Collaborative Decision Making in Uncertain Environments

by Joseph Lee Baxter, BSc

Thesis submitted to The University of Nottingham,
School of Computer Science,
for the degree of Doctor of Philosophy,
May 2009

Abstract

Two major issues in the design of multi-robot systems are those of communication and co-ordination. Communication within real world environments cannot always be guaranteed. A multi-robot system must, therefore, be able to continue with its task in the absence of communication between team members. Co-ordination of multiple robots to perform a specific task involves team members being able to make decisions as a single entity and as a member of a team. The co-ordination needs to be robust enough to handle failures within the system and unknown phenomena within the environment.

In this thesis, the problems of communication and co-ordination are discussed and a new type of multi-robot system is introduced in an effort to solve the inherent difficulties within communication and co-ordination of multi-robot systems.

The co-ordination and communication strategy is based upon the concept of sharing potential field information within dynamic local groups. Each member of the multi-robot system creates their own potential field based upon individual sensor readings. Team members that are dynamically assigned to local groups share their individual potential fields, in order to create a combined potential field which reduces the effect of sensor noise. It is because of this, that team members are able to make better decisions.

A number of experiments, both in simulation and in laboratory environments, are presented. These experiments compare the performance of the system against a non-sharing control and a hybrid system made up of a global path planner and a reactive motor controller. It is demonstrated that the new system significantly outperforms these other methods in a search type problem.

From this, it is concluded that the novel system proposed in this thesis successfully tackled the search problem, and that it should also be possible for the system to be applied to a number of other common multi-robot problems.

Acknowledgements

I would like to take this opportunity to thank my academic supervisors, Dr. Jonathan Garibaldi and Prof. Edmund Burke, for their support and guidance throughout my study. I would also like to thank my industrial supervisor, Dr. Mark Norman from Merlin Systems, for the opportunity to use the Miabot Pro. Special thanks to Patrick Peglar and Simon Norman at Merlin Systems, for their technical support. I would also like to express my thanks to Phil Birkin from the University of Nottingham for his input throughout my research. This work was supported by an EPSRC industrial CASE award, made available through the Knowledge Training Network (KTN), and so many thanks go to Melvin Brown and all those at the Smith Institute.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Multi-robot systems have been an established research area for more than a decade. Even so, compared to its parent research topic of single-robot systems, it is still a relatively young area of interest.

Typical problems tackled by single-robot systems include Urban Search and Rescue (USAR) scenarios. See figure 1.1a[1] for an example of a single-robot system being deployed in the aftermath of the World Trade Centre disaster. These systems are generally tele-operated, owing to the urgency of the task. In rescue operations, the time taken for a robot to rescue itself from local minima could cost lives.

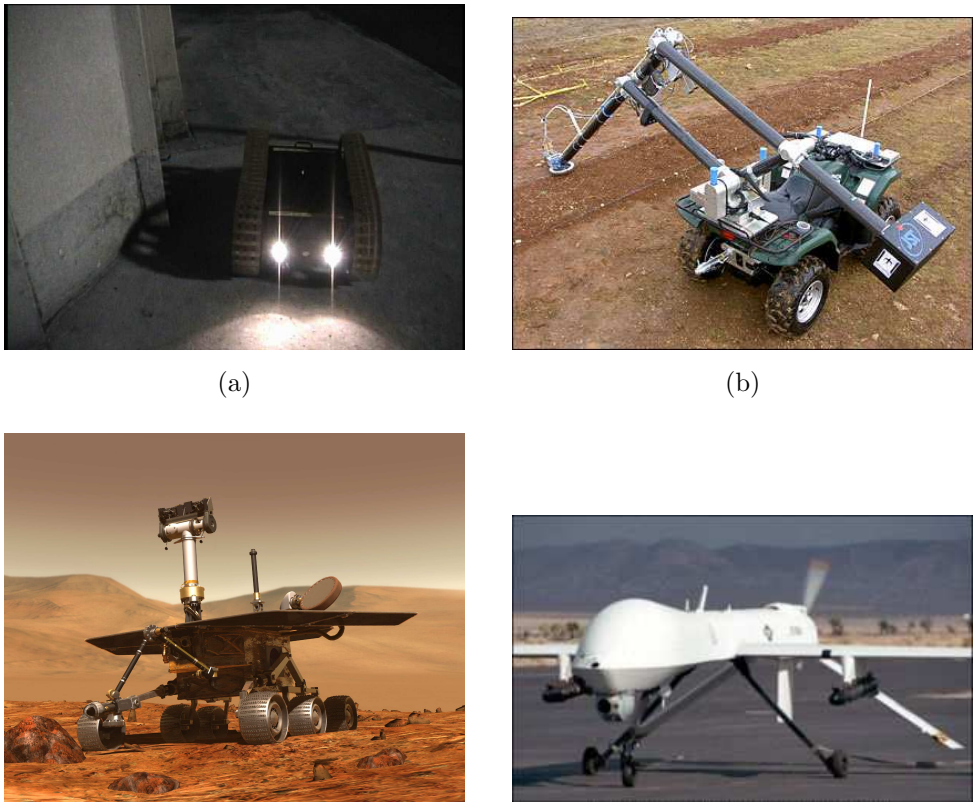
Mine-sweeping is another typical single-robot system task — see figure 1.1b[2] for an example of a mine sweeper robot. Most systems are remote controlled until they reach the mine field. Once at the mine field, the system is fully autonomous. This is usually due to the fact that the system's definition of the world is limited to a mine field. That is, the system has no concept of being outside a mine field. If placed outside a mine field, the system would still attempt to find mines until told to stop by a human operator.

The Mars "Spirit" rover, see figure 1.1c[3], has been on the surface of Mars for just over four years, and is one of the best examples of a robotic system working in

[1]Image from http://crasar.csee.usf.edu
[2]Image from http://www.bbc.co.uk
[3]Image from http://www.nasa.gov

an uncertain environment without the possibility of human intervention. During the relatively small periods of communication between a controller on Earth and a rover on Mars, the rover is given a list of objectives to complete before the next scheduled communiqué. These objectives are carried out completely autonomously.

Unmanned Aerial Vehicles (UAVs) are used by numerous armed forces throughout the world to perform reconnaissance duties. An example of a remote controlled Predator UAV is shown in figure 1.1d[4]. During July 2005 to June 2006, the Predator UAV flew over thirty three thousand flying hours during the Iraq conflict [33]. Recently a pair of UAVs flew across parts of Antarctica, demonstrating the system's robustness to adverse weather conditions.



(a) (b)

(c) (d)

Figure 1.1: (a) Robotic Urban Search and Rescue. (b) Robotic Mine Sweeper. (c) Mars Exploration Rover. (d) Predator Unmanned Aerial Vehicle.

[4]Image from http://www.wikipedia.com

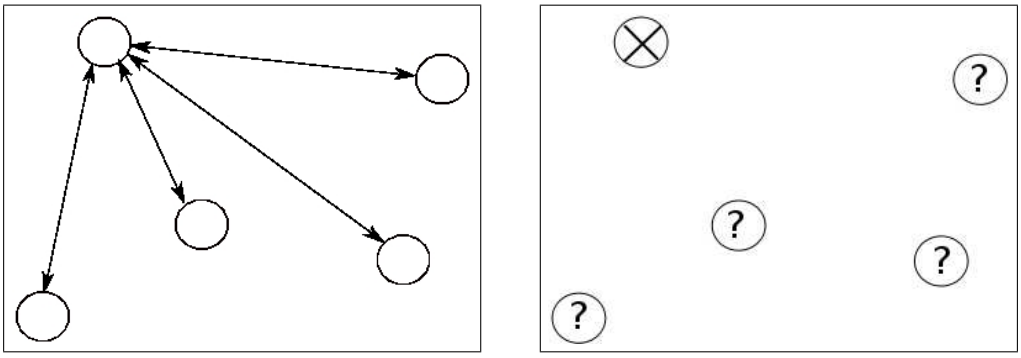
A central issue motivating this thesis is the question of how deploying multi-robot systems to these scenarios could improve performance. The use of a multi-robot system for a USAR task, for example, could reduce the time needed to search a given area. As multi-robot systems lend themselves to the use of smaller robots enabling areas of a site previously inaccessible to become searchable.

Mine-sweeping could be more efficiently solved by a multi-robot system. Whereas a single-robot system is limited to detecting one mine at a time, a multi-robot system, by its very nature, can detect multiple mines.

A related issue is how a multi-robot system could perform better than what is obviously a very robust single-robot system in the Mars rovers. The main area for improvement is undoubtedly, numbers. Only two Mars exploration rovers are currently deployed on the entire surface of Mars ($144,798,500km^2$). There have been attempts to put more robotic systems on Mars, but the difficulty of such a task is huge. For example, the loss of the Beagle 2 Mars robot is presumed to be due to the failure of landing parachutes deployed during landing, perhaps due to a thinner than expected atmosphere. By sending a multi-robot system containing hundreds (perhaps thousands) of robots, the probability of total mission failure due to crash landings may be reduced dramatically and, in theory, a larger area may be explored in a shorter space of time.

In a report on the Iraqi conflict [22], the US Air-force lost 53 out of 139 Predator UAVs (38%) at a cost of $4.5 million each. UAVs undoubtedly provide effective surveillance, but the cost of replacing them during conflict can become astronomical. Developing cheaper, smaller UAVs that provide surveillance of a given area in teams could be one way of tackling the problem of spiralling costs.

