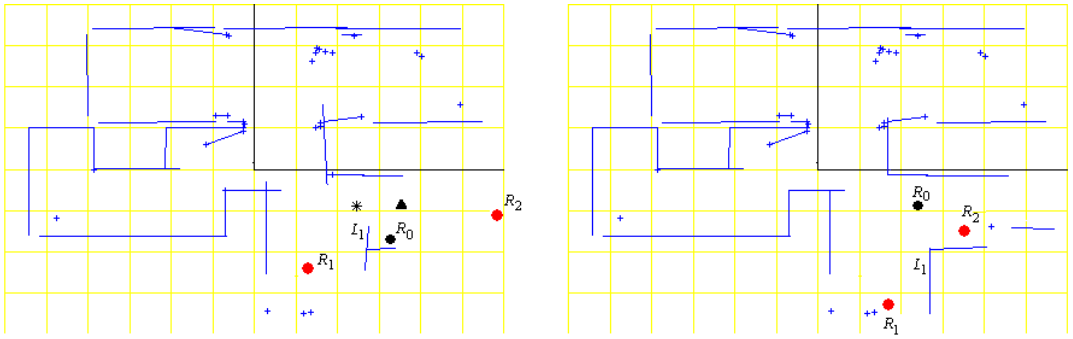
close positions. The connections in the MST control network are $R_0 - R_1$ and $R_0 - R_2$. R_1 and R_2 are Explorer simbots, R_0 is a Maintainer simbot. R_0 generates a reactive plan to move towards the position * (Figure 10.15 (a)). The * position is the best position for the current map that contains large inaccuracies in the estimated extremes of the line L_1. Δ is the position that would have been calculated if the extremes of the line L_1 were estimated accurately. The Δ position is closer to the current position of R_0. Moving to the Δ position would allow R_1 and R_2 to keep exploring the environment. The movement of R_0 to the * caused a transition to the Pusher role for R_2 because the area was unsafe to explore due to the movement away from that area from R_0. The time required to explore the environment is then increased.

Figure 10.15: Section of the simulated medium environment.

To assess the effect of these inconsistencies caused by the uncertainty in the measurements we propose to compare the performance of BERODE-2 when there is no uncertainty in the sensors and simbot positions. We will refer to this version of BERODE-2 as the certain version whereas the version that integrates the uncertainty will be referred as the uncertain version. In the certain version the simbots are assumed to have a perfect laser sensor that senses the circumference bearing with a 5^o resolution. The range is the same as that of the sonar sensors. The simbots build a probabilistic grid map of the environment based on these measurements. The Map Interface Module stores the measurements instead of the features. The sensing process for a position is assumed to require the same time as the time required by the *low cost* platform (1.8sec.).

From the previous sections it was observed that the consistency of the map degraded as the global feature updates became less frequent. In this simulation we propose to compare the certain and uncertain versions for different update frequencies. Two metrics are used for the comparison: the percentage of additional exploration

time for the uncertain version with respect to the certain version; and the percentage of time that the network remains as a fully connected network. It is expected that the percentage of additional exploration time will be larger when the updates are less frequent. It is also expected that the certain version will maintain the network fully connected more time in percentage than the uncertain version.



a) t_1 before map management b) t_2 after map management

Figure 10.16: An example of suboptimal planning for the Maintainer simbot R_0. a) The simbot generates the optimal plan for the current feature map to the $*$ position. b) The simbot moves towards the planned position. The map management process updates the perpendicular lines that intersect.

The settings for the simulation are the same as those of the simulations from Section 8.6 (page 239). The simulation used the medium environment for the *RF* and *LOS* model. The size of the simbot team was $n = 2, 3, .., 16$. The size of the local networks was k=min ($n^{0.5}$, $2n/5$), which is the largest size for the local network in the best trading range (Section 9.4, page 277). The frequency of the feature updates at local and global level were S_L=10 and S_G=10, 20, 30, 40 sensing steps respectively.

Figure 10.17 presents the percentage of additional exploration time for the uncertain version with respect to the certain version of BERODE-2. It is observed that as expected the additional time increases as the global updates are less frequent. For global times $S_G \leq 20$ the additional time is similar for all the sizes of the simbot networks. For global times $S_G > 20$ the additional time increases as the size of the simbot network increases. This is attributed to the larger degree of inconsistency in the maps generated by large simbot networks compared to small simbot networks. The inconsistency problem was analyzed in the previous section.

Figure 10.18 presents the percentage of time fully connected for the uncertain and certain versions of BERODE-2 for different global update times. From this figure it is

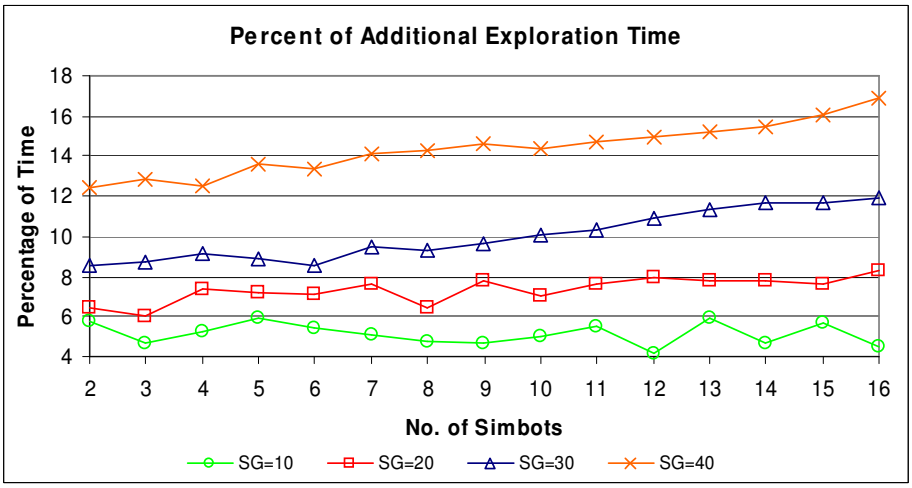


Figure 10.17: Percentage of additional time required to build a map for the uncertain version with respect to the certain version of BERODE-2 for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

observed that:

- As expected the certain version maintains the network fully connected more time in percentage than the uncertain version for all the global update times.

- For all the trends there is a minimum percentage that the network remains as a fully connected network regardless of the size of the network. This validates previous findings from Section 9.4.1 (page 286) about the existence of this minimum for less frequent global updates.

- The minimum percentage is bigger for the certain version than for the uncertain version. This is expected because the uncertainties introduce erroneous information in the reactive plans used to maintain the network connected.

- For the certain version the percentage of time fully connected is the same for the different global update times (Figure 10.19(b)). This is expected because the simbots use local features as an aid to generate the plans to maintain the network connected. Local features are transmitted more frequently than global features. The factors that affect the percentage of time as a fully connected network are the frequencies of the beacon and local feature updates (Section 9.4.1, page 286). A beacon update is the position update that the simbots send to their direct connections.

- For the uncertain version the percentage of time fully connected tends to decrease when the global updates are infrequent ($S_G > 20$). This can be observed more clearly in Figure 10.19(b). Previous simulations (Section 9.4.1) showed that the factors that affect the percentage are the frequencies of the beacon and local feature updates. These results are supported by our findings for the certain version. The decrease for the uncertain version is attributed to the larger degree of inconsistency in the maps for infrequent global updates.

- The percentage stabilizes at the minimum value for a smaller number of simbots in the certain version (n=11) than on the uncertain version (n=13) of BERODE-2. This is expected because the complexity of maintaining larger simbot networks is reduced when there is no uncertainty in the positions of the simbot.

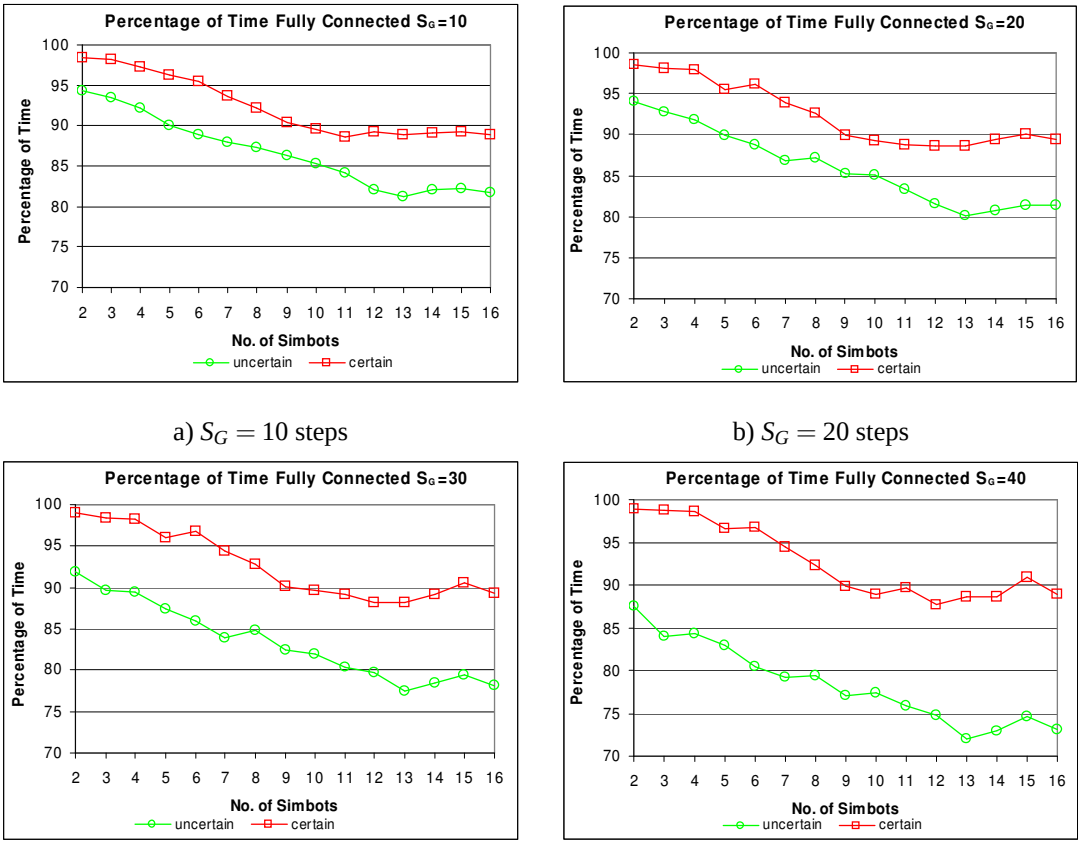
The simulation from Section 9.4.1 showed that there is a minimum percentage of time that a simbot network of any size remains fully connected. In this simulation the frequency of the global update was a fixed value (S_G=20) while the frequencies for the local and beacon updates were varied. In this section the existence of the minimum percentage was validated for different global updates. The minimum percentage exists despite the issue with the current implemented Map Interface Module which only affects the value of this minimum but does not compromise its existence. Moreover, the simulation with the certain version confirms our theory that the minimum percentage is independent of the frequency of the global update frequency.

We conclude that the presence of uncertainty in the positions of the simbots increases the difficulty of the task of network maintenance. In the presence of uncertainty the network is more likely to become disconnected. This is reflected in a longer time required for building a map. Most of the additional time is used by the simbots in moving to positions that restore full connectivity for the network. Nevertheless the full connectivity of the network was maintained most of the time ($\cong$85%).