Multi-robot systems bring their own problems of course: namely reliable, efficient and effective communication and co-ordination. These problems can be generalised to a debate between the adoption of either a centralised or distributed system. In a centralised system all members of the multi-robot system are co-ordinated through a single centralised controller (a remote PC or a “leader” robot). See figure 1.2a for an example of a leader robot co-ordinating each member of the group. Solutions are usually found at the global level and are near-optimal. That is, during a multi-

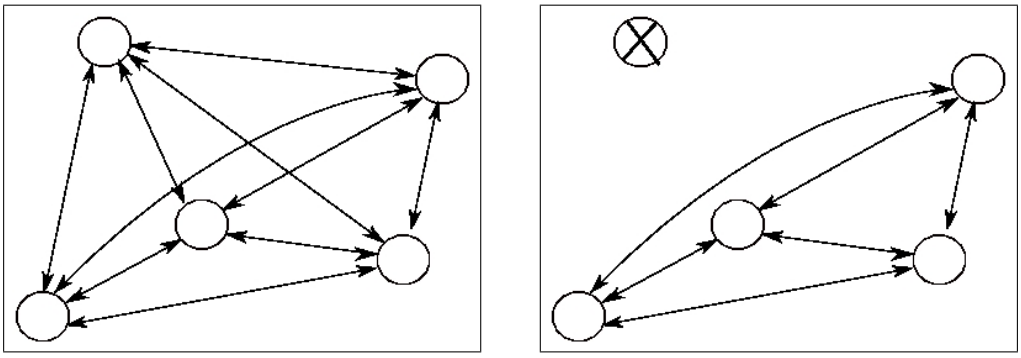


(a) (b)

Figure 1.2: (a) Centralised system — All decisions are made by a remote PC or "leader" robot. (b) The "leader" robot fails, causing a system wide failure. Robots/agents are represented by circles. Communication is shown by arrows.

robot exploration task, the central controller would plan collision free paths for each robot within the system, attempting to maximise the total area covered within the environment. The more robots within the group, the more computationally expensive co-ordination becomes. As each member of the system is given specific tasks to perform, the communications load between members is reduced, but this also imposes several points of failure within the system — most notably the central controller. However, it is also the case that if a member given a specific task fails, this has an accumulative effect on the rest of the system's performance. In figure 1.2b the leader robot suffers a failure and is no longer able to co-ordinate the rest of the group. As the other group members have no local decision making capabilities they will not be able to complete the task.

In a distributed system, each member controls its own actions and co-ordination is achieved through communication between members. Figure 1.3a provides an example of a distributed system; each individual robot is capable of making decisions, and interactions with other group members are used to improve that decision making process. As a consequence, if one member of the system has a failure, the rest of the system can continue with the task. The more homogeneous the members of the team, the more robust to failure the system will become. See figure 1.3b for an example of a robot failure within a distributed system. As communications are not



(a) (b)

Figure 1.3: (a) Distributed system — Decisions are made by individuals. (b) One of the robots fails, causing minimal disruption to the system. Robots/agents are represented by circles. Communication is shown by arrows.

centralised and each robot is capable of making its own decisions, the effect of the failure on the overall system's performance is minimal. However, the communications load of the system is increased. In distributed systems, solutions are usually solved at the local level, with the global solution being solved as an emergent property of the system. However, these solutions are usually sub-optimal. This is due to the fact that, unlike a centralised system where a single controller agent has access to all available information, in a distributed system each member has access to incomplete information and hence individual robots cannot possibly achieve optimal solutions. Solutions are said to be an emergent property of the system, as through completing simple tasks it is possible for the system to complete a more complex task. For example, a group of robots that have the simple task of collision avoidance, may also solve the coverage task as an emergent property.

The task of the multi-robot system designer is to balance the need for near-optimal solutions with robustness and communication limitations.

1.2 Goals of this Thesis

The overall aim of this thesis is to design and implement a novel multi-robot system that is capable of performing a given task in an unknown environment. The system

was designed to encapsulate the following beneficial aspects for a multi-robot system: robustness, scalability and fault tolerance, with the specific goals being as follows:

1. **To design and implement a new type of multi-robot system that is weakly co-ordinated at the local level, but unaware at the global level.**

   The co-ordination of an entire multi-robot system can be very expensive in terms of communications bandwidth. The co-ordination of the entire system is also not always give an increase in performance. For example, to robots in completely different parts of environment for example would not necessarily need to co-ordinate there tasks. As such the system introduced in this thesis only co-ordinates robots within at the local level (the robots are within an predefined distance of one another), robots not within these local groups do not co-ordinate with the robots within them.

2. **To design and implement a communication/co-ordination method that is inexpensive in terms of computation and bandwidth.**

   The effectiveness of a multi-robot system is closely related to the effectiveness of its communication/co-ordination strategy. A system that relies upon communication strategy that is slow will result in a system with a slow response to communications. Similarly a system that relies on an over complex co-ordination strategy will result in a system that is slow to respond to new stimuli.

3. **To design and implement a multi-robot system that is not reliant upon information gathered *a priori*.**

   The overall aim of the project was to implement the system in an unknown environment. Therefore, it was not possible to provide the system with knowledge gathered *a priori*. This restriction is common among many real world scenarios, such as USAR.

4. **To design and implement a multi-robot system that is not reliant upon explicit information gathered from other robots.**

It is important that the individual robots have the ability to make their own decisions, even if the robots are a part of a team. This enables the system to become fault tolerant in regard to the loss of robots. The robustness of the system is also improved as individual robots can react to changes in the environment not yet detected by other robots within the system.

1.3 Research Methodology

Conducting experiments on real robots was the driving methodology of this thesis. By developing a new multi-robot system with aim of deploying that system on real robots. It was possible develop a system not restricted by artificial constraints, but limited by physical constraints. For example, the system was designed with the limitations and nuances of sensors in mind, rather than designing the system based upon generic models of sensors. However, simulations do have their place in developing robotic systems i.e. fast prototyping. These simulations are useful for gaining knowledge on how a system will react in a specific environment with specific conditions. The knowledge gained from these simulations can then be used as a starting point for developing a system to be deployed in real robotic systems, which are designed to work in numerous environments under numerous conditions.

As such, the low fidelity simulator Stage is used to prototype the system, which is eventually delpoyed in the Miabot class of robot. See chapter 3 for more detail on the simulator and the real robotic hardware.

1.4 Thesis Contributions

The three major contributions of this thesis are as follows:

1. A new type of multi-robot system, which performs no co-ordination or communication at the global level, but is weakly co-ordinated at the local level, is introduced.

2. A method of sharing information through fusing sensory information into a

potential field will be shown to be an effective method of communication and co-ordination in a search type task.

3. It will be shown that taking a more pessimistic view in terms of sensor belief is advantageous in cluttered environments, whilst performing a search type task. It will also be shown that taking a more optimistic view is advantageous in sparse environments, whilst performing a search type task.

1.5 Dissemination

The following articles have been published as a result of the research presented in this thesis.

1.5.1 Book Chapter

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "Multi-Robot Search and Rescue: A Potential Field Based Approach", in *Autonomous Robots and Agents*, series: Studies in Computational Intelligence book series, Vol. 76, Mukhopadhyay, Subhas; Sen Gupta, Gourab (Eds.), Springer-Verlag, pp 9-16, 2007.

1.5.2 Refereed Conference Papers

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "Statistical Analysis in MiroSot", *In the proceedings of the FIRA World Congress 2005*, Singapore, 14th-16th December, CD-ONLY, 2005.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "The Effect of Potential Field Sharing in Multi-Agent Systems", *In the proceedings of 3rd International Conference on Autonomous Robots and Agents (ICARA 2006)*, Palmerston North, New Zealand, 12th-14th December, pp 33-38, 2006.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "Real-world Evaluation of a Novel Potential field sharing method", *In the 5th International Confer-*

ence on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008), Linz, Austria, 19th-21st June, pp 73-78, 2008.

1.5.3 Refereed Journal Papers

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman,"Shared Potential Fields and their Place in a Multi-Robot Coordination Taxonomy", *Robotics and Autonomous System*, **to be published**, 2009.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi, S. Groenemeyer & M. Norman, "Multi-robot Co-ordination Using Shared Potential Fields", **submitted to** *IEEE Transactions on Robotics*, 2009.

1.5.4 Presentations

The following talks have focused on the work presented in this thesis.

- J.L. Baxter, "The Effect of Potential Field Sharing in Multi-Agent Systems", in *Automated Scheduling Optimisation and Planning Research Group seminar*, University of Nottingham, Nottingham, 30th November, 2006.

- J.L. Baxter, "The Effect of Potential Field Sharing on Real Robots", in *Automated Scheduling Optimisation and Planning Research Group seminar*, University of Nottingham, Nottingham, 1st November, 2007.

- J.L. Baxter, "Collaborative Decision Making in Uncertain Environments", in *Knowledge Transfer Network Presentations, Smith Institute Alan Taylor Day*, St Catherine's College, Oxford, November 26th, 2007.

1.6 Overview of the Thesis

The remainder of this thesis is split into seven chapters. Chapter 2 defines a number of common robotic problems. It then goes on to give a detailed literature review of robotic architectures and multi-robot systems. Several key architectures will be

discussed and critically analysed, falling into the regions of reactive, deliberative and hybrid systems. Farinelli's multi-robot taxonomy will be described and examples given for each group, along with a critical analysis. The potential field method used in numerous robotic systems will also be explained through a number of examples, and critically analysed. Finally, robotic systems that are closely related to the potential field sharing method described in chapter 4 will be discussed in detail.

Chapter 3 describes the robotic hardware and software architecture used throughout the experimentation within this thesis. Detailed configurations and specifications are given. Limitations of the hardware and software deployed are also discussed. Finally, a discussion on the current architecture of the system and the proposed future architecture of the system ends the chapter.

Chapter 4 describes the new multi-robot system. The system will be broken down into its sub-processes, with each being described in detail. A comparison with the traditional potential field method, in regard to susceptibility to the known limitations, will also be presented.

Chapters 5, 6 and 7 discuss the experiments undertaken in simulation and in the robot laboratory. A series of simulation experiments were carried out, a one target search problem and a two target search problem, over three different environments, with groups of robots ranging from two to eight. A series of laboratory experiments were also carried out. First, the one target experiment was repeated in the laboratory setting. Then, an experiment comparing a hybrid system against the novel reactive system was conducted. A statistical analysis of each experiment is given, demonstrating that the new system performs better than both a non-sharing control and a hybrid system.

Chapter 8 concludes the thesis, with a discussion on the major contibutions of the thesis as well as a discusion on which goals were achieved. Finally the limitations of the system proposed and possible future directions of research are discussed.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, a number of common robotic problems tackled by the robotic systems will be defined. The three major implementations of robotic architectures (reactive, deliberative and hybrid) will be discussed along with Farinelli's taxonomy of multi-robot systems. As the system presented in this thesis is loosely based upon potential field theory, the common approaches to the potential field method in robotics will be described. A detailed discussion on the literature most closely related to the system described in this thesis is also presented. A summary will conclude this chapter.