10.6 Conclusions

This chapter has analyzed the consistency of the maps build by the simbots in BERODE-2 using the *LOS* and *RF* communication models. The simulations show that BERODE-2 is scalable with respect to the consistency of the maps. As more simbots are added to the network, the maps increase in consistency according to the three levels of comparison: feature, metric and topologic level of the map. Newman

(1999) proved that in the limit for an infinite number of observations the uncertainty of the features is reduced to the uncertainty in the initial position of the robot. In Fenwick's (Fenwick et al., 2002) research it was shown that the reduction in uncertainty is accelerated when more robots are used due to the combined information from multiple robots. The simulation from Section 10.3 confirms this finding. The maps built by larger simbot networks are less uncertain than the maps built by smaller simbot networks because they integrate more observations.



a) $S_G = 10$ steps

b) $S_G = 20$ steps

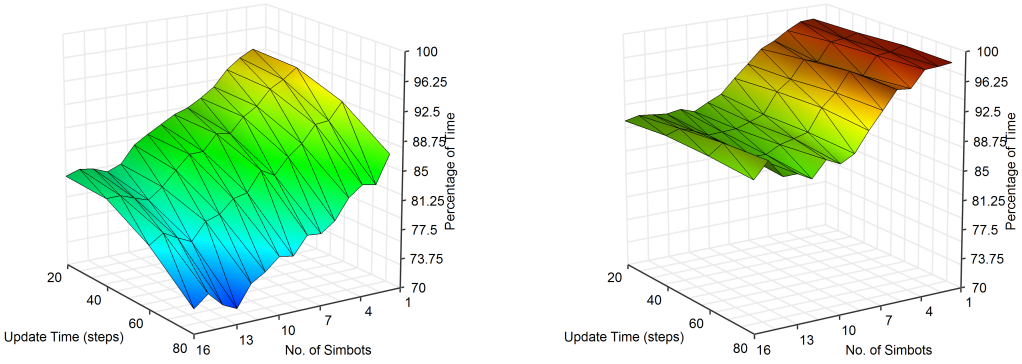
c) $S_G = 30$ steps

d) $S_G = 40$ steps

Figure 10.18: Percentage of time fully connected for the uncertain and certain versions of BERODE-2 for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

In the feature comparison it was observed that the number of repeated features increases as the size of the simbot network increases. This problem was analyzed in Section 10.3.1 where we found that the repetition problem is specific to the current implementation. In this implementation the corrections of the *EKF* are not incorporated into the features that are shared with the network. We had expected that the omission

of these corrections would not cause failures in the matching process; however this has not proved to be the case. In Section 10.3.2 we discussed the use of local maps to solve this problem of repeated features. Previous research (Rodriguez-Losada, 2004) has implemented an *EKF* based on local maps to map environments with multiple robots. Their simulations showed that the robots generated maps as consistent as the maps generated with a global *EKF*. The computational and communication implications of this solution for BERODE are analyzed in this section. The use of an *EKF* based on local maps enables the mapping of larger environments because the computational cost is a function of the number of features in the local map. The use of local based maps for the *EKF* is an ongoing research area and has proven to be a time consuming task. For this reason the implementation of an *EKF* based on local maps is beyond the scope of this thesis and remains as a future area of research for BERODE-2.



a) uncertain b) certain

Figure 10.19: Percent of time fully connected for a) the uncertain and b) certain versions of BERODE-2 for different global update times S_G=10, 20, 30, 40 steps using the *RF* and *LOS* models in the medium environment.

The problem of repeated features was analyzed in Section 10.4 to determine the usefulness of the *EKF* global implementation when the global features updates are less frequent. It was concluded that the global *EKF* implementation is suitable for medium simbot network sizes ($n \leq 14$) in our office test environments due to its ease of implementation and its low additional computational cost. The limit on the size of the simbot network will vary depending on the amount of map features. In cluttered environments the limit will be smaller because the cost of the global *EKF* is $O(N)$ where N is the number of features in the map.

The current implementation is not robust to infrequent updates because the number

of repeated features increases when the updates are less frequent. For large numbers of robots ($n > 14$) or infrequent communications it would be wise to avoid the problem by using the local *EKF* method.

The simulation from Section 10.5 assessed the effect of sensor uncertainty on BERODE-2. It was concluded that as expected having uncertainty in the positions of the features and the simbots increases the difficulty of the task of network maintenance. In the presence of uncertainty the network is more likely to become disconnected. This is reflected in a longer time required for building a map. Most of the additional time is used by the simbots in moving to positions that restore full connectivity for the network.

The simulation from Section 9.4.1 (page 286) discovered the existence of a minimum percentage of time as a fully connected network for any simbot size. The existence of the minimum percentage was validated for less frequent global updates. The minimum percentage exists despite the robustness problem in the current implementation. This problem only affects the value of this minimum but it does not compromise its existence. Moreover, additional simulations showed that the minimum percentage is independent of the frequency of the global updates. This is because the main cause of disconnections in the simbot network is outdated information about the positions of the direct connections for a simbot. Less frequent global updates do not cause disconnections because the simbots whose main task is network maintenance tend to remain stationary when there is no updated feature information. This finding is important because it proves the usefulness of the hierarchic communication approach. If necessary the required bandwidth could be decreased by communicating the global features less frequently without increasing the risk of having a disconnected network.

In this research a global map was used because of its ease of implementation. However this implementation has proven to be inadequate for multi-simbot exploration when large numbers of simbots ($n > 14$) are involved. Previous research (Rodriguez-Losada, 2004) has shown that an *EKF* implementation based on local maps is adequate for multi-robot exploration, but its implementation is beyond the scope of this thesis and remains as a future area of research for BERODE-2. The following subsection presents a summary of the main contributions of this chapter.

10.6.1 Contributions

The main contributions from this chapter are:

- The validation of the scalability of BERODE-2 with respect to the consistency of the maps. As more simbots are added to the network the maps are more consistent at the three levels of comparison: feature, metric and topologic level of the map.

- Analysing the surplus of unmatched features in the current implementation. For simplification the integration of corrections from the *EKF* was omitted in the features sent to the simbot network. This omission generated a surplus in the number of features. It was shown that this surplus was not related to the general architecture of BERODE-2 but to a specific feature of the current implementation.

- The confirmation of previous findings by Fenwick et al. (2002) which show that the reduction in the uncertainty of the feature and robot positions is accelerated with respect to the number of robots due to the combined information from multiple robots.

- The validation of the existence of a minimum percentage of time as a fully connected network for any simbot size when global updates are infrequent. This minimum was found in Section 9.4.1 (page 286). The minimum percentage exists despite the delay problem with the implemented Map Interface Module. That only affects the value of this minimum but does not compromise its existence.

- Finding that the minimum percentage of time as a fully connected network is independent of the global update frequency when the global features do not omit corrections (certain version of BERODE-2).

- The assessment of the effect of sensor uncertainty on BERODE-2. Uncertainty in the positions of the features and the simbots makes the task of keeping the simbots in a fully connected network more difficult. The exploration time increases when there is uncertainty because the network becomes disconnected more frequently than when there no uncertainty. Time is then spent in restoring the full connectivity of the network.

Chapter 11

Conclusions

This chapter begins by summarising the thesis. This is followed, in Section 11.2, by the achievements and contributions. Section 11.3 presents the limitations of the work. Finally, in Section 11.4, ideas on further avenues of research are discussed.

11.1 Thesis Summary

This thesis has presented a novel architecture called BERODE (BEhavioural ROle DEcentralized) to efficiently explore and map an initially unknown environment using a group of robots with local communication capabilities. Our strategy to achieve exploration efficiency consists of the minimization of task overlapping. For this reason in BERODE the robots are kept as a single connected and adaptable communication network to guarantee the coordination between the robots. The robots coordinate their actions by assuming behaviours that depend on their direct communication connections. This improves efficiency by allowing varying number of robots to take the Explorer role depending on circumstances. Our simulations suggest that BERODE in practical implementations will be robust to infrequent communication, scalable in terms of communication and number of robots, and will explore the environment more efficiently than networks that rely on fixed control networks. The implementation we have tested has a scalability problem with respect to the amount of map features. We have shown reasons to suppose that a modification to the implementation will greatly raise this ceiling (Section 10.3.2, page 325).

11.1.1 Exploration with Multiple Robots

We began in Chapter 1 by discussing the use of groups of robots in hazardous environments. These scenarios are typically partially or completely unknown. Robots working in these scenarios need to build a representation of the environment as they explore it looking for potential hazards (e.g. mine removal, search and rescue, surveillance, etc.). In these scenarios typically the communication between robots is possible only within a limited range.

As was seen in Chapter 2, two types of approaches have been proposed to explore an initially unknown environment: Centralized and Decentralized. These approaches have failed to achieve robustness, scalability and efficiency simultaneously. Centralized approaches are not scalable because a central agent coordinates and generates plans for all the robots. This agent gathers and distributes information frequently. As more robots are added to the network the required communication bandwidth increases exponentially and becomes a bottleneck. In decentralized approaches the robots frequently explore areas that have been already explored by other robots because there is no coordination in the exploration. Previous research in exploration remarks that the advantage of having several robots is exploited more effectively when the robots coordinate their actions. To coordinate the robots they have to remain within communication range. Hence, a decentralized architecture where the robots explore the environment in a coordinated fashion while keeping the connectivity of the communication network that they form seems to be a promising approach to integrate the scalability and robustness from distributed architectures, and the efficiency from centralized approaches. That is the kind of architecture developed in this thesis.

11.1.2 The Feature Map Built by the Robots

An important aspect in exploration is to decide the type of map representation that the robots are going to build. As the robots move through the environment they have to concurrently build the representation and localize themselves within this representation. There has been plenty of research in this area which is known as Simultaneous Localization and Mapping (*SLAM*).

The main motivation of this research was that our BERODE architecture should be suitable for implementation in *low cost* robots (e.g. Millibots (Navarro-Serment et al., 1999)). Most previous research has used expensive precision sensors such as lasers to reduce the problems that multiply as uncertainty increases. Our approach is to reduce

uncertainty by combining cheap sensors of different types rather than using one much more expensive sensor. *Low cost* small robots are suitable for some exploration tasks because they are expendable and teams can afford losses. We designed and built a *low cost* sensing platform based on sonars and infrared sensors. The platform had a cost of around £50. We argue that these sensors currently have the best trade-off between cost, and sensor characteristics (range and uncertainty).

In Chapter 3 we discussed the suitability of the different *SLAM* approaches for their implementation in *low cost* robots. An *EKF* (Extended Kalman Filter) approach was regarded as the most suitable *SLAM* approach for building maps with *low cost* robots because its consistency and convergence properties have been proved for single and multiple robot scenarios. The type of map that is built and updated with an *EKF* is a feature based map. The *EKF* approach is less computationally expensive than the other approaches. Nevertheless its cost is still high O(N^2), where N is the number of features. The use of *EKFs* based on local maps has been proved to reduce the cost to O(k^2), where k is the number of features in the local map (Rodriguez-Losada, 2004).

11.1.3 The BERODE (BEhavioural ROle DEcentralized) Architecture

Chapter 4 introduced the BERODE architecture and describe it in terms of the general team goals of efficient exploration and mapping. In our architecture each robot builds its own feature map of the environment. Chapter 5 describes the architecture from the point of view of an individual robot. The robots periodically transmitted beacon signals and their observed features. The beacon signals contain the estimated position of the robot and its uncertainty. The robots incorporated feature observations of other robots with the same process as for locally extracted features because they shared same global Cartesian frame of reference, which was the initial position of the robot with the smallest ID[1] number. We considered that an environment was explored entirely when any one robot considered that its feature map was complete. This occurred when the feature map was projected into a probabilistic grid map and the size of the portions of the environments for which there is no evidence were below a used defined threshold

We proposed to keep the robots as a single connected network to minimize task overlapping and improve coordination. The robots are kept as a single connected communication network by creating and updating an MST (Minimum Spanning Tree) con-

[1] The robots have an ID number that allows them to identify each other in the network.

trol network. The MST control network contains only the minimum necessary links keep the communication network connected. We proposed several heuristics to calculate the MST control network based on the connections of the communication network and an abstraction heuristic.

We introduced a behavioural role approach for the robots, where the robots selected their behaviour based on the current MST control network. The exploratory behaviour of the robot network emerged from the interaction of the individual behaviours of the robots. The interaction is achieved through the imposition of virtual forces derived from the connections in the MST control network

We proposed four behavioural roles: Explorer, Maintainer, Pusher and Recoverer. The behavioural roles balance between the tasks of exploration and network maintenance. The Explorer role was focused on exploration while the rest of the roles were focused on keeping the network connected. The number of Explorer robots changes depending on the configuration of the environment and network topology. The non-Explorer robots kept the communication network connected by generating short-term plans to move to locations where the energy from the virtual forces was minimized. We developed a novel force model were the forces are modelled as *virtual heterogeneous springs*. We described the spring forces as heterogeneous because they were asymmetric[2] and their free spring length was a range of values rather than a single value. The free spring length was a function of the quality of the connection and the behavioural roles of the pair of robots that form the connection. The Explorer robots were not directly subject to spring forces; instead these robots guided the exploration process by generating a plan to move to the safest unexplored area in terms of communication quality and moving in that direction to the limits of their communication safety, in effect dragging the rest of the network along.

The computational cost of planning in BERODE was small because the robots mostly generated short term plans to move towards close locations. With respect to planning, most of the time each robot maintained a local view of its map. Only when an Explorer robot did not found close unexplored areas it did acquired a global view of the map to search for unexplored areas in this map.

The communication network connections change as the robots explore the environment. Maintaining a fixed control network is then a difficult task, thus an adaptive MST control network seems to be more suitable for these dynamic conditions. We pro-

[2] Asymmetric forces are forces that can have different magnitudes and signs in the two directions of the connection.

posed that during the exploration the robots could recalculate the MST control network to improve signal quality of the MST connections.

One of the important goals of this thesis is to achieve scalability with respect to the number of robots. As more robots are added to the network the communication bandwidth required increases, becoming a bottleneck for the system. We proposed a two level approach for communication to achieve scalability, where the levels were: local and global level. The local level for a robot was formed by the robots that were within a *k-hop* distance in the MST control network. We argued that the degree of required coordination for a pair of robots is related to their *k-hop* distance on the MST control network. Information is then shared frequently at the local level and less frequently at the global level.

In Chapter 5 we presented the implementation of BERODE in individual robots. The architecture was implemented using modules. Each module addressed a specific task (e.g. the planning module built the path that the robot had to follow). The robots' control architecture was the same regardless of their behavioural roles. The modules were sequentially executed. The implementation and parameterization of some of the modules depended on the behavioural role.

In Chapter 6 we presented the mapping module used by the robots to build the maps in BERODE. The module implemented an *EKF* that built a feature map representation of the environment. The environment was represented using line and point features. A model to extract features from multiple viewpoints was presented.

11.1.4 The testing of BERODE in Simulation

In Chapter 7 we presented the simulator that we use to test the BERODE architecture. This chapter describes the *LOS* and *RF* communication models that were used in our simulations. We used these communication models to test BERODE in simulation because they are the most used in implementations with multiple real robots. Experiments that show that the simulated sensors and communication devices are reasonable and conservative approximations to the experimental data were discussed.

The simulated *LOS* and *RF* communication models modelled the delays caused by retransmissions in the *MANET* (Mobile *ad hoc* network). The effect of interference was modelled by delaying the messages a random time with a certain probability. In the simulated *LOS* model any obstacle in the direct path of the signal blocked the entire signal. The signal strength for the simulated *RF* model is determined using

Rappaport's model (Rappaport, 2001). In this model the signal strength depends on the distance between the robots, the number of obstacles between them and the attenuation caused by the obstacles. The attenuation of the *RF* signal for different materials was obtained from experiments with a Bluetooth hardware device. The effects of multi path reflections are modelled by adding Gaussian Noise (with mean zero) when there is no *LOS* between transmitter and receiver.

To have meaningful simulations it is important to model relevant aspects of the type of performance achieved by a real robot. Our motion model is based on parameters obtained from experiments with a Koala robot. Our simulated robot incorporates the veering phenomenon observed in real robots due to the uneven wheel traction.

We built an experimental *low cost* sensing platform based on sonars and infrared sensors[3]. The simulation models for the sonar and infrared sensor model the accuracy and obstacle detection reliability observed in tests with these hardware devices. We tested them using different types of obstacle and materials. Our simulation models are based on the worst case observed for the tested obstacles in comparison with our test results. The simulation models are conservative.

We described a novel technique to combine the information from sonar and infrared sensors. We conducted two investigations to assess the mapping module using the *low cost* sensing platform. In the first investigation a real robot built a map of a corridor. In the second investigation a simulated environment that contains an exploration loop is used to highlight the advantage of combining sonar and infrared sensors over the use sonar alone information. It was observed that the combined approach generated more accurate and compact map representations of the environment than that of sonar alone sensing. Moreover, the representations were straighter. The sonar alone approach generated curved representations of straight lines such as corridors. We concluded that the combination of sonar and infrared produced better estimates of the robot's position than sonar alone; moreover the stochastic maps are more compact and accurate.

Chapter 8 presented initial simulations to test the implementation of BERODE for the simulated *RF* and *LOS* models. The simulated robots were referred to as simbots. The mapping module described in Chapter 6 was improved by adding mechanisms to cope with problems present in multi-simbot scenarios (e.g. detecting simbots as obstacles). The implementation of BERODE for the *LOS* model was adapted to cope with the on/off nature of line of sight. In this implementation we incorporated obsta-

[3] Our platform described in Section 7.7 (page 207) uses Devantech SRF04 sonars and Sharp GPxx infrared sensors.

cle forces to the force model used to keep network connectivity to direct the simbots towards open areas where the communication was less likely to be lost.

An initial simulation revealed poor performance of BERODE when exploration was inefficient or temporarily ceased. We implemented two procedures to address this problem. The procedures were a mechanism to exchange roles and a mechanism that periodically checked that there was at least one Explorer robot in the network. When there was no Explorer robot in the network a tendering mechanism was triggered. The tendering mechanism was used to determine which candidate Explorer robot to choose. The simulations showed that the procedures improved the efficiency in the exploration at a cost of a small increase in the required communication bandwidth (10% in the worst case). The improved version was used in subsequent simulations and was referred as BERODE-2.

11.1.5 Robustness, Scalability and Efficiency of BERODE-2

In Chapter 9 we presented simulations that assessed the robustness, scalability and efficiency of the improved version of BERODE. The simulations showed that BERODE-2 was robust with respect to frequency of local level transmissions because the percentage of time that the simbots remained as a single connected network slowly decreased as the frequency of the local level transmission was decreased. Moreover, we found that there is a minimum percentage of time that a simbot network of any size remains as a single network when the distance that the simbots move between beacon signals is small ($<$0.30m). The existence of this minimum percentage is not a function of the frequency of transmissions at the global level. This suggests that BERODE-2 is scalable in terms of communication because the task of network maintenance is independent of the frequency of the transmissions at the global level.

The simulations showed that adding more simbots to BERODE-2 decreases the exploration[4] time. The decrease was logarithmic at worst and supports the findings from previous research (Burgard et al., 2006) that suggest that there is a maximum number of robots that can efficiently explore an environment of a certain size.

We found an inversely proportional trading between the exploration time and the network ratio. The network ratio is the ratio between the sizes of the local network and the entire robot network. We showed that for small network ratios the trading between exploration time and network ratio was the best. Our simulation showed that

[4] The exploration time was the time that the robots required to built a complete map of the environment. The exploration stops once any one robot has a complete map.

BERODE-2 was more efficient, scalable and robust with respect to communications than fixed control networks. BERODE-2 is more efficient because it requires less time to explore the environment. BERODE-2 is more scalable because it had a much smaller decrease in the time that the robots remain as a single connected network when the number of robots was increased. It had a better success rate at finishing the exploration tasks (95.6% against 70.3% in our simulated environments), which suggests that in general it would be more robust.

11.1.6 Map Consistency of BERODE

In BERODE-2 each simbot built its own feature based map. The maps generated by the simbots frequently contained small differences. Chapter 10 presented simulations that analyzed the consistency of the maps built by different simbot network sizes. The consistency of the maps was analyzed at the feature, topologic and metric level. The simulations showed that the maps were more similar at the three levels when the number of robots was increased. Although it was expected that the maps would contain a few repeated features, it was observed that the number of repeated features grew when the number of simbots was increased. The increase in the number of repeated features was found to be caused by the delay in the integration of the information.

Our analysis of the simulations revealed that the increase was specific to the current implementation of the map building module. In this implementation the corrections realized by the *EKF* were not integrated into the features that the simbots shared periodically. It was concluded that given its ease of implementation and its low additional cost in computational terms the current implementation is suitable for medium simbot network sizes ($n \leq 14$)[5].The implementation of an *EKF* based on local maps was suggested as a solution to the repetition problem. This implementation has been shown to be robust to delays (Rodriguez-Losada, 2004). While simple in concept this would have involved a radical reimplementation beyond the scope of this thesis. The idea however, was tested by removing the delays in the integration of information. This showed that the repetition problem disappears when the features do not omit the corrections from the *EKF*. The additional cost in terms of communication bandwidth for the *EKF* based on local map was estimated to be around 10%.

In our simulations we assessed the effect of sensor uncertainty in BERODE-2. It was concluded that, as expected, when there was more uncertainty in the positions of

[5] In our office test environments. This number will reduce as map features increase, and vice versa.

the features and the simbots the network was more likely to become disconnected. The robots' use their feature maps to generate their plans. In the presence of uncertainty the maps contained inaccuracies (e.g. inaccurate estimation of the extremes for the line features). These inaccuracies in the maps occasionally caused bad planning decisions that cause the disconnection of the simbot network. The simbots then required more time to build the maps because they had first to restore the connectivity of the network before continuing with the exploration of the environment. Nevertheless the network was kept as a single connected network most of the time (≅85%).

11.2 Summary of Contributions

The main contribution of this thesis is the description of a new architecture for multi-robot exploration that is more efficient, scalable and robust with respect to communications than fixed control networks. Our simulations from Section 9.6 (page 295) compared BERODE-2 with the fixed control networks and showed that:

1. Efficiency: BERODE-2 requires less time to build a complete map of the environment than the fixed control networks.

2. Scalable: When the number of robots was increased, the decrease in the time that the robots remained as a single connected network was much larger for the fixed control network than for BERODE-2. Moreover for BERODE-2 there was a minimum time that the robots remain as a single connected network regardless of the size of the robot network. This was not the case for the fixed control networks.

3. Robustness: In our specific test environments BERODE-2 showed a better success rate (95.6% against 70.3%) at finishing the exploration task than the fixed control networks. The exploration task failed to finish when the network became disconnected and the robots fail to re-establish the connectivity.

The distinctive features of BERODE with respect to previous fixed control network approaches are:

- Adaptive MST (Minimum Spanning Tree) control network: We have proposed a novel adaptive approach to keep a network of robots connected as they traverse an environment. Most of the previous approaches have relied on *a priori* control relations to keep the network fully connected.

- Role Based Distributed Control: We proposed a novel role based approach to achieve coordination in a decentralized fashion. The robots adopted a behavioural role based on the current MST control network. The behavioural roles balance between the tasks of exploration and network maintenance.

- *Heterogeneous virtual force* model: We proposed a novel virtual force model to keep the robots as a single connected network. The forces were modelled as asymmetric[6] virtual springs with a range rather than a point of null force. The magnitude of the forces depended on signal quality and the behavioural roles of the pair of robots that formed the communication link.

- Hierarchic communication levels: We proposed a hierarchic approach for information distribution of two levels local and global to reduce the communication costs.