2.2 Problem Definitions

In this section, definitions for several common robotic problems that are tackled by single and multi-robot systems are given. References to literature are given and, where appropiate, further explanation is given in the relevant sections (sections 2.3, 2.5 and 2.6).

2.2.1 Path Planning

The oldest problem in mobile robotics is that of path planning (tackled in the literature by Leroy *et al.* [56], Chand *et al.* [71] and Parsons *et al.* [75]. It can be defined as follows:

> Calculating the necessary motor operations needed by a robot (or group of robots) to travel from a start location to a target location, whilst avoiding collisions with obstacles within the environment.

The complexity of the problem varies depending on the density of objects within the environment; the inclusion of dynamic objects within the environment; the amount of environment information given *a priori*; and the sensory information made available to the path planner. The common method for path planning is to calculate a near optimum solution *a priori* from the available environmental information. As the robot moves along this path it is adjusted to avoid any unforeseen events such as dynamic obstacles. An example of the path planning problem is given in figure 2.1. Three possible paths have been calculated — the robot will attempt to follow the path with the smallest total weight. Each arc between way-points is given a weight in relation to its length, distance from obstacles and type of terrain. Each arc's weight in a path is summed.

The path planning problem is commonly the base problem of the more high level problems described in this section.

2.2.2 Coverage

The coverage problem (tackled by Batalin *et al.* [9], Gazi *et al.* [44] and Howard *et al.* [49]) can be defined as follows:

> The task of controlling a robot (or group of robots) to maximise the amount of sensory information gathered within an environment. This can be in the form of traversing the environment, whilst guaranteeing total area coverage. Another approach is to form a sensor net of the environment in an effort to maximise the area covered by the net.

The complexity of the problem varies depending upon the number of robots deployed, the size of the environment and the type of sensors used. The typical solution to the first kind of coverage problem is to assign way-points in different sections of the environment (assuming a map is given *a priori*). An example is given in figure 2.2. A robot attempts to traverse the environment, visiting each way-point along its route. A typical solution to the second kind of coverage problem (sensor net) is to

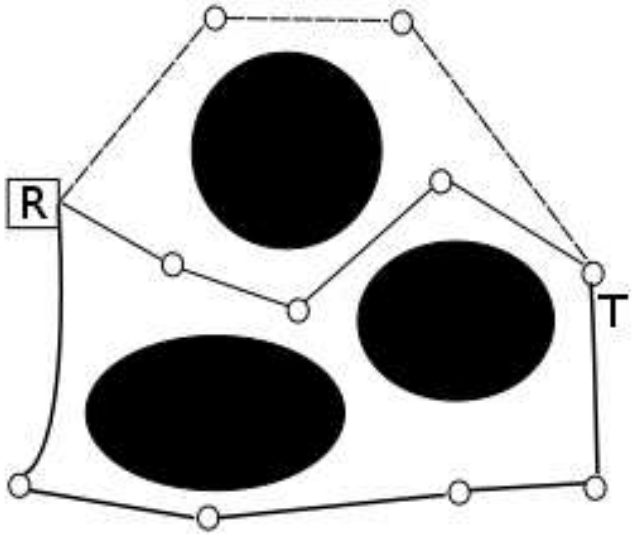


Figure 2.1: A robot (R) calculates a number of possible paths to the desired target location (T). Each path (dashed line, solid line and bold line) is made up of a number of way-points (white circles) the cost of traversing between these way-points is related to the distance, difficulty of terrain and density of objects (black circles) present.

make members of the robotic team actively avoid one another, thus forcing the robots to explore more of the environment. In figure 2.3, a group of four robots attempt to create a sensor net within the environment.

A sub-type of the coverage problem is shown in Figure 2.2, the surveillance problem (tackled by Martins-Filho *et al.* [59] and Oates *et al.* [69] in the literature). The robot is equipped with a sensor that can detect anomalies within the environment. The surveillance problem can be defined as:

> A robot (or group of robots) has the task of traversing an (attempting to maximise coverage of) environment, whilst detecting anomalous sensor readings from the environment.

Another sub-type of the coverage problem is the graze problem (tackled by Balch *et al.* [6] in the literature), which be defined as:

> A robot (or group of robots) has the task of traversing an (attempting to maximise coverage of) environment, whilst performing 'work' at each location it visits.

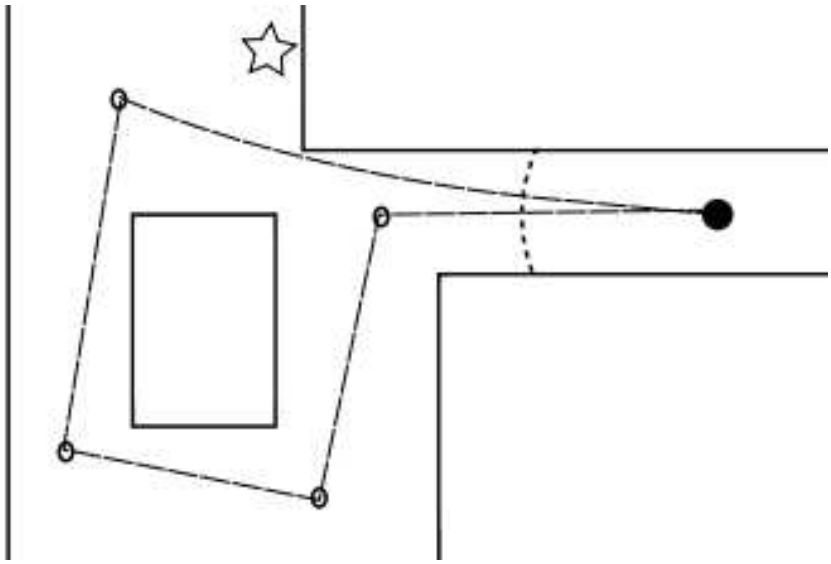


Figure 2.2: A robot (black circle) traverses an environment attempting to maximise the total area covered. The dashed line is the desired path, the circles are the way-points. The dashed arc is the sensory limit of the robot. The star is an anomaly within the environment.

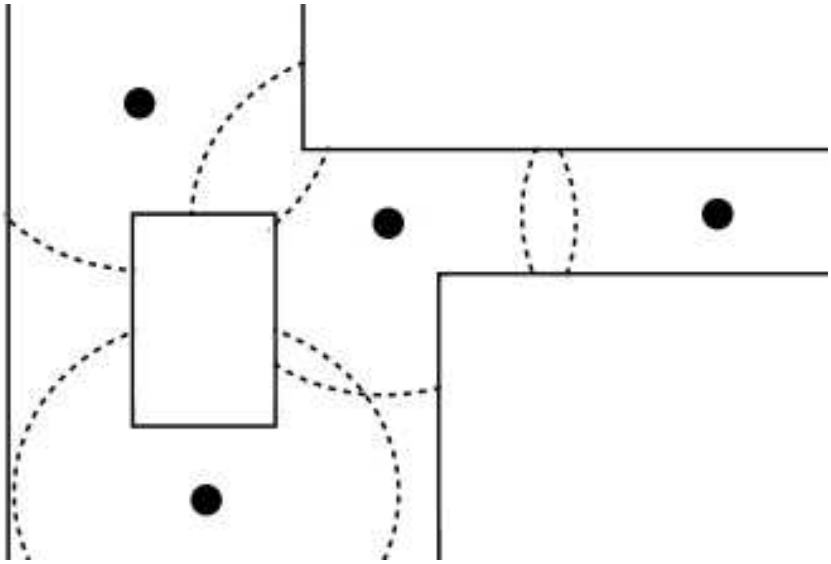


Figure 2.3: A group of robots (black circles) form a sensor net to get the optimal coverage area for the given amount of robots. The dashed circles are the sensory limits of the robots.

In the example in figure 2.2 the robot might be expected to spend a pre-defined amount of time at each way-point conducting a number of sub-tasks.

2.2.3 Exploration

The exploration problem (as tackled by Parker *et al.* [72, 73, 74], Rekleitis *et al.* [80], Simmons *et al.* [83] and Zlot *et al.* [97] in the literature) can be defined as follows:

> A robot (or group of robots) has the task of traversing an environment, whilst avoiding collisions with obstacles within the environment, gathering information.

This problem is related to the path planning problem, as at its base level it is simply a series of planned paths. The information gathered is typically a map of the environment. The complexity of the problem varies according to the type and accuracy of sensory equipment used. Typically a robot will attempt to localise itself within the environment (assuming a partial map is given *a priori*) using odometry and sensor readings. Using these sensor readings the robot will update its map of the environment.

A noteworthy sub-type of the exploration problem is hunting (as tackled in the literature by Cao *et al.* [30]). It can be defined as:

> A robot (or group of robots) has the task of traversing an environment, whilst searching for and capturing anomalous entities within the environment.

An example of the hunting problem is given in figure 2.4 where two robots attempt to capture an anomalous entity within the environment.

2.2.4 Foraging

Another common mobile robot problem is foraging (tackled by Balch *et al.* [6] and Sugawara *et al.* [87] in the literature). It can be defined as follows:

> A robot (or group of robots) has the task of traversing an environment, whilst avoiding collisions with other entities within the environment, collecting 'food' and depositing it in a pre-defined location (if the environment is known *a priori*).