- Variable number of robots exploring the environment. In BERODE the number of robots exploring the environment changes depending on the configuration of the environment and network topology. In previous approaches there is only one robot that is in charge of the exploration while the rest of the robots keep the network connected. In some of these approaches the robots try to explore the environment when possible; however there is only one robot that guides the exploration. Having the possibility of several Explorer robots is shown to give a useful increase in mapping speed.

BERODE-2 was inspired by recently proposed *low cost* robot platforms such as Millibots (Navarro-Serment et al., 1999). These platforms focused on the deployment of large numbers of robots. Our simulations suggest BERODE-2 is more suitable for implementation in *low cost* robots than previously proposed fixed control networks.

Our simulations showed that BERODE-2 is scalable and robust with respect to communication because:

1. The connectivity of the robot network depends only on the periodical sending of beacon signals (Section 9.3, page 270 and Section 9.4.1, page 286). If beacon signals are frequent enough (in our specific implementation every 0.3m for robots that travel at 0.15 m/s) in BERODE-2 the percentage of full connectivity asymptotically approaches a high minimum (in our implementation about 80%)

[6] An asymmetric force is a force that can have different magnitudes and signs in the two directions of the connection.

as the number of robot increases, whereas in fixed networks connectivity keeps decreasing.

2. The percentage of time that the network remains connected depends only on the periodical exchange of local mapping information (Section 9.4.1, page 286). This percentage depends only on the size of the local neighbourhood and the exchange frequency at the local level.

In our simulations (Section 9.4, page 277) we found that the trading between the exploration efficiency and communication cost is best when the size of the local networks is in the range [0, min($n^{-0.5}$, 2/5)] where n is the number of robots in the network. In this range of the increase in communication bandwidth with respect to the number of robots is linear. The exploration efficiency always improves when the size of the local network is increased but at an increasingly slower rate. There is not much improvement when the size of the local network is bigger than $2n/5$. For large robot networks ($n \geq 40$) there is an accelerating increase in the communication bandwidth space when large local networks are used. This accelerating increase will soon surpass the capacity of available communication technologies as the number of robots increases. This suggests that BERODE could be implemented in larger robot networks because less bandwidth is required since most of the information only needs to be transmitted locally to ensure network connectivity.

One of the important aspects to achieve multirobot coordination is the communication range. Although having robots with an effectively unlimited communication range may seem in principle attractive it is undesirable for the purposes of scalability and interference. The bandwidth required increases as more robots are added to the network. Moreover when power consumption is a constraint in the system it is desirable to restrict communication to short distances because the amount of power required for transmitting a signal in free space is a function of the square of the distance. In indoor environments the situation is much worse because of the structure and the obstacles.

In our simulations (Section 9.5, page 289) with the *RF* model we identified a communication range threshold above which the exploration efficiency ceases to improve for a robot network. This finding is important because communication can be restricted to short distances to reduce power consumption without decreasing the exploration efficiency.

In Fenwick et al.'s (2002) research it was shown that the reduction in uncertainty is accelerated when more robots are added to the network. This is due to the combined information from multiple robots. Our simulations (Section 10.3, page 316) confirm this finding for our simulated *low cost* robots and showed that the consistency of the maps increased when larger robot networks are used. This confirms our suggestion that large teams of expendable *low cost* robots can be used to produce maps which are useful not only for the robot's own navigation, but for human use.

We can't conclude that these properties and findings would necessarily transfer to a real world implementation, but we have taken steps (Chapter 7) to make our simulations as reasonable and conservative in the relevant aspects. Section 11.5 discusses details of how the BERODE architecture could be implemented using real robots.

11.3 Limitations

Our simulated environments were static with respect to communication. In real world environments, frequently the communication conditions change over time (e.g. temporary disruption of communication due to the movement of elevators in a building). In our simulations we decreased the frequency of communication between the robots. The simulations showed a proportional decrease in the time that the network was kept connected as the frequency was decreased. This suggests that the performance of BERODE-2 degrades gracefully rather than catastrophically as the amount of temporary disruptions in an environment becomes larger.

Our simulated environments did not contain dynamic obstacles. Real world environments are dynamic; however the problem of reliably building maps in dynamic environments is still an open issue even for a single robot.

There are currently three limitations in the implementation of BERODE-2 related to the problem of building maps. These limitations are still open issues and are:

- The problem of decision making when simbots have inconsistent maps that cause the simbots to take different decisions. In the simulations the simbots used *a priori* structural knowledge (e.g. parallelism, perpendicularity) about the office-like environment. The use of the *a priori* structural knowledge is optional and encoded in a few user defined parameters. The use of the structural knowledge allowed the simbots to build maps that contained only small metric differences. Preliminary simulations showed that for more than a few simbots it was difficult

to obtain good position estimates when no structural knowledge was used. The maps then had large inconsistencies that made it difficult to assess the performance of BERODE.

- There is currently a limitation in the size of the environment that the simbots can explore. The current map building module uses a global feature map that is updated using an Extended Kalman Filter (*EKF*). The computational cost of the update of the map is O(N^2) where N is the number of features in the map. *EKFs* are regarded as the best performers in this area; their O(N^2) computational cost is an inevitable consequence of this approach and can be handled using local maps.

- The current implementation of the map building module has a surplus of unmatched features problem. This surplus increases as the number of simbots increases. The implementation is suitable for medium team sizes ($n < 14$). Our experimental simulations suggest that the implementation of an *EKF* based on local maps would solve this problem for larger team sizes.

11.4 Future Research

The BERODE architecture looks promising as a worthwhile architecture for future research. Some future research could be conducted in simulation or reality according to the resources available, whereas some is probably best done in real implementations. This section is biased towards simulation, the following section describes the main points that should be considered in a real world implementation of BERODE. Some interesting developments for future research would be:

- Developing an adaptive *virtual* force model: The implementation of BERODE for the *LOS* model in Chapter 5 was based on the argument that the difficulty to maintain a connection between a pair of robots was related to their Euclidean distance. For robots that are distant any small movement in structured or cluttered environments could easily compromise the connection. The thresholds for the proposed *virtual spring* model were based on this argument. The implementation of an adaptive threshold approach should improve the performance of BERODE in environments with different degrees of clutter (e.g. warehouse, cluttered office, narrow corridor). The robots could adjust the thresholds based

on the degree of clutter on their current map and on the reliability of their connections. For instance in a cluttered office where the communication is unreliable the robots should have higher thresholds than in large empty halls.

- Managing the communication range to improve power consumption and minimize interference: In Chapter 9 we identified a threshold for the communication range. Robot networks with communication ranges bigger than the threshold require the same time to build a complete map of the environment. The threshold range covers a minimum area in the environment for the *RF* model. The area covered by a signal with a certain range depends on the clutter of the environment. The threshold could be used by the robots to adjust their communication range dynamically by estimating of the percentage of area that their signal covers according to their currently built map.

- Implementation of an *EKF* based on local maps. The use of local maps has been suggested as the solution to the surplus feature problem of the current implementation. Losada (Rodriguez-Losada, 2004) has shown that an *EKF* based in local maps generates maps as consistent as the maps generated with a global *EKF*. The implementation of an *EKF* based in local maps is time consuming. Future research should contemplate its implementation.

- Robots sensing other robots to improve position estimations. In the current implementation the robots do not integrate observations of each other to improve their position estimations. Future research should contemplate the use of these observations to improve the accuracy of robot position estimation, and consequently the accuracy of the map.

11.5 Implementation of BERODE for Real Robots

This section describes research which would more appropriately be done in a real implementation of BERODE. There are several research issues related to the implementation with real robots. These issues and ideas to solve them are listed as follows:

- Sensors of the same kind have important difference in performance. These differences in performance can be reduced by performing a calibration routine at the beginning of the exploration process. For instance, the robot sensors can be calibrated by sensing an obstacle located at a known distance.

- Communication *hot spots* can impair the exploratory behaviour of the team of robots. To keep the network connected a robot moves to a location where the signal quality is predicted to be better. A robot located close to a large enough *hot spot* will remain in this area halting the exploratory behaviour of the robot network. Simple methods of detecting and breaking out of such traps need to be devised and tested.

- Improve robustness to cope with the partial or total malfunction of robots in the network. For instance a robot with a broken sensor is likely to extract false features. The integration of these features to the map will generate inconsistencies. Validation mechanisms should address this issue. In these mechanisms the reliability of the features could be associated with the number of robots that have observed the feature.

- Improve robustness of the map building module to cope with dynamic environments. Real environments are dynamic. In the current implementation once a feature is incorporated into the feature map it is never removed. In our simulations we observed that the false detection of point features slowed down the exploration process because the robots use the feature map to keep their connections within communication range. In a real world implementation the static world assumption should be removed. Mechanisms to remove features that have recently been observed to be absent should be incorporated. This would allow the implementation of BERODE in dynamic environments. The mechanism to remove features can be computationally expensive if viewed as a purely computational problem but the robots can also go and check. A designer can then implement the most adequate solution depending on the relative costs of computation and movement for that specific robotic platform.

- The relative initial known positions. In our simulations we have assumed that the robots start off at known locations. In a real world implementation the robots can be launched sequentially from the same initial position. More general would be to use the robots sensors to sense each other; the robots can be dropped at close locations and execute an initialization routine where they determine their relative initial locations to a high accuracy by using their numbers, close proximity, and repetition.

Appendix A

Consistency Property of a Local MST

The property of consistency is verified formally visualizing the robot network as a graph where the robots are the nodes and the edges are the network connections.

In general for a graph $G(z,e)$ with z nodes, e edges has an MST=$G(z,z\text{-}1)$ where $MST \subseteq G(z,e)$. η is the set of z nodes, ε is the set of the e edges and $e_{i,j}$ represents the edge between nodes i and j. Two nodes z_i and z_j whose edge $e_{i,j} \notin \varepsilon$ have sub graphs $G_i(z_i,e_i)$ and $G_j(z_j,e_j)$ for their k neighbourhood where $G_i(z_i,e_i) \subseteq G(z,e)$ and $G_j(z_j,e_j) \subseteq G(z,e)$.

Theorem A.1 (*K-MST Joining*)

If the edge $e_{i,j}$ is added to the set ε for any $MST_k = G(z_k, z_k - 1)$ of the sub graph $G_k(z_k,e_k) = G_i(z_i,e_i) \cup G_j(z_j,e_j)$ the $MST_n = MST_k \cup MST_k^{'}$ where $MST_k^{'} = (MST_i \cup MST_j)'$ is an MST of $G(z,e)$ only when G_i and G_j share at least a common edge e_m ($e_m \in \varepsilon_i, e_m \in \varepsilon_j$) prior to the addition of $e_{i,j}$.