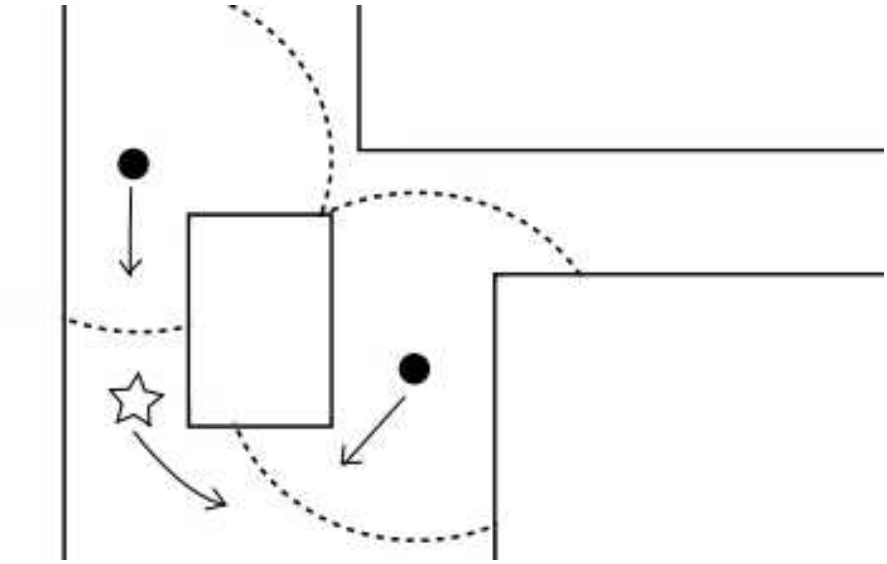


Figure 2.4: Two robots (black circles) attempt to capture an anomalous entity (star). Motion is signified by arrows. The dashed circles are the sensory limits of the robots.

The complexity of the problem is related to the spatial positioning of the 'food', the amount of co-ordination within the group of robots attempting the task and the accuracy of the 'food' detecting sensor. Typically a robot will traverse an environment randomly until a source of 'food' is discovered. The robot may then (depending on the system deployed) notify other robots within the system of the location of the 'food'. An example is given in figure 2.5. In which a group of three robots attempt to collect 'food' from the environment and deposit it in a pre-defined safe location.

A noteworthy sub-type of the foraging problem is the search and rescue problem (tackled by Jennings *et al.* [51, 52] in the literature). It can be defined as follows:

> A robot (or group of robots) has the task of traversing an environment, whilst avoiding collisions with other entities within the environment, and searching for a pre-defined target, manipulating it to a pre-defined safe location (if the environment is known *a priori*).

As such, the level of co-ordination in the search and rescue task is generally higher than that in a standard foraging task, as the manipulation process is a highly coupled one. An example is given in figure 2.6. Two robots need to co-ordinate their actions in order to 'rescue' the target.

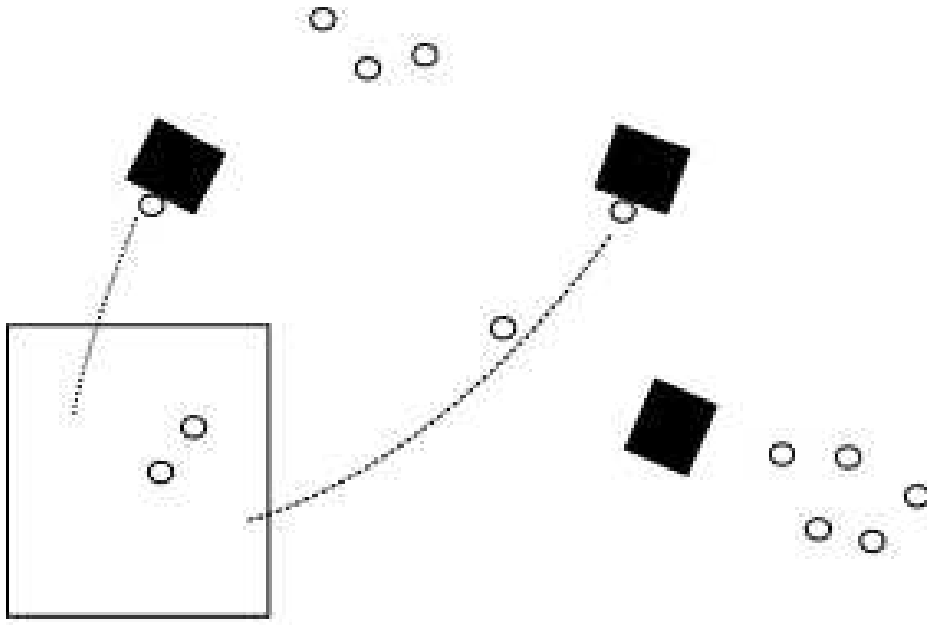


Figure 2.5: A group of robots (black squares) collect 'food' (white circles) and take them to a pre-defined location (white square).

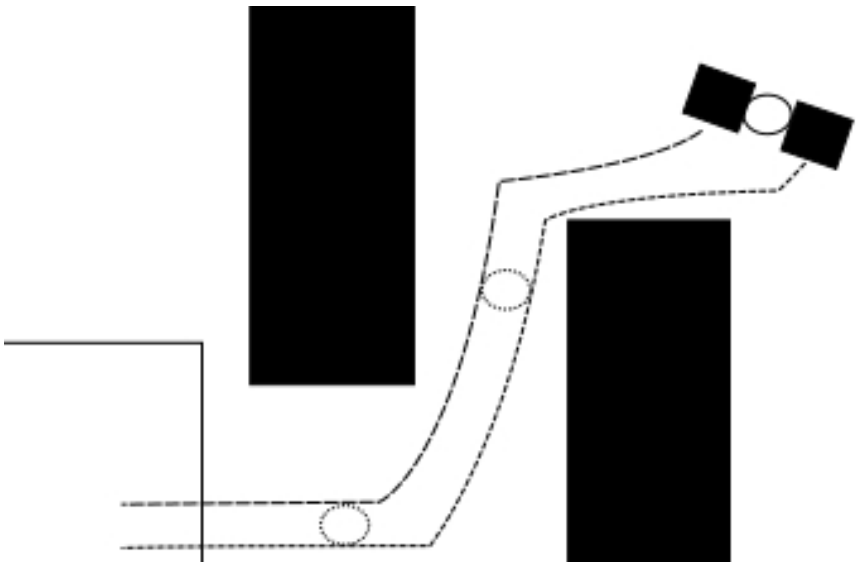


Figure 2.6: Two robots (black squares) attempt to manipulate the target (white circle) back to the designated safe area (white square), whilst avoiding obstacles (black rectangles).

2.2.5 Formation Control

A problem which is commonly explored is formation control (tackled by Balch *et al.* [7, 8], Monterio *et al.* [67] and Ogren *et al.* [70] in the literature). It can be defined as:

> A group of mobile robots attempt to traverse an environment (known or unknown) whilst maintaining a pre-defined formation (relative positions from one another).

The complexity of the problem increases with the number of robots involved, the complexity of the underlying traversal problem, and the complexity of the desired formation. Typically a 'leader' robot will be assigned, which has the task of traversing the environment. The other robots will have the task of keeping in formation with this 'leader' robot. Formations are kept by punishing robots for moving too far apart or too close together. An example of the formation control problem is given in figure 2.7. One group of robots forms a column formation and the other a line formation.

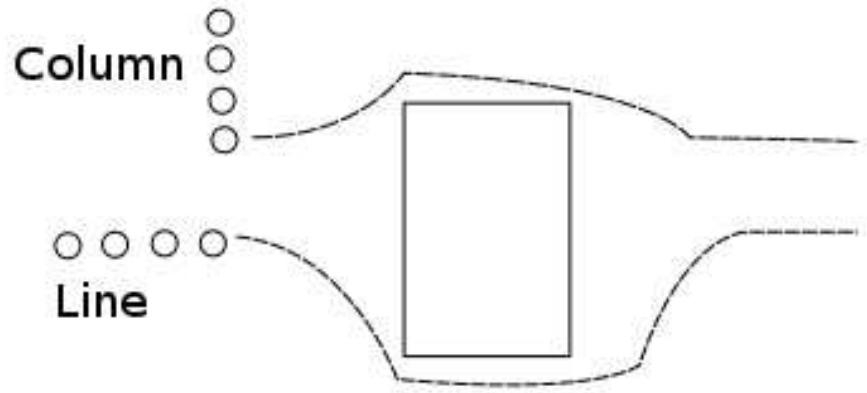


Figure 2.7: Two groups of robots (white circles) attempt to traverse the environment in formation. One group in line formation, the other in column formation.

2.2.6 Robot Football

The robot football problem, as tackled in the literature (Tews *et al.* [88], Veloso *et al.* [92] and Werger *et al.* [94]), is a hybridisation of the above problems and can be defined as:

> A group of robots attempts to play a game of football (based on rules defined by an international committee). As with real football, the aim is to have a higher score than your opponents.

A number of different robot football leagues exist, all varying in complexity, from systems using global tracking systems to position team members and opposition players, to systems that rely entirely upon local sensor information. Fundamentally, the task is the same.

Teams can put the solutions of other common robotic problems to good use in the robot football domain. These include formation control, coverage, and hunting, all of which could be advantageous to a team's strategy. Examples of how the above problems could be implemented in the robot football problem are given in figure 2.8.

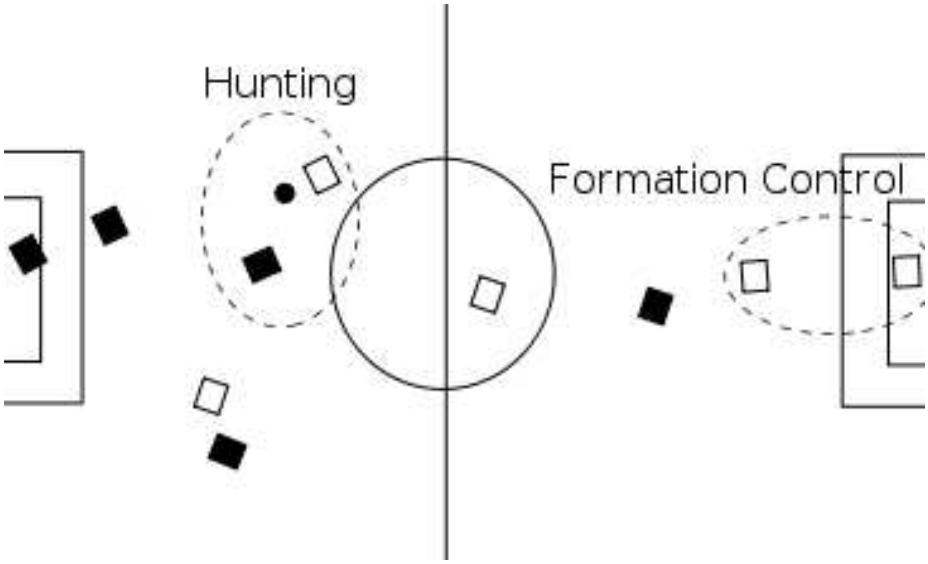


Figure 2.8: A conglomeration of the hunting and formation control problems. The two teams are represented by black and white squares the ball by a black square.