Proof:

MST_n is an MST of G if the number of edges in MST_n is $n(\varepsilon_n) = z - 1$ prior and after the addition of the edge $e_{i,j}$. The nodes i and j with sub graphs $G_i(n_i,e_i)$ and $G_j(n_j,e_j)$ have $MSTs$ with $n(\varepsilon_i) = (z_i - 1)$ and $n(\varepsilon_j) = (z_j - 1)$ edges respectively, the number of edges in $(MST_i \cup MST_j)'$ is

$$n(\varepsilon_k') = n(\varepsilon) - \big(n(\varepsilon_i) + n(\varepsilon_j) - n(\varepsilon_i \cap \varepsilon_j)\big) \tag{A.1}$$

$$n(\varepsilon_k') = (z-1) - (z_i - 1) - (z_j - 1) + n(\varepsilon_i \cap \varepsilon_j) \tag{A.2}$$

$$n(\varepsilon_k') = z + 1 - z_i - z_j + n(\varepsilon_i \cap \varepsilon_j) \tag{A.3}$$

If nodes i and j have q common edges $\left(n(\varepsilon_i \cap \varepsilon_j) = q\right)$ before their edge $e_{i,j}$ is added to the set ε the number of nodes $n(\eta_k)$ in G_k is $n(\eta_k) = z_i + z_j - q$ and $G_k(z_k, e_k)$ is a connected graph. Any MST_k has $n(\varepsilon_k) = z_i + z_j - q - 2$ edges. The number of edges in MST_n is $n(\varepsilon_n) = n(\varepsilon_k) + n(\varepsilon_k') = z - 1$ therefore MST_n is an MST of G. If the $e_{i,j}$ is added to the set ε the number of edges in $G_k(z_k, e_k)$ increases but the number of edges in any MST_k remains the same $n(\varepsilon_k)$. Then for any MST_k the number of edges in MST_n is

$$n(\varepsilon_n) = \left(n(\varepsilon_k) + n(\varepsilon_k^{'}) - n(\varepsilon_k \cap \varepsilon_k^{'})\right) \tag{A.4}$$

$$n(\varepsilon_n) = (z_i + z_j - q - 2) + \left(z + 1 - z_i - z_j + n(\varepsilon_i \cap \varepsilon_j)\right) - n(\varepsilon_k \cap \varepsilon_k^{'}) \tag{A.5}$$

where $n(\varepsilon_k \cap \varepsilon_k^{'}) = 0$ and $n(\varepsilon_i \cap \varepsilon_j) = q$

$$n(\varepsilon_n) = (z_i + z_j - q - 2) + \left(z + 1 - z_i - z_j + q\right) - 1 = z - 1 \tag{A.6}$$

Therefore $MST_n$is a MST of the graph $G(z,e)$.

If before the edge $e_{i,j}$ is added to the set ε the nodes i and j with graphs $G_i(z_i,e_i)$ and $G_j(z_j,e_j)$ respectively do not share at least a common edge their graphs are disjoint, therefore $MST_i \cup MST_j$ is not an MST of $G(z,e)$. If the edge $e_{i,j}$ is added to the sub graph $G_k(z_k, e_k)$ becomes a connected graph. The edge $e_{i,j}$ is a *bridge*[1] of G_k. In a MST there is only possible path between any two nodes. Then any MST_k contains the edge $e_{i,j}$ and has

$$n(\varepsilon_k) = n(\varepsilon_i) + n(\varepsilon_j) + 1 = (z_i - 1) + (z_j - 1) + 1 = z_i + z_j - 1 \tag{A.7}$$

edges. If $e_{i,j} \subset (MST_i \cup MST_j)'$ then $n(\varepsilon_i \cap \varepsilon_j) = 1$. Applying Eq. A.5 is proven that MST_n is a MST of the graph $G(z,e)$. If $e_{i,j} \not\subset (MST_i \cup MST_j)'$ then $n(\varepsilon_i \cap \varepsilon_j) = 0$. Applying Eq. A.5 is proven that MST_n is no longer a MST of the graph $G(z,e)$ because the number of edges in MST_n is z.

[1] A *bridge* is a vertex whose removal disconnects the graph.

Appendix B

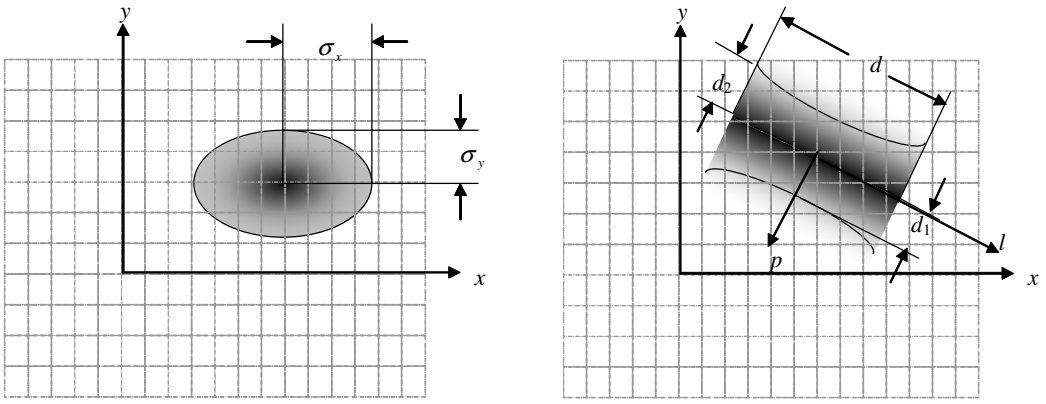
Feature Map Projection

The Map Building Module of BERODE extracts line and point features from the sensors. The planning modules of BERODE use a probabilistic grid map to plan robot movements and to predict the signal quality of their connections. The probabilistic grid map is built using the point and line features. The projection incorporates the uncertainty and viewpoints of the feature. The uncertainty of the parameters is obtained from the *EKF* in the Map Building Module. The viewpoints are positions from which the feature was observed. These viewpoints are grouped in clusters for storage and computational efficiency. Figure B.1 shows an example of the projection of the uncertainty of the point and line features in the probabilistic grid map. A point feature $P(x,y)$ has uncertainties σ_x and σ_y along the x and y axis respectively. A line feature $L(\rho, \phi)$ with a length d has uncertainties σ_ρ and σ_ϕ. The middle point of the line has an uncertainty $d_1 = \sigma_\rho^2$ along the perpendicular (p axis) to the line orientation. The uncertainty along the p axis is $d_2 = (\sigma_\rho + (d\sigma_\phi))^2$ at the extremes of the line.

The probabilistic grid map is updated according to Bayes theorem. Following Thrun's (Thrun, 2000c) formulation a grid cell m_{xy}^t in the map m is updated at time t as follows

$$P(m_{xy}^t) = 1 - \left[1 + \frac{P(m_{xy}^t|z_t,x_t)}{1-P(m_{xy}^t|z_t,x_t)} * \frac{1-P_{priori}(m_{xy}^t)}{P_{priori}(m_{xy}^t)} * \frac{P(m_{xy}^{t-1})}{1-P(m_{xy}^{t-1})}\right]^{-1} \quad \text{(B.1)}$$

where $P(m_{xy}^{t-1})$ is the probability of a cell of being occupied before the map is updated at time t. $P_{priori}(m_{xy}^t)$ is *a priori* probability of a cell of being occupied. This probability is user defined and is typically set to $P_{priori}(m_{xy}^t) = 0.5$ in indoor environments. $P(m_{xy}^t)$ is the probability of a cell of being occupied after the cell is updated. $P(m_{xy}^t|z_t,x_t)$ is the probability of a cell of being occupied for a measurement z_t taken



a) Projected Point Feature b) Projected Line Feature

Figure B.1: An example of the projection of the uncertainty of the features in a probabilistic grid map for a a) point and a b) line feature.

from position x_t. In our projection the viewpoints are the positions and the measurements contain the feature parameters and uncertainty. A feature may have been observed more than once from a viewpoint because the viewpoints represent clusters of the observations. The clusters are formed using a nearest neighbour approach. Every time that a feature is observed from a new viewpoint a new cluster is created if the distance between the new viewpoint and the closest cluster is above a certain user defined threshold. The Eq. B.1 is then applied a proportional number of occasions with respect to the number of observations.

The projected grid map is built from scratch for each plan. At the beginning of the projection all the grid cells are set to $P(m_{xy}^{t=0}) = 0.5$. The grid cells that lie inside the uncertainty space of the features are updated as occupied space ($P(m_{xy}^t|z_t,x_t) = P_{oc}$). The grid cells that lie inside the free space generated by the viewpoints of a feature are updated as free space ($P(m_{xy}^t|z_t,x_t) = P_F$). The occupancy model for a point $P(x,y)$ is

$$P_{oc} = P_{MAX} - (P_{MAX} - 0.5)\left(\frac{d(P(x,y),m_{xy}^t)}{\sigma_{x^2} + \sigma_{y^2}}\right) \tag{B.2}$$

where P_{MAX} is a user defined variable and is the maximum probability for a cell of being occupied given that the measurement indicates it is occupied space. In our implementation P_{MAX}=0.8. $d(P(x,y),m_{xy}^t)$ is the Cartesian distance between the point $P(x,y)$ and the grid cell m_{xy}^t.

Occupied Space

The occupancy model for a line $L(\rho, \phi)$ feature is

$$P_{oc} = P_{MAX} - (P_{MAX} - 0.5)\left(\frac{d_P(L(\varphi,\phi), m^t_{xy})}{\sigma_P}\right)^2 \tag{B.3}$$

$$\sigma_P = \sigma_\varphi + d_L(m^t_{xy}, P_c(x_c, y_c)) * \sigma_\phi) \tag{B.4}$$

where $d_P(L(\varphi,\phi), m^t_{xy})$ is the perpendicular distance of the grid cell m^t_{xy} to the line $L(\varphi,\phi)$ (p axis on Figure B.1). $d_L(m^t_{xy}, P_c(x_c, y_c))$ is the distance along the line axis (l axis on Figure B.1) between the grid cell and the middle point of the line $P_c(x_c, y_c)$. σ_P is then the uncertainty along the p axis.

Free Space

The free space model for a point is

$$P_F = P_{MIN} + (0.5 - P_{MIN})\left(\frac{d(v(x,y), m^t_{xy})}{d(P(x,y), v(x,y)) - (\sigma_{x^2} + \sigma_{y^2})^{1/2}}\right)^2 \tag{B.5}$$

where $v(x,y)$ is the position of the viewpoint. P_{MIN} a user defined variable and is the maximum probability for a cell of being free given that the measurement indicates it is free space. In our implementation P_{MAX}=0.2. Figure B.2 presents an example of the projection of a point feature for three viewpoints.

The free space model for a line is

$$P_F = P_{MIN} + (0.5 - P_{MIN})\left(\frac{d_P(v_{av}(x,y), m^t_{xy})}{d_P(L(\varphi,\phi), v(x,y)) - \sigma_P}\right)^2 \tag{B.6}$$

where $v_{av}(x,y)$ is the average viewpoint between two successive viewpoints. The viewpoints of a line are relative to the lp axis of a line. The origin of the line is the middle point of the line. The viewpoints are stored in increasing distance from the line axis (l axis in Figure B.1). The average viewpoints generate free space rectangular shapes on the lp coordinated system of the line. The length of the rectangle is the difference along the line axis between the consecutive viewpoints, while the width is the average of the coordinate in the p axis. The first and last viewpoints are also projected as rectangular triangles where the height of the triangle is the perpendicular distance to the line. Figure B.3 presents an example of the projection of the viewpoints of a line feature. It is observed that the occupied space of the line has a hyperbolic shape while the free space of the projection is comprised by rectangles and triangles.

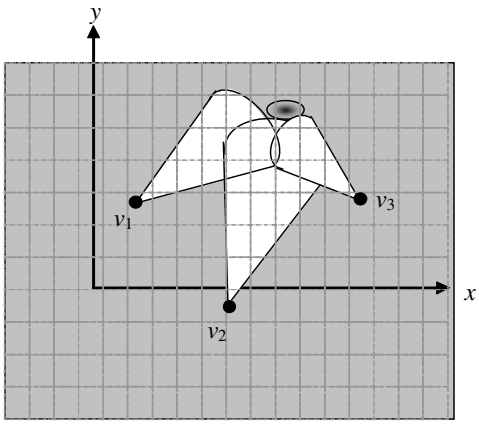


Figure B.2: An example of the projection of a point feature with three viewpoints v_1, v_2 and v_3 in the probabilistic grid map.

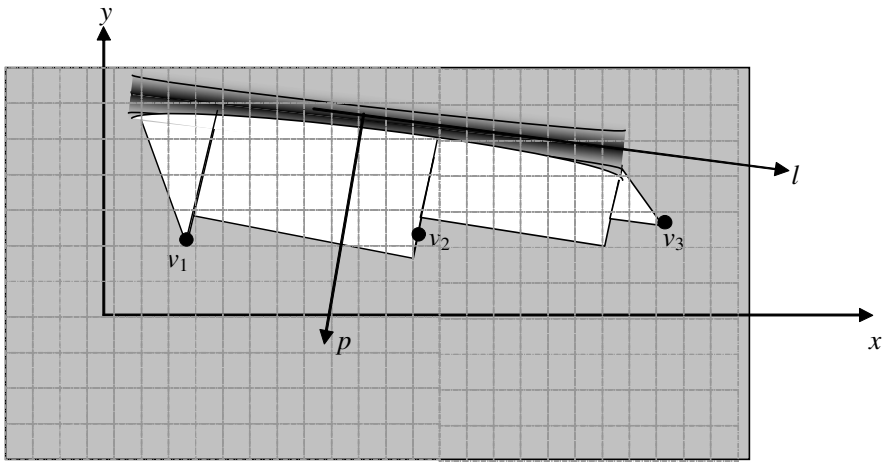


Figure B.3: An example of the projection of a point feature with three viewpoints v_1, v_2 and v_3 in the probabilistic grid map.

Appendix C

The Application Level Communication Protocol

This appendix presents the application level protocol used by the supervisor controller in the simulator to quantify the bandwidth required for the robots in BERODE. The protocol implements three types of communication: *broadcast*, *acknowledged*, and *request*. The supervisor calculates the Bps (bytes per second) required for each robot based on the number of messages received and their content.

C.1 The Application Level Protocol

To efficiently distribute and gather information for the robot network an application level protocol was developed for BERODE. The protocol implements three types of communication: *broadcast*, *acknowledged*, and *request*. In the *broadcast* communication a robot transmits information where the confirmation of reception of the message by the receiver robots is not required. In the *acknowledged* and *request* communication the confirmation of reception by the receiver robots is required. The *acknowledged* communication is used by the robots to communicate local information to a group of robots. The *request* communication is used to gather information from a group of robots.

The protocol implements a message format as follows:

$$<T, ID, N_T, M_T, \eta, Content>$$

where:

T: Type of the message (*brd, ack, reply-ack, req, reply-req*)

ID: name of the sender node

N_T: current transmission level

M_T: maximum retransmission level

η : set of nodes that have received the message.

In a *broadcast* communication the robot sends a message of the *brd* type. In the *acknowledged* communication the sender send a message of the *ack* type and waits for a certain time for a confirmation of reception of the message from the destination nodes, if after this time no confirmation is received the message is retransmitted. The retransmission process is repeated a few times before the process gives up. The *reply-ack* message type is used to confirm the reception of *acknowledged* communications. The *request* message type is a message were a robot queries about properties of other robots. This type of message can be described as a two stage process: *diffusion* and *gathering*. At the *diffusion* stage a *req* message type querying a certain property is sent. At the *gathering* stage all the robots that received the query send their information. The information is sent using a *reply-req* message type. At the *diffusion* and *gathering* stages messages are repeated a few times if necessary in the same way as in the *acknowledged* communications to guarantee the reception of the message.

The MST control network is used as the communication network. A node n_i that receives a message from a node n_j ignores the message if n_j is not a control connection because their connection is not part of the MST control network. The following paragraphs describe the message sequencing algorithm for each message type.

***Broadcast* communication**

For the *broadcast* communication the source node sends an initial message of the type

$$< T\text{=}brd, \quad N_S, 0, M_T, \eta, Content>$$

The initial set of nodes η contains the source node and its control connections. All the robots in η that receive the message and contain N_S as a control connection integrate the information from *Content*. If a robot N_R that received the message has at least one control connection not contained in η then the node adds its control connections that are not contained in η to this list and retransmits the message updating the N_T transmission level as

$$< T\text{=}brd, \quad N_R, 1, M_T, \eta, Content>$$

The process of receiving and retransmission of the message is repeated until $N_T < M_T$.

***Acknowledged* communication**

For the *acknowledged* communication the source node sends an initial message of

the type

$$< T\text{=}ack, \quad N_S, 0, M_T, \eta, Content>$$

The initial set of nodes η contains the source node and its control connections. All the robots in η that receive the message and contain N_S as a control connection integrate the information from *Content*. If a robot N_R that received the message has at least one control connection not contained in η then the node adds its control connections that are not contained in η to this list and retransmits the message updating the N_T transmission level as

$$< T\text{=}ack, \quad N_R, 1, M_T, \eta, Content>$$

This message serves as the acknowledgement message for the original sender. If a robot N_R that received the message does not contain a control connection that is not contained in η then the robot sends an acknowledge message

$$< T\text{=}reply\text{-}ack, \quad N_R, 0, 1, \eta, \text{acknowledge } N_s >$$

If after a certain time the original sender does not receive the acknowledgement from all its control connections the message is retransmitted with an enumeration value

$$< T\text{=}ack, \quad N_R, 1, M_T, \eta, 2\text{:}Content>$$

To avoid the situation in which robots that received the original message integrate in more than one occasions the information of *Content*. The process of receiving, acknowledge and retransmission of the message is repeated until $N_T < M_T$.

***Request* communication**

The *request* communication has two stages: *diffusion* and *gathering*. At the *diffusion* stage a *req* message type querying a certain property is sent. At the *gathering* stage all the robots that received the query send their information. For a request communication the source node sends an initial message of the type

$$< T\text{=}req, \quad N_S, 0, M_T, \eta, Robot\ Tree\text{: } Query>$$

The initial set of nodes η contains the source node and its control connections. *Query* is the property queried and *Robot Tree* is the tree structure of the robots that have to return their information where the root is the robot that originates the request. All the robots in η that receive the message and contain N_S as a control connection wait for the gathering stage to retrieve the information queried (*Query*) and store the *Robot List*. If a robot N_R that received the message has at least one control connection not contained in η then the node adds its control connections that are not contained in η to this list and retransmits the message updating the N_T transmission level as

$$< T\text{=}req, N_R, 1, M_T, \eta, Robot\ Tree\text{: } Query >$$

This message serves as the acknowledgement message for the original sender. If

a robot N_R that received the message does not contain a control connection that is not contained in η then the robot sends an acknowledge message

$$< T\text{=}reply\text{-}req, \quad N_R, 0, 1, \eta, \text{acknowledge } N_s >$$

If after a certain time the original sender does not receive the acknowledgement from all its control connections the message is retransmitted with an enumeration value

$$< T\text{=}req, \quad N_R, 1, M_T, \eta, 2\text{: } Robot\ Tree\text{: } Query >$$

To avoid the situation in which robots that received the original message integrate in more than one occasions the information of *Content*. The message is retransmitted by the robots until $N_T < M_T$. At this point the robots that are leafs in the *Robot Tree* received the request and the *diffusion* stage finishes. The *gathering* stage begins with the robots that are in the M_T level (leafs) sending a message of the type

$$< T\text{=}req, N_S, 0, 1, \eta, Query\text{: } Inf >$$

Where the initial set of nodes η contains the leaf node and its control connections. All the robots in η that receive the message send an acknowledge message

$$< T\text{=}reply\text{-}req, \quad N_R, 0, 1, \eta, \text{acknowledge } N_s : Query\text{: } Inf >$$

As previously explained if the original sender does not received the acknowledgement from all its control connections the message is retransmitted with an enumeration value

$$< T\text{=}req, N_S, 0, 1, \eta, 2\text{: } Query\text{: } Inf >$$

Once a robot received the messages containing the queried property from all its children nodes (according to the *Robot tree* structure) the robot appends these messages in *Inf* along with its property queried and retransmits the message. The process of receiving, acknowledge and retransmission of the message at the gathering stage is repeated until $N_T < M_T$.

Appendix D

Algorithm for LOS prioritization for the RF model

This appendix presents the *LOS* prioritization algorithms for the recalculation of the MST control network and the Network Manager Module for the Explorer robots. As explained in Section 8.5.2 it is better to try to maintain *RF* communication for connections that are on line of sight because this reduces the likelihood of having robots trapped in local minima (communication *hot spots*).

D.1 Prioritization for the MST Control Network Calculation

When a robot decides to recalculate the MST control network in the *RF* implementation the robot gathers the information about the signal quality for the connections on the robot network. The robot determines the connections for which there is line of sight between the robots that form the connection using a ray tracing operation on its projected grid map. If it is possible to build the MST control network using only the connection with *LOS* the recalculation process is called using only these connections, otherwise all the connections are used in the recalculation.

Algorithm 4 shows the algorithm for the prioritized recalculation of the MST control network. The robots form a graph G with a set of connections SQ, where $SQ(i, j)$ is the signal quality value for the connection between the robots i and j. $LOSconnection(i, j)$ is the function that determines if the connection between the robots i and j has a line of sight. *LOSrobot* is a boolean array that is used to determine if a robot has

at least one connection with line of sight. As observed from the algorithm if there is at least one robot with no *LOS* connections the recalculation process is called using all the connections (the *SQ* set of connections), otherwise the process is called using the *LOS* connections (SQ_{LOS}).

Algorithm 4 Calculation of MST control network for LOS prioritization

```
LOSnetwork = true
for i = 1 to n do
    LOSrobot[i] = false
end for
for i = 1 to n do
    for j = i + 1 to n do
        if LOSconnection(i, j) is true then
            LOSrobot[i] = LOSrobot[j] = true
            SQ_LOS(i, j) = SQ(i, j)
        else
            SQ_LOS(i, j) = ∞
        end if
    end for
end for
for i = 1 to n do
    if LOSrobot[i] is false then
        LOSnetwork = false
    end if
end for
if LOSnetwork is true then
    MST ← calculate Network using SQ_LOS
else
    MST ← calculate Network using SQ
end if
```

D.2 Prioritization for the Network Manager Module for the Explorer Robots

The implementation of the Network Manager Module for the Explorer behavioural role monitors the local network and modifies this network if it determines that it can be im-

proved. In the prioritized version of this module the connections are pre-processed to determine if they are *LOS* connections. If there is at least one *LOS* connection with a signal quality above the τ_{safe_exp} threshold the module is called using the SQ_{LOS} array, otherwise the module is called using all the set of connections SQ. The τ_{safe_exp} threshold is the desired signal quality value above which the robot continues the exploration process.

Algorithm 5 shows the algorithm used to pre-process and prioritize the connections for a robot *ID*. Following the notation defined in the previous section, SQ is the set of connections, $SQ(i, j)$ is the signal quality value for the connection between the robots i and j, SQ_{ID} is the signal quality value for the direct connections of robot *ID* and SQ_{LOS} is the signal quality value for the connections that have line of sight.

Algorithm 5 Communication constraint selection using LOS prioritization

$SQ_{ID} \leftarrow$ get Signal Quality for direct connections of robot *ID*
for $i = 1$ to n **do**
  if $LOSconnection(ID, i)$ is $true$ **then**
    $SQ_{LOS}[i] = SQ_{ID}[i]$
  else
    $SQ_{LOS}[i] = 0$
  end if
end for
$Best_{LOS} = SQ_{LOS}[i]$
for $i = 1$ to n **do**
  if $Best_{LOS} < SQ_{LOS}[i]$ **then**
    $Best_{LOS} = SQ_{LOS}[i]$
  end if
end for
if $Best_{LOS} > \tau_{safe\_exp}$ **then**
  Call the Network Manager Module using SQ_{LOS} as set of connections
else
  Call the Network Manager Module using SQ_{ID} as set of connections
end if

Bibliography

Anderson, S., Simmons, R., and Goldberg, D. (2003). Maintaining line of sight communications networks between planetary rovers. In *Proceedings of the Conference on Intelligent Robots and Systems*.