2.2.7 Summary

The problems defined and discussed in this section do not form an exhaustive list. However, they do cover the major basic problems found in robotics; generally other problems will be sub-types of the problems defined above or variations thereof.

In the following sections, a number of different robotic systems will be described. Each of them will naturally lend themselves to one or more of the problems defined here. For example, the potential field methods discussed in section 2.6 lend themselves to the coverage and formation control problems. When designing a robotic architecture it is important to take into account the type of problem that is being attempted, as the wrong choice could result in a multitude of unforeseen issues in the future. This is not to say that generic problem solving systems cannot be designed. However, the more generalised the system, the more likely it is that the solutions obtained will be sub-optimal. The task of the robotic system designer is to balance the optimality of the solutions with the usability of the system over a number of different tasks.

2.3 Robotic Architectures

In this section, the three major robotic architectures will be described: reactive, deliberative and hybrid systems. A number of examples will be given for each. The section will be concluded with a critical analysis of the three architectures.

2.3.1 Reactive Systems

In his book, Braitenburg conducted a number of thought experiments that showed the emergence of complex behaviours from a collection of simple sensor/actuator interactions [21]. For example, figure 2.9 shows Braitenburg's vehicle 3b; this simple vehicle is made up of two motors and two sensors, with the motors and sensors cross-connected. High sensor input results in low motor output. The vehicle can be described as an explorer, as it is attracted to regions of high sensor input but moves on to investigate other regions. It can be argued that these set of experiments were

the foundations for the reactive robotic systems methodology [2].

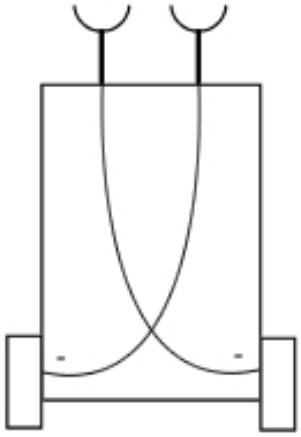


Figure 2.9: Two motors and two sensors are negatively cross connected, resulting in low motor output for high sensor input.

The most fundamental feature of reactive systems is the lack of abstract representational knowledge about the environment the robot can acquire, or is given *a priori*. In fact it is actively avoided as the process can be time consuming and in highly dynamic environments ultimately futile. Simply put, a reactive system has a *stimulus-response* relationship with the world rather than the traditional *perceive-reason-act* relationship a deliberative system has with the world (as discussed in section 2.3.2).

An early pioneer of the methodology was Rodney Brooks who developed the *Subsumption Architecture* [23, 25, 27, 28]. The architecture involves the concept of layered levels of behaviour, with the low level "survival" behaviours at the bottom of the hierarchy and the high level "goal" behaviours at the top. Figure 2.10 shows an example of the behaviour levels of a robot whose goal it is to traverse an unknown environment making some sort of observation. Each of these layers of behaviour run in parallel; the higher level behaviours can subsume control over the lower levels but not vice-versa. In the example, if the avoid behaviour detects an obstacle it would subsume control over the wander behaviour and so the robot will avoid the detected obstacle. Once the obstacle is successfully avoided, the robot defaults to the lower

level behaviour of wander. The more layers of behaviour within the architecture, the more complex tasks the robot can perform.

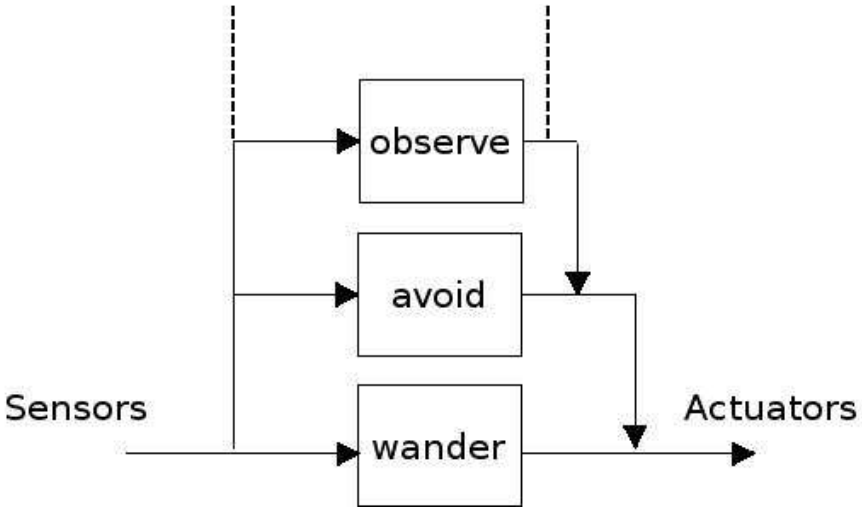


Figure 2.10: The higher levels such as "avoid" subsume command over the actuators from the lower levels such as "wander".

A rival reactive controller to Brooks's subsumption architecture is Arkin's motor schemas [1]. Arkin states that different motor behaviours (schemas) are required for different tasks. As with Brooks's subsumption architecture these behaviours are relatively simple on an individual basis but when combined produce more complex behaviours. Each of these schemas outputs a motion vector, which is multiplied by a gain value (this indicates the relative importance of each behaviour); the vectors are then summed. This command is then sent to the robot's motor controller

Balch *et al.* [7] successfully implemented Arkin's motor schemas in a multi-robot formation control problem. A number of motor schemas were implemented to give the desired behaviour of formation control:

- *move to goal*: Move towards a specific target location.
- *avoid static obstacle* Move away from a detected barrier.
- *avoid robot* Do not get too close to other robots.
- *maintain formation* Keep relative position with other robots.

Three methods for formation position determination were defined:

- *Unit-centre-referenced*: Each robot within the formation computed its own formation position relative to the average x and y positions of all other robots within the team.
- *Leader-referenced*: Each robot within the formation computed its own formation position relative to the position of a 'leader' robot. The 'leader' robot did not attempt to maintain formation; it was up to the other robots within the team to maintain the formation.
- *Neighbour-referenced*: Each robot within the formation computed its own formation position relative to another pre-defined robot.

It was found that the *unit-centre-referenced* approach provided the best results [7]. However, this approach may not be applicable to some scenarios. For example, if the 'leader' robot was replaced by a human leader, it would not be possible for them to calculate the unit centre on the fly and avoid obstacles at the same time — as such a *leader-referenced* approach would be more applicable. The *unit-centre-referenced* approach was also communications intensive, and so it is not applicable in scenarios where the bandwidth is limited.

Other behaviour based systems include Matarić 's basis behaviours [61]. These basis behaviours are control laws for locomotion to create complex group behaviours. For example, by performing a sum of the outputs from the *safe-wander*, *disperse*, *aggregate* and *home* behaviours, a *flocking* group behaviour was created. Descriptions of the behaviours are given below:

- *Safe-wander*: agents move around the environment whilst avoiding collisions.
- *Disperse*: agents maintain an arbitrary minimum distance from one another.
- *Aggregate*: agents maintain an arbitrary maximum distance from one another.
- *Home*: agents can navigate to a predefined region of the environment.

- *Flocking*: agents have a structured movement that minimises interference and protects individuals.

Both the *flocking* and a *foraging* group behaviours were implemented on a group of mobile robots, known as the "Nerd Herd". See figure 2.11 for a photograph of the Nerd Herd. However, these basis behaviours could not provide provable optimal solutions in complex domains. In [60], Matarić introduced social learning between mobile robots, in which robots learnt how to perform a behaviour through *imitation* and when to perform it through *social facilitation*. In essence, the robots could perform *mimicry*. Descriptions of these social behaviours are given below:

- *Imitation*: the ability to observe and repeat behaviour.
- *Social facilitation*: the process of selectively expressing certain behaviours.
- *Mimicry*: the ability to repeat the behaviour of another robot, without understanding the goal.

In order to learn the behaviour of other robots, Matarić proposed three forms of social knowledge and related reinforcement:

1. Direct reinforcement from movement towards goal.
2. Observation of other agent's behaviour.
3. Observation of reinforcement (reward/punishment) given to other robots.

The effectiveness of the proposed forms of social knowledge varied depending on the complexity of the rule(s) being learnt. This was expected. The high level of difficulty of certain rules, particularly altruistic social rules, would seem to lend them to genetic learning (a biologically inspired branch of machine learning), which would be in-line with biological studies in which animals do not learn altruism towards their kin but are born with it [64].

Monterio and Bicho applied a dynamical systems approach to behaviour-based formation control. Task constraints were represented as attractors or repellers. In

Figure 2.11: Matarić's "Nerd Herd" that she used in her social learning experimentation. Photo taken from [61].

the initial case study three agents attempted to traverse an unknown environment, whilst maintaining a triangle formation [67]. One of the three agents was assigned as the "leader", which had the task of driving from some initial position to a target location. The other two agents had the task of maintaining the triangle formation based upon the "leader" agent's position. The "leader"'s behaviour was generated by the summation of an attraction force-let (which attracted the system towards the target direction) and a repulsion force-let (which repulsed the system away from obstacles). The behaviour of the other agents was created by the same system dynamics; the "leader" was the target. One of the agents attempted to stay to the left of the "leader", the other the right. In [16], they applied the research to a two robot case, attempting to maintain column, oblique and line formations. In simulation, they extended the number of robots up to six. In both sets of experiments, simulation studies showed that smooth trajectories were generated which avoided collisions with other objects within the environment.

Balch *et al.* [6] made some interesting observations about communication within reactive multi-robot systems. They showed that communication improves system performance significantly in tasks with little *implicit communication* — communication via changes to the environment which other robots can detect. For example, the

foraging task described in section 2.2.4. In-line with these finding they also showed that, communication in tasks with *implicit communication* such as *graze* (discussed in section 2.2.2), was unnecessary. They also found that complex communication methods — the transmission of a robots current goal, gave little benefit over a basic communication methods — the transmission of robots current state.