Antonelli, G., Arrichiello, F., Chiaverini, S., and Setola, R. (2005). A self-configuring MANET for coverage area adaptation through kinematic control of a platoon of mobile robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

Arkin, R. and Diaz, J. (2002). Line-of-sight constrained exploration for reactive multiagent robotic teams. In *AMC'02 7th International Workshop on Advanced Motion Control*.

Bailey, T., Nebot, E. M., Rosenblatt, J., and Durrant-White, H. (2000). Data association for mobile robot navigation: A graph theoretic approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2512–2517.

Baker, B. S., Fortune, S., and Grosse, E. (1985). Stable prehension with three fingers. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 114–120, New York, NY, USA. ACM Press.

Balch, T. and Arkin, R. (1998). Behavior-based formation control for multi-robot teams. In *IEEE Transactions on Robotics and Automation*.

Balch, T. and Hybinette, M. (2000). Social potentials for scalable multirobot formations. In *IEEE International Conference on Robotics and Automation (ICRA-2000)*.

Bar-Shalom, Y. and Fortmann, T. E. (1988). *Tracking and Data Association*. Academic Press.

Barshan, B. and Kuc, R. (1990). Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):560–569.

Besl, P. J. and Mckay, N. D. (1992). A method for registration of 3d shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, number 214 in 2, pages 239–256.

Biber, P. (2003). The Normal Distributions Transform: A New Approach to Laser Scan Matching. Technical Report 3, Wilhelm Schickard Institute for Computer Science, Graphical-Interactive Systems (WSI/GRIS), University of Tbingen.

Bosse, M. (2003). *Atlas, A Framework for Scalable Mapping*. PhD thesis, Massachusets Institute of Technology.

Brooks, R. and Flynn, A. M. (1989). Fast, Cheap and Out of Control: A Robot Invasion of the Solar System. *Journal of the British Interplanetary Society*, 42:478–485.

Burgard, W., Cremers, A. B., Fox, D., Hahnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an Interactive Museum Tour-Guide Robot. *Artificial Intelligence*, 114(1-2):3–55.

Burgard, W., Moors, M., and Schneider, F. (2002). Collaborative exploration of unknown environments with teams of mobile robots. In Beetz, M., Hertzberg, J., Ghallab, M., and Pollack, M., editors, *Plan-Based Control of Robotic Agents*, volume 2466 of *Lecture Notes in Computer Science*. Springer Verlag.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2006). Coordinated Multi-robot Exploration. *IEEE Transactions on Robotics and Automation*, 20:120–145.

Cai, Y., Hua, K. A., and Phillips, A. (2005). Leveraging 1-hop neighborhood knowledge for efficient flooding in wireless ad hoc networks. In *Proceedings of the International Performance Computing and Communications Conference*.

Carpin, S. and Parker, L. E. (2002). Cooperative leader following in a distributed multi-robot system. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Chatila, R. and Laumond, J. P. (1985). Position referencing and consistent world modeling for mobile robots. In *In Proceddings of the IEEE International Conference of Robotics and Automation*, pages 138 –145.

Chatila, R. and Moutarlier, R. (1989). Stochastic multisensory data fusion for mobile robot location and environment modelling. In *Proceedings of the International Symposium on Robotics Research*, Tokyo.

Choset, H. and Nagatani, K. (2001). Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization without Explicit Localization. *IEEE Transactions on Robotics and Automation*, 17(2):125 – 137.

Cohen, B., Diu, C., and Laloe, F. (1977). *Quantum Mechanics*, volume 2. John Wiley and Sons.

Cormen, T., Leiserson, C., and Rivest, R. L. (1990). *Introduction to Algorithms*. Cambridge: The MIT Press.

Desai, J. P., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple mobile robots. In *Proccedings of the IEEE International Conference on Robotics and Automation*, pages 2864–2869.

Dias, M. B. and Stentz, A. (2001). A Market Approach to Multirobot Coordination. Technical Report CMU-RI -TR-01-26, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Dias, M. B., Zlot, R. M., Stentz, A., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation*.

Dudenhoeffer, D. D. and Jones, M. P. (2000). A formation behavior for large-scale micro-robot force deployment. In *WSC '00: Proceedings of the 32nd Winter Conference on Simulation*, pages 972–982, San Diego, CA, USA. Society for Computer Simulation International.

Faugeras, O. and Luong, Q. (2001). *The Geometry of Multiple Images: The Laws that Govern the Formation of Multiple Images of a Scene*. The MIT Press.

Fenwick, J. W., Newman, P. M., and Leonard, J. J. (2002). Cooperative concurrent mapping and localization. In *Proceedings of the International Conference on Robotics and Automation*, pages 1810–1817.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395.

Fredslund, J. and Mataric, M. J. (2002a). A general algorithm for robot formations using local sensing and minimal communication. In *Proceedings, 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, pages 100–107, Marina del Rey, CA.

Fredslund, J. and Mataric, M. J. (2002b). Robot formations using only local sensing and control. In *IEEE Transactions on Robotics and Automation*.

Gage, D. (1995). Cost-optimization of many-robot systems. In *Proceedings of SPIE Mobile Robots IX*, pages 260–266.

Gordon, D. F., Spears, W. M., Sokolsky, O., and Lee, I. (1999). Distributed spatial control, global monitoring and steering of mobile agents. In *ICIIS '99: Proceedings of the 1999 International Conference on Information Intelligence and Systems*, page 681, Washington, DC, USA. IEEE Computer Society.

Grabowski, R., Navarro-Serment, L., Paredis, C., and Khosla, P. (1999). Heterogeneous Teams of Modular Robots for Mapping and Exploration. *Autonomous Robots - Special Issue on Heterogeneous Multirobot Systems*.

Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. In *IEEE Transactions on Robotics and Automation*, volume 17, pages 242–257.

Gutmann, J.-S., Weigel, T., and Nebel, B. (2001). A Fast, Accurate and Robust Method for Self-Localization in Polygonal Environments using Laser Range Finders. *Advanced Robotics*, 14(8):651–667.

Haehnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of International Conference on Intelligent Robots and Systems*, volume 1, pages 206– 211.

Howard, A., Mataric, M. J., and Sukhatme, G. S. (2002a). An incremental deployment algorithm for mobile robot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Howard, A., Mataric, M. J., and Sukhatme, G. S. (2002b). Mobile sensor network deployment using potential fields: A distributed scalable solution to the area cov-

erage problem. In *Proceedings of the 6th International Conference on Distributed Autonomous Robotic Systems (DARS02)*.

Insider, T. (2003). Bluetooth wireless networking explained. http://www.thetravelinsider.info/roadwarriorcontent/bluetooth.htm. Date of publication: November 20, 2003.Date last modified: November 18, 2005.

Jantz, S. and Doty, K. (2002). PDA based real-time vision for a small autonomous mobile robot. In *Florida Conference on Recent Advances in Robotics*.

Jennings, J., Donald, B. R., and Rus, D. (1994). Analyzing Teams of Cooperating Mobile Robots. Technical Report TR94-1429, Darmouth College.

Kahn, J. (1997). Wireless infrared communications. *Proceedings of the IEEE*, 85(2):265–298.

Kannan, B., Parker, L. E., Fu, X., and Tang, Y. (2003). Heterogeneous mobile sensor net deployment using robot herding and line-of-sight formations. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*.

Kantor, G. A., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereira, G., and Spletzer, J. (2003). Distributed search and rescue with robot and sensor teams. In *The 4th International Conference on Field and Service Robotics*.

Keenan, J. and Motley, A. (1990). Radio coverage in buildings. *British Telecom Technology*, 8(1):19–24.

Kelly, I. D. and Martinoli, A. (2004). A Scalable, On-Board Localisation and Communication System for Indoor Multi-Robot Experiments. *Sensor Review*, 24(2):167–180. Special issue on Sensor Simulation and Smart Sensors, C. Loughlin, editor.

Kleeman, L. and Kuc, R. (1995). Mobile Robot Sonar for Target Localization and Classification. *Internationall Journal of Robotics Research*, 14(4):295–318.

Knight, J., Davidson, A., and Reid, I. (2002). Towards constant time SLAM using postponement. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 405–413.

Konolige, K., Fox, D., Ortiz, C., Agno, A., Eriksen, M., Limketkai, B., J. Ko, J. M., Schulz, D., Stewart, B., and Vincent, R. (2004). Centibots: Very large scale dis-

tributed robotic teams. In *Proceedings of the International Symposium on Experimental Robotics (ISER-04)*.

Kornienko, S., Kornienko, O., and Levi, P. (2005). *Minimalistic Approach towards Communication and Perception in Microrobotic Swarms*. In *Proceedings of IROS 2005*, pages 1–2. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Edmonton, Canada: Springer-Verlag.

Lam, L., Lee, S.-W., and Suen, C. Y. (1992). Thinning Methodologies-A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885.

Latimer, D., Srinivasa, S., Shue, V. L., Sonne, S., Choset, H., and Hurst, A. (2002). Towards sensor based coverage with robot teams. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA '02)*. IEEE.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Press.

Lee, D., Chung, W., and Kim, M. (2002). Probabilistic localization of the service robot by map matching algorithm. In *Proceedings of the International Conference on Control, Automation and Systems*, pages 1667–1627.

Leonard, J., Rikoski, R., Newman, P., and Bosse, M. (2002). Mapping Partially Observable Features from Multiple Uncertain Vantage Points. *International Journal of Robotics Research*, 21(10):943–975.

LoPresti, E., Simpson, R., Miller, D., and Nourbakhsh, I. (2002). Evaluation of sensors for a smart wheelchair. In *Proceedings of the RESNA (Rehabilitation Engineering and Assistive Technology Society of North America) Annual Conference*.

Lu, F. and Milios, E. (1997). Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *Journal of Intelligent Robotics Systems*, 18(3):249–275.

Madhavan, R., Fregene, K., and Parker, L. E. (2004). Distributed Cooperative Outdoor Multirobot Localization and Mapping. *Journal of Autonomous Robots*, 17(1):23–39.

Mataric, M. J. (1998). Using Communication to Reduce Locality in Distributed Multiagent Learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357–369.

MaxStream (2003). *Indoor Path Loss: Application Note*. MaxStream.

McLurkin, J. and Smiths, J. (2004). Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *in 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*.

Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42.

Min, T. and Min, H. K. (1998). A decentralized approach for cooperative sweeping by multiple robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 1, pages 380 – 385.

Molkdar, D. (1991). Review on radio propagation into and within buildings. In *Microwaves, Antennas and Propagation*, volume 138, pages 61–73. IEE.

Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastSLAM. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 1985–1991.

Moravec, H. and Elfes, A. E. (1985). High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121.

Muoz-Gomez, L., Alencastre-Miranda, M., and Swain-Oropeza, R. (2004). A multirobot mapping approach using different motion planning methods. In *The 10th IASTED International Conference on Robotics and Applications*. IASTED.

Navarro-Serment, L., Grabowski, R., Paredis, C., and Khosla, P. (2002). Millibots. *IEEE Robotics and Automation Magazine*, pages 31 – 40.

Neira, J. and Tardos, J. J. (2001). Data Association in Stochastic Mapping using the Joint Compatibility Test. *Transactions of Robotics and Automation*, 17(6):890–897.

Newman, P. (1999). *On the Structure and Solution of the Simultaneous Localization and Mapping Problem*. PhD thesis, University of Sydney, Australia.

Newman, P., Fenwick, J., and Leonard., J. (2002a). Cooperative concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1810–1817, Washington, USA.

Newman, P., J.Leonard, Tardós, J., and J.Neira (2002b). Explore and return: Experimental validation of real time concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1802 –1809. IEEE.

Nguyen, H., Pezeshkian, N., Gupta, A., and Farrington, N. (2004). Maintaining communication link for a robot operating in a hazardous environment. In *Proceedings of the 10th International Conference on Robotics and Remote Systems for Hazardous Environments*.

of multiple mobile robots via communication, C. (1998). H. hu and i. d. kelly and d. a. keating and d. vinagre. In *Proceedings of SPIE. Mobile Robots XIII and Intelligent Transportation Systems*, pages 94–103, Boston, Masachusetts.

Parker, L. (2000). Current state of the art in distributed autonomous mobile robotics. In *Distributed Autonomous Robotic Systems*, pages 3–12.

Perkins, C. E. and Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, NY, USA. ACM Press.

Perkins, C. E. and Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA. IEEE Computer Society.

Pezeshkian, N., Raymond, M., Gupta, A., Nguyen, H., and Spector, J. (2003). Autonomous communication relays for tactical robots. In *proceedings of the 11th International Conference on Advanced Robotics*.

Pimentel, S. and Montenegro, M. (2002). Multi-robot exploration with limited-range communication. In *congreso brasileiro de automatica*.

Powers, M. and Balch, T. (2004). Value-based communication preservation for mobile robots. In *International Conference on Robotics and Automation*.

RadioMetrix (2004). Spaceport modem. http://www.radiometrix.co.uk/products/spm2.htm. Date of publication: November, 2004.

Rappaport, T. S. (2001). *Wireless Communication: Principles and Practice*. Prentice Hall, 2nd edition.

Reif, J. and Wang, H. (1995). Social potential fields: A distributed behavioral control for autonomous robots. In *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459.

Rekleitis, I., Dudek, G., and Milios, E. (2000). Multi-robot collaboration for robust exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'00*, volume 4, pages 3164 –3169.

Rekleitis, I., Lee-Shue, V., New, A. P., and Choset, H. (2004). Limited communication, multi-robot team based coverage. In *IEEE International Conference on Robotics and Automation*, pages 3462–3468, New Orleans, LA.

Rimon, E. (1990). *Exact Robot Navigation using Artificial Potential Functions*. PhD thesis, Yale University. Adviser-Daniel E. Koditschek.

Robinson, C. L., Block, D., Brennan, S., Bullo, F., and Cortes, J. (2004). Nonsmooth analysis and sonar-based implementation of distributed coordination algorithms. In *International Conference on Robotics and Automation*, pages 3000–3005, New Orleans, LA.

Rodriguez-Losada, D. (2004). *SLAM Geometrico en Tiempo Real para Robots Moviles en Interiores Basado en EKF*. PhD thesis, Universidad Politecnica de Madrid.

Rodriguez-Losada, D. and Matia, F. (2003). Integrating segments and edges in feature-based SLAM. In *Proceedings of the IEEE 11th International Conference on Advanced Robotics*. IEEE.

Rodriguez-Losada, D. and Matia, F. (2004). Local maps fusion for real time multirobot indoor simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation*, pages 1308–1313.

Royer, E. and Toh, C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2).

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*.

Scott, D. J. and Yasinsac, A. (2004). Dynamic probabilistic retransmission in ad hoc networks. In *International Conference on Wireless Networks*, pages 158–164.

Seidel, S. Y. and Rappaport, T. S. (1992a). 914 mhz path loss prediction models for indoor wireless communication in multifloored buildings. *IEEE Transactions on Antennas and Propagation*, 40(2):207–217.

Seidel, S. Y. and Rappaport, T. S. (1992b). A ray tracing technique to predict path loss and delay spread inside buildings. In *IEEE GLOBECOM*, pages 1825–1829.

Shatkay, H. and Kaelbling, L. P. (1997). Learning topological maps with weak local odometric information. In *International Joint Conference on Artificial Intelligence*, pages 920–929.

Shucker, B. and Bennett, J. K. (2005). Virtual Spring Mesh Algorithms for Control of Distributed Robotic Macrosensors. Technical report, University of Colorado Department of Computer Science.

Sibley, G., Rahimi, M., and Sukhatme, G. (2002). Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1143 –1148.

Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M.and Thrun, S., and H., Y. (2000). Coordination for multi-robot exploration and mapping. In *Procceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 852 –858.

Singh, K. and Fujimura, K. (1993). Map making by cooperating mobile robots. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, volume 2, pages 254 –259.

Smith, R., Self, M., and Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Fourth International Symposium on Robotics Research*, pages 467–474, Cambridge, MA, USA. MIT Press.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating Uncertain Spatial Relationships in Robotics. *Autonomous Robot Vehicles*, pages 167–193.

Solanas, A. and Garcia, M. A. (2004). Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

Spears, W. M. and Gordon, D. F. (1999). Using artificial physics to control agents. In *ICIIS '99: Proceedings of the 1999 International Conference on Information Intelligence and Systems*, page 281, Washington, DC, USA. IEEE Computer Society.

Sweeney, J. D., Brunette, T., Yang, Y., and Grupen, R. A. (2002). Coordinated teams of reactive mobile platforms. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Tardos, J. and Castellanos, A. (1999). *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Kluwer Academic Publishers, Boston.

Tardos, J., Neira, J., Newman, P., and Leonard, J. (2002a). Robust Mapping and Localization in Indoor Environments Using Sonar Data. *International Journal of Robotics Research*.

Tardos, J., Neira, J., Newman, P., and Leonard, J. (2002b). Triangulation-Based Fusion of Sonar Data with Application in Robot Pose Tracking. *The International Journal of Robotics Research*, 21(4):311–330.

Thibodeau, B., Fagg, A., and Levine, B. (2004). Signal Strength Coordination for Cooperative Mapping. Technical Report 04-64, University of Massachusetts Amherst.

Thrun, S. (2000). Probabilistic Algorithms in Robotics. Technical Report CMU-CS-00-126, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Thrun, S. (2001). An Online Mapping Algortithm for Teams of Mobile Robots. *International Journal of Robotics Research*, 20(5):335–363.

Thrun, S. (2002). Robotic Mapping: A Survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millenium*, pages 1 – 35. Morgan Kaufmann.

Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schimdt, T. (1998a). Map Learning and High-Speed Navigation in RHINO. *Artificial Intelligence and Mobile Robots*, pages 21–52.

Thrun, S., Burgard, W., and Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics & Automation*.

Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., and Kuipers, B. (1998b). Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Thrun, S. and Martin, C. (2002). Real-time acquisition of compact volumetric 3D maps with mobile robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 311–316.

Ulam, P. and Arkin, R. (2004). When good communication go bad: communications recovery for multi-robot teams. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 3727– 3734.

Vazquez, J. and Malcolm, C. (2004). Distributed multirobot exploration maintaining a mobile network. In *Proc. of the 2nd International Conference on Intelligent Systems*, volume 3, pages 113–118.

Verma, A., Everett, H., Manouk, N., and H.G.Nguyen (2002). Autonomous mobile communication relays. In *Proceedings of the SPIE Conference on Unmanned Ground Vehicle Technology*.

Volpe, R. and Khosla, P. (1990). Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments. *IEEE Transactions on Systems, Man, and Cybernetics. Special Issue on Unmanned Vehicles and Intelligent Systems*, 20(6).

Wagner, A. and Arkin, R. (2003). Internalized plans for communication-sensitive robot team behaviors. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 3, pages 2480 – 2487.

Wang, P. S. P. and Zhang, Y. Y. (1989). A Fast and Flexible Thinning Algorithm. *IEEE Trans. Comput.*, 38(5):741–745.

Weiss, G. and von Puttkamer, E. (1995). A map based on laserscans without geometric interpretation. In *Proceedings of Intelligent Autonomous Systems*, volume 4, pages 403–407.

Wijk, O. and Christensen, H. (2000). Triangulation-Based Fusion of Sonar Data with Application in Robot Pose Tracking. *IEEE Transactions on Robotics and Automation*, 16(6):740–752.

Williams, D. (2003). *PDA Robotics: JAMECO*. McGraw-Hill, 1 edition.

Williams, S. (2001). *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, University of Sydney.

Williams, S., Dissanayake, G., and Durrant-Whyte, H. (2002). An efficient approach to the simultaneous localization and mapping problem. In *Proceedings of the IEEE International Conference on Automation and Robotics*, volume 15, pages 406–411, Washington.

Winfield, A. F. (2000). Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In *Distributed Autonomous Robotic Systems*, volume 4, pages 273–282.

Yamaguchi, H. (1997). Adaptive formation control for distributed autonomous mobile robot groups. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2300 –2305.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM Press.

Yamauchi, B., Schultz, A. C., and Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3715–3720.

List of Notation

Notation for the BERODE architecture

n	Number of robots
k	Size of the local network
ε	Set of connections of the robot network
κ	Set of control connections of the robot network
κ_i	Set of control connections for robot i
ϕ_i	Set of communication constraints for robot i
$\varepsilon_{i,j}$	Connection status between robots i and j
$\kappa_{i,j}$	Signal quality between robots i and j
L_i	Level of safety for robot i
$w_{i,j}$	Weight of the edge between robot i and j for the calculation of the MST control network
$D_{i,j}$	Discomfort distance between robots i and j
σ_A	Attractive threshold for the *virtual spring* model
σ_R	Repulsive threshold for the *virtual spring* model
σ_{safe}	Safe threshold for the safety level
σ_{prec}	Precautionary threshold for the safety level
$k_{compresion}$	Stiffness of a compressed spring
$k_{stretching}$	Stiffness of a stretched spring
$SQ_{predicted}$	Signal quality predicted
$\sigma_{inc}(t, \sigma_{start}, \sigma_{end}, t_{start}, t_{end})$	Increasing interpolation function to obtain a value for the time t in the time range $[t_{start}, t_{end}]$ with range values $[\sigma_{start}, \sigma_{end}]$
$\sigma_{dec}(t, \sigma_{start}, \sigma_{end}, t_{start}, t_{end})$	Decreasing interpolation function to obtain a value for the time t in the time range $[t_{start}, t_{end}]$ with range values $[\sigma_{start}, \sigma_{end}]$
$Z(t)$	State of the robot at time t
t_{exp}	Expiration time associated to the Pusher role
t_{pusher}	Time that a robot remains on the Pusher role
$t_{network}$	Time that an Explorer robot waits for signal quality improvement

t_{hold}	Time that a robot remains stationary
$E(x,y)$	Energy for a sampled position (x, y)
$U(x,y)$	Potential energy for a sampled position (x, y)
$Y(K)$	Yukawa potential function for a distance K
SQ_i	Predicted signal quality for the i^{th} robot
S_L	Update time for transmissions at the local level
S_G	Update time for transmission at the global level
$N(\mu,\sigma)$	Gaussian distribution with median μ and variance σ

Notation for the Extended Kalman Filter

N	Number of features in the feature map of the robot
$\hat{x}$	State vector for the Extended Kalman Filter
C	Covariance matrix for the state vector $\hat{x}$
C_{ij}	Cross covariance between the features i and j
$\hat{x}(t\,\|t-1)$	Predicted state at time t for $\hat{x}$ given the state at time t-1
$\hat{u}(t)$	Control input at time t
$\hat{x}_r$	Position vector for the robot
$\hat{x}_m$	Position vector of the last m sensing positions of the robot
$\hat{x}_{f_i}$	Position vector for the feature at position i on the state vector
z_i	Observation of feature at position i on the state vector
h_i	Observation model for the feature i
$g(\hat{x},\hat{z})$	Function to transform a relative measurement $\hat{z}$ taken at position $\hat{x}$ to the global reference frame
$h_{f_i}(\hat{x}_r,\hat{x}_{f_i})$	Function to transform the position of the feature $\hat{x}_{f_i}$ with respect to the position of feature $\hat{x}_r$
$P(x,y)$	Point feature with Cartesian coordinates (x, y)
$L(\rho,\phi)$	Line feature with Polar coordinates (ρ,ϕ)