Minguez *et al.* [65, 66] introduced a reactive controller, the Nearness Diagram (ND) algorithm, that navigated a robot to a goal location whilst avoiding obstacles. It also actively avoided local trap situations in "U" shaped obstacles common in potential field methods (see section 2.6.1 for more details on the limitations of the potential field method). The algorithm was deployed on a single robot in a number of highly cluttered and dynamic environments. In all experiments, the robot successfully navigated the environments avoiding collisions and trap situations.

The system as shown in figure 2.12 works as follows [1]:

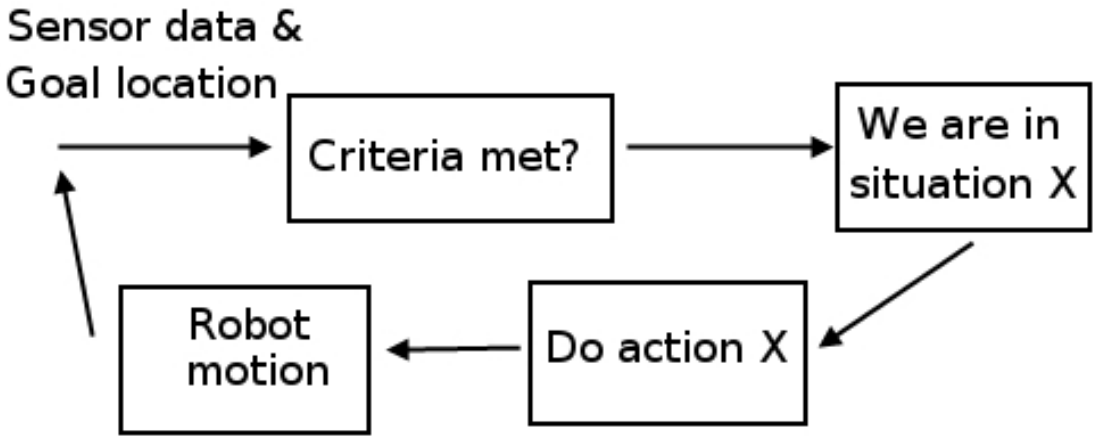


Figure 2.12: Based upon sensory input and a target location. The robot decides what situation it is in, and performs the related motor commands.

1. The robot collects sensory data and is given a goal location. The robot looks for gaps within the obstacle distribution; the closest gap to the goal that is navigable is defined as the free walk area.

[1] It is noteworthy to detail the system further, as the ND algorithm will be used as the reactive controller, in the hybrid system used in chapter 7

2. Based on the available data the algorithm attempts to match the current environment situation with a number of pre-defined situations. This is achieved through traversing a binary decision tree based on a number of criteria. One of four criteria are activated based on the existence and position of obstacles within the security zone of the robot. Criterion one is the safety criterion: this is either high safety (if no obstacles exists within the security zone) or low safety (if obstacles do exist). If the robot is in low safety, criterion two is activated (the dangerous distribution criterion). This is in either one of two states: Low Safety One (LS1) in which obstacles exist on one side of the security zone, and Low Safety Two (LS2) in which obstacles exist on both sides. If the robot is in high safety, criterion three is activated (goal within free walk area criterion). If the goal is within the free walk area — the robot is in the High Safety Goal in Region (HSGR) state. If not, criterion four is activated (wide/narrow walk area criterion): this is in either one of two states, High Safety Wide Region (HSWR); if the robot's free walk area is wide, High Safety Narrow Region (HSNR); if the robot's free walk area is narrow. Examples of each situation are given in figure 2.13.

3. Each situation has a set of actions related to them that will solve the task of avoiding obstacles whilst moving towards the goal. See table 2.1 for details.

4. Once the set of actions has been completed, this process is repeated until the robot is at the goal.

In [31], Chaimowicz *et al.* implemented dynamic role assignment within a group of co-operative mobile robots, which had the task of searching an environment for a number of targets and moving them to a pre-defined location. In order to encourage co-operation, each target required more than one robot to manipulate it. A number of roles which corresponded to motor controllers were defined prior to the task:

- *Exploration* which involves a random search of the environment.
- *Attach Lead* which is activated when a robot discovers a target. The robot broadcasts the available task and the required number of robots.

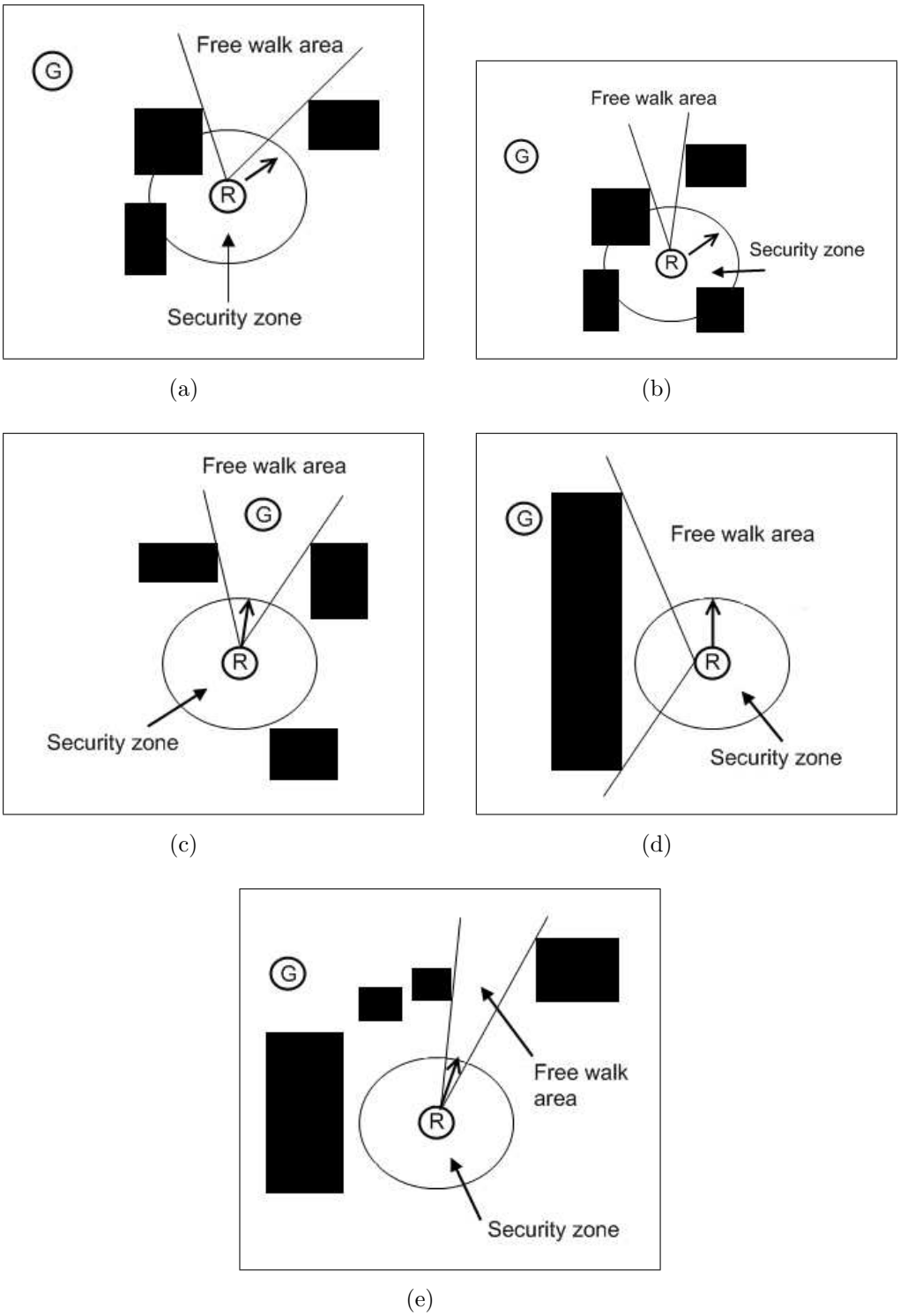


Figure 2.13: (a) Low safety 1 example. (b) Low safety 2 example. (c) High safety goal in region example. (d) High safety wide region example. (e) High safety narrow region example. Figure adapted from [65].

Table 2.1: Nearness Diagram algorithm. Situation — Action relationships.

Situation	Action
LS1	Move the robot away from the obstacle, and towards the closest gap of the free walk area.
LS2	Centre the robot between the two closest obstacles at both sides of the gap of the free walk area. Whilst moving towards the gap.
HSGR	Drive the robot towards the goal.
HSWR	Moves the robot alongside the obstacle.
HSNR	Drives the robot through the centre of the free walk area.

- *Approach* in which robots move towards a target.
- *Attach* in which robots are close enough to a target in order to manipulate it.
- *Transport* in which robots co-operate to manipulate a target to the desired location.

Robots switch among roles in a way that is dependant upon current state and broadcast information from the leader robots. Experiments were conducted with twenty *holonomic* (this term means that the robots can move freely along their x and y axes) robots and thirty target objects randomly distributed throughout the environment. The number of robots needed to transport an object was also randomly distributed between two and five robots. Positional information and communication was presumed to be error free. The results showed that the task completion time was related to the number of role re-allocations, with the time increasing with a lower number of re-allocations. This was expected, as the lower the number of re-allocations, the more likely a robot would be in an exploration role, and hence not able to transport an object.

A newer area of research for robotic control is that of artificial immune systems. One such system is the Dendritic Cell Algorithm (DCA) originally developed for network security by Greensmith *et al.* [48], and deployed in mobile robots by Oates *et al.* [69]. They implemented the DCA as the highest layer within a subsumption architecture. The system was implemented in the security robot domain — the DCA

was used to classify objects within an environment as either normal or anomalous. For example, a door that is open (that is normally closed) is classified as an anomalous result. It is suggested that once an anomaly is reported, a human security guard would be notified. Currently the system is in its infancy; it has been deployed on a Pioneer robot (a common research robot) with the *safe signal* being sourced from the laser range finder, the *danger signal* being sourced from the sonar array, and finally the *PAMP* (or signature of abnormal behaviour) being provided by the camera. In the set of initial experiments conducted by Oates *et al.*, the Pioneer robot traversed an environment classifying small pink obstacles as anomalous (as the laser range finder did not detect them), and large pink obstacles as normal (as the laser range finder detected them).

2.3.2 Deliberative Systems

The traditional approach to robotic control is to decompose the system into functional modules as shown in figure 2.14. Typically the perceive module involves the robot collecting sensor information to create a current state/world model. In the reason module, the robot calculates how to get from this current state to the desired state. Finally the act module involves the robot executing a number of tasks computed in the reason module. These modules are continuously looped until a final state/world model is reached. The efficiency of deliberative systems is reliant upon a static environment. Dynamic environments cause delays, due to loops between the *perceive-reason* modules.

Figure 2.14: Robots collect information about the environment/task, decide what to do, then perform the necessary motor commands.

STRIPS (STandford Research Institute Problem Solver) is a typical example of a deliberative system [42]. STRIPS was given a problem space defined by the initial

state of the world, a set of operators (action routines e.g. push box), with preconditions, post-conditions and a goal condition. STRIPS then attempted to identify operators that reduced the differences between the present world model and the goal. STRIPS was able to solve three general robotic tasks: turn on light switch, push three boxes together, and go to a location in another room. If given an accurate map of the environment and the starting locations of the three boxes, the light switch and itself. The tasks were not implemented in the physical domain; instead the solution was demonstrated in simulation only.

Action selection methods such as Bonet *et al.* looked at the problem of planning as a real time heuristic search problem [18], in which the search space is limited and agents move in constant time. The problem was split into discrete time intervals and the agent made a decision at each step as to what to do next. They proposed the ASP algorithm — a combination of a real-time A* (best-first, graph search) algorithm and a heuristic function. Over a number of block world problems they compared it against Kautz *et al.*'s SATPLAN (a combination of a stochastic search algorithm and problem encodings based upon propositional logic [53]), and Blum *et al.*'s GRAPHPLAN (a STRIPS-like planner which always returns the shortest possible partial-order plan, if a valid plan exists [17]). The ASP algorithm performance in the simple problems was comparable. However, in the more complex problems, the ASP algorithm performed the best, in terms of number of steps needed to compute a solution. These solutions were inferior to SATPLAN's solutions, although still reasonable.

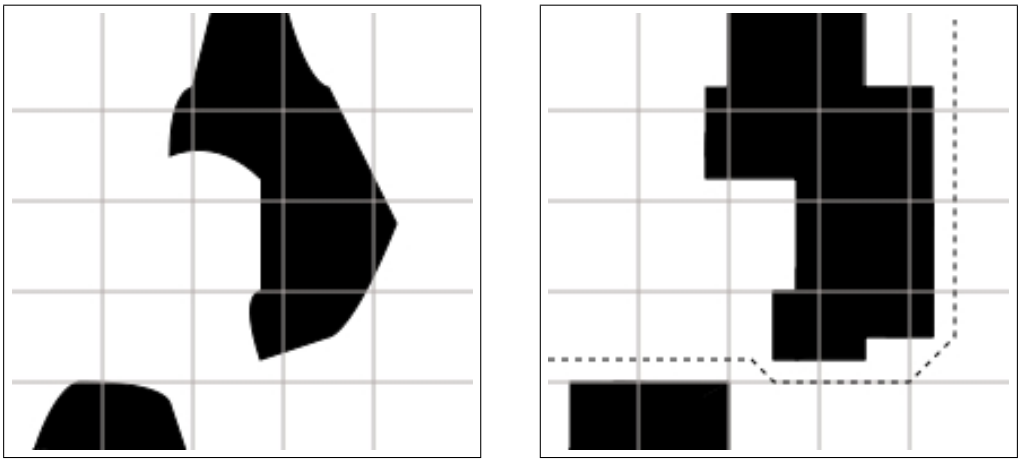
Many early deliberative systems relied upon complete information of the environment in order to construct plans. A number of approaches have been put forward for planning with incomplete information (initial state not known, sensory information available). Bonet *et al.* extended their previous work by formulating a plan with incomplete information into a problem concerning a heuristic search in belief space [19, 20]. The Real Time Dynamic Programming (RTDP) algorithm was proposed which combined a real time greedy search algorithm with dynamic programming updates. The RTDP produced results which were competitive with the best conformant planners (CGP — a graphplan based planner that produces non-contingent plans whenfaced with uncertainty [85] and CMBP — an algorithm that returns a set of all

possible conformant plans of minimal length, if such solutions exist [32]).

Etzioni *et al.* [40] provide an extension to STRIPS with the important assumption that the information collected about the world state was correct. The SNLP algorithm (a STRIPS-like planner [63]) was extended to allow plans to be generated with incomplete information. The algorithm was shown to be successful in solving problems in the UNIX domain. Petrick *et al.* [77] also tackled problems in the UNIX domain (planning to achieve UNIX goals, using UNIX shell commands as primitive actions). They used the concept of the planners knowledge state. Any actions taken were modelled in such a way that they modified the knowledge state of the planner, not the physical state of the environment. This high level abstraction made the system much more scalable than previous planning with incomplete information systems. As with other incomplete information systems inferential power was sacrificed for speed.

Parsons *et al.* proposed a path planner for multiple mobile robots, which gave complete collision free solutions [75]. The environment was modelled as a bounded planar workspace. Obstacles within the environment were known *a priori* and were of general polygonal shape. Robots were convex polygons. The task was to generate paths for each robot from a staring location to a target location, avoiding collisions with obstacles or other robots. A cell decomposition of free space was computed (this involved computing the free space of a single robot and then computing the sub-set of this free space that upheld an inter-robot constraint) and the resulting adjacency graph was searched for a path.

In [56], Leroy *et al.* tackled the path co-ordination problem for hundreds of robots. The path co-ordination problem can be defined as: n robots within an environment compute independent paths, co-ordinating when necessary to avoid collisions with one another. Again, a cell decomposition approach was applied. However, instead of computing the exact shape of obstacles, a bounding box representation was calculated (see figure 2.15). A classic cell decomposition approach was compared to the bounding box approach. The bounding box approach improved the scalability of the system. Indeed, results showed that the system could generate collision free paths for hundreds of robots within a reasonable amount of time e.g. a 150 robot case in under 5 minutes.



(a) (b)

Figure 2.15: (a) Cell decomposition of environment. (b) Bounding box representation of environment and generated path. Figure adapted from [56].

Chand *et al.*'s "book retrieval" robot [71] relied upon information given *a priori*. In fact the available routes it could take whilst moving to and from bookcases was embedded within the environment (by utilising floor markings). The robot followed these markings using an infrared line follower. What happened to the robot if it lost the line was not discussed. The task of retrieving a book was divided into sub-tasks: go to bookcase, retrieve book and bring book to loan area. Each task was programmed as a separate behaviour (motor controls), each of which were completed in sequence.

2.3.3 Hybrid Systems

Hybrid robotic systems aim to merge the benefits of deliberation with the robust nature of a reactive system. It is argued that purely reactive robotic systems are not appropriate for every possible robotic application. For example, on an assembly line the world (from the robot's point of view) can be modeled relatively accurately, and so a deliberative approach would be preferred. However, purely deliberative approaches can encounter difficulties when situated within the real (highly dynamic) world.

Arkin was one of the first to suggest the use of deliberation in conjunction with

a reactive control scheme to improve performance in robot navigation [1, 4]. The Autonomous Robot Architecture (AuRA), also used in [7] (as discussed in section 2.3.1), takes advantage of two types of world knowledge: (1) *Persistent* knowledge which is information concerning the environment given *a priori* that is relatively static and (2) *Transitory* knowledge which is information collected dynamically as the robot moves around the environment. This knowledge was not a prerequisite for navigation but resulted in a more efficient and flexible navigation. It should also be noted that this knowledge was only used when needed and only to reconfigure the reactive control schema. A number of experiments were conducted on real robots including a docking task in which a robot had the task of navigating a cluttered environment towards its charge station. The robot never computed a global path; instead it continuously reformulated its motor schema's based upon sensory information.

Arkin *et al.* considered a line of sight constrained approach to multi-robot exploration [3]. The line of sight constraint was a method of simulating communication constraints that may occur in a real world application. For example, a group of mobile robots exploring a building, with a high amount of metal in the structure may not be able to rely on the traditional RF method of communication. Two types of line of sight were defined:

- *direct*: in which a robot r can directly sense robot a.
- *indirect*: in which a robot r can directly sense robot a, which can directly sense robot b. Therefore robot r can indirectly sense robot b.

Three navigation strategies were proposed — the first approach "anchored wander" was a purely reactive system, in which one member of a team of robots acted as the communications "anchor". This robot did not move from its initial position within the environment. The other robots wandered around the environments in sequence (only one robot was moving at any one time) until a breach of the line of sight rule occurred, in which case the robot backtracked. The second strategy was the "Quadrant-biased anchored wander"; in this method a limited amount of world knowledge was given to the team of robots *a priori*. The environments in which experiments took place were divided into quadrants; the knowledge given *a priori* was

the quadrant in which a target resided. Each robot within the team (apart from the "anchor" robot) had its motion biased towards the direction of this quadrant. In the third and final strategy, (i.e. the "informed exploration") approach more environmental knowledge was given to the team of robots *a priori*. The team was given a rough estimate of the location of a target and enough map information to allow path planning. Results showed that in a relatively simple environment knowledge was not a necessity, with the "anchored wander" reaching 95% coverage. However, in a more complex environment knowledge provided a significant advantage.

Clark *et al.* introduce the concept of dynamic networks, where robots with limited sensing range formed local networks of robots *ad hoc* during a task. A centralised planner (one per network) plotted collision free trajectories in both 2-dimensional [35] and 3-dimensional [34] environments. See figure 2.16 for an example of two robots forming a dynamic network in order to plot collision free paths. After the trajectories had been plotted it was up to the individual robot's reactive controllers to avoid any unexpected obstacles, until a new trajectory was planned. Experiments conducted on real robots validated the systems' performance on groups of up to 8 robots in environments containing 5 stationary obstacles and 5 moving obstacles.

Liu *et al.* proposed a hybrid architecture for a robot football team [57], where appropriate behaviours were executed based upon estimations of applicability. For example, the closer a robot was to the opponent's goal the higher the applicability of the "shoot" behaviour.

Jarvis *et al.* used a deliberative planner to find a solution to a Travelling Salesman Problem (TSP) [78]. The solution was then used by robots as their optimum path through a partially known environment. They tackled a search and rescue problem in a disaster environment. It was assumed that some knowledge of the environment was given *a priori* i.e. a blue print of the building. This knowledge was used with a probabilistic model that determined the most likely areas of the environment people might have fled to during/after the disaster e.g. under door frames during earthquakes. These likely positions are used as vertices on the graph of the TSP. The weight of the edges was related to the time taken to travel the distance and the difficulty of the terrain. With this near optimum solution, robots then traversed the environment

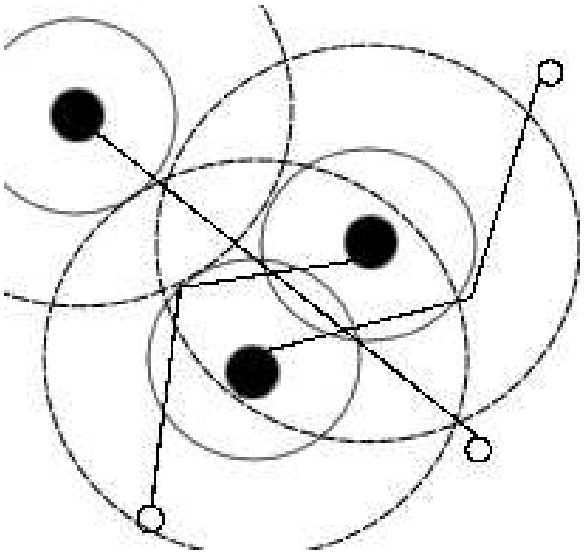


Figure 2.16: Two robots form a local network and collision free trajectories are plotted (solid lines). Another robot, outside of the local group plans a trajectory (non collision free). The solid circles represent the robots' sensor range. The dashed circle the local group radius.

trying to keep as close as possible to the pre-planned path. It was envisaged that some type of reactive controller would be used to avoid any unexpected obstacles. An example solution is given in figure 2.17, where each segment of the map is given a value based upon the probability of a target being located within that segment [78]. A TSP solution is then generated that will visit each of these segments.

Martins-Filho *et al.* tackled the mobile robot surveillance problem [59]. Their main concern was to provide unpredictable trajectories to a mobile robot which was traversing a known environment. It is important to note that the trajectories were unpredictable to an observer, but deterministic and so the robots' supervisor knew in advance the future positions of the robot. This is useful in situations were the supervisor would like to know the path of the robot *a priori*. The trajectories were sent to a motion control layer. No obstacle avoidance was undertaken in the experiments. However, adding such a process would be trivial.

Buckhard *et al.* integrated a *Belief-Desire-Intention* (BDI) type architecture into their robot football team [29]. BDI has an advantage over traditional Artificial Intelligence (AI) in that reasoning can be done in real time. A snapshot of environmental

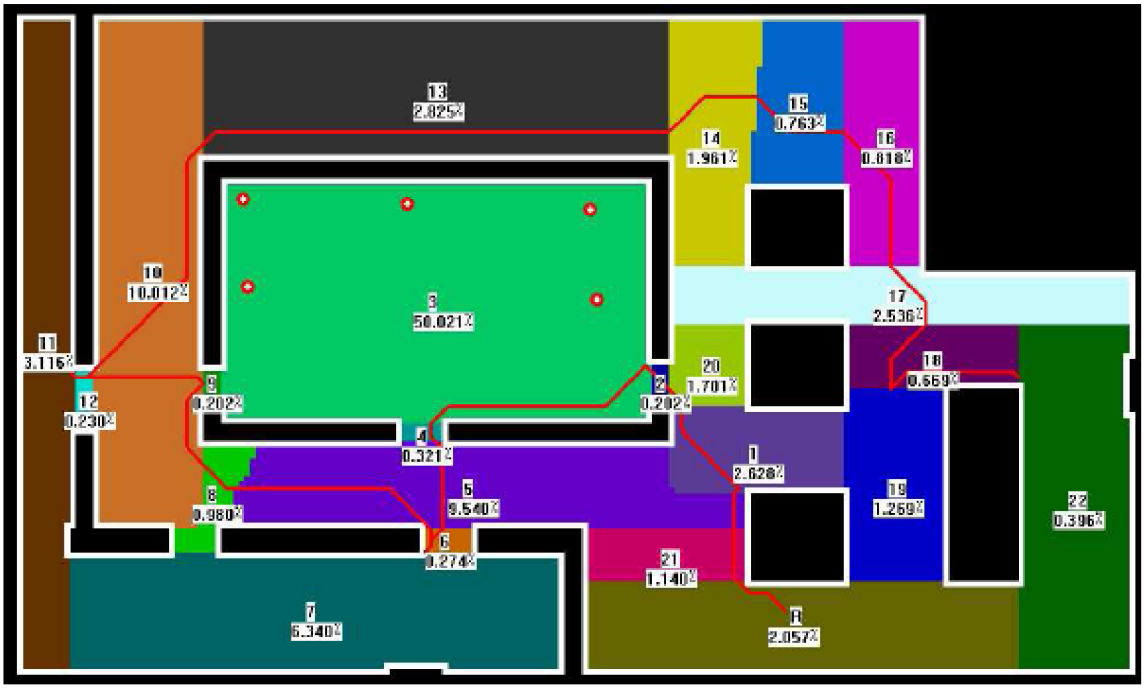


Figure 2.17: The robot starting location is at point R. The line is the path it will follow. The percentages represent the likelihood of finding a survivor in that map segment. The circles represent the actual locations of the survivors. Figure from [78].

data was defined as the world. This data could be incomplete or incorrect. Hence this set of worlds was defined as a set of *believed worlds*. As an agent can act in these worlds in a number of ways, a sub-set of worlds known as *desired worlds* existed. As only a sub-set of these worlds will be obtainable due to constraints, an agent had a choice from a set of *intended worlds*. In a game of football the set of *desired worlds* would consist of an ordered (in terms of utility) set of options e.g. score goal, intercept ball and block ball. The set of *intended worlds* would consist of 'intercept ball' and 'block ball' as the agent needs the ball in order to score a goal. As the global goal is to win the match which involves scoring goals, which necessitates having the ball, 'intercept ball' has a higher utility. The option of 'intercept ball' was realised by a number of skills (configurable plans) which consisted of the basic behaviours of the agent i.e. run, kick and dribble. The system was successfully deployed on the joint RoboCup '98 world champion team — AT Humboldt.

2.3.4 Critical Analysis of Robotic Architectures

As discussed in section 2.3.1, reactive systems do not take advantage of any knowledge gathered *a priori* and avoid the use of abstract representation. It is suggested that such techniques are time consuming and ultimately futile in highly dynamic environments, as the more complex the environment, the more computation is needed. Although this is true to an extent, computational power has increased rapidly since the initial conception of both deliberative and reactive systems. Reactive systems simply react to the current state of the environment based upon recent sensor information. As such, the efficiency of a reactive system is highly tied to the efficiency of the sensors — more precise information about the environment leads to better action selection; conversely poor information about the environment can lead to poor action selection. From a solution point of view, reactive systems do not guarantee near-optimal solutions due to their inherent lack of knowledge of the environment/task — indeed they do not guarantee any solution. Reactive systems are generally confined to relatively simple, non-critical tasks where knowledge of the environment is unnecessary and the simplicity of the task leads to cheap manufacture, e.g. the vacuum robot, Roomba.

Deliberative systems as discussed in section 2.3.2 use information gathered *a priori* about the environment/task to plan a near optimal solution. Early deliberative systems were dependant upon complete and accurate information; any unexpected event led to either poor performance or total task failure. Techniques that incorporated the use of incomplete and inaccurate information have improved the robustness of deliberative systems significantly. As noted, the improvement of processing power has also allowed deliberative systems to enter more dynamic environments. However, they still suffer from poor performance in highly dynamic environments. Similarly to the reactive systems that rely upon efficient sensor information, deliberative systems rely upon efficient mapping and localisation techniques. Deliberative systems are generally confined to static robots in manufacturing applications, as the "world" can be assumed to be known and all possible eventualities can be pre-determined.

Recent robotic systems tend to use high level deliberative planners combined with

low level reactive controllers. These hybrid systems as described in section 2.3.3 attempt to gain the benefits of both systems, whilst minimising their drawbacks. They attempt to get as near optimal solutions as feasibly possible in dynamic environments. Generally, they will stick to a near-optimal path within an environment/task unless forced to leave it by an unexpected event, where they will attempt to reacquire the near-optimal path as soon as possible. Hybrid systems are often deployed in situations that require highly robust controllers, but some guarantee of task completion is required, for example, a number of security mobile robots use reactive controllers to navigate the environment avoiding collisions. Just implementing a reactive system would not guarantee complete coverage of a given area. Hence a deliberative planner is used to force the robot to travel to different points within the environment.

2.4 Multi-robot Taxonomies

Section 2.3 described the architectures employed on individual robots. In this section, a number of possible multi-robot taxonomies will be discussed. One of the taxonomies will be used to categorise the multi-robot systems in section 2.5 as well as the new multi-robot system presented in chapter 4. This section will conclude with a critical analysis of the multi-robot taxonomies and an explanation on the choice of taxonomy.

2.4.1 Balch

Balch proposed taxonomies for multi-robot task and reward [5]. The research was centred around the use of reinforcement learning techniques for training groups of robots.

Balch's taxonomy of reward is given in table 2.2. In this taxonomy, five salient features were proposed. These features can be combined to describe a systems reward structure. Reward structures differ between systems as, in certain situations, the reinforcement function may not be the same as the performance metric. For example, if the robots within the system do not provide enough information, via their senors, for their performance to be evaluated accurately. As with the task taxonomy example, a reinforcement learning technique can be classified using the reward taxonomy.

Table 2.2: Taxonomy of reward.

Feature	Description
Source of reward	
INTERNAL_SOURCE	Reward is internal based on sensor values.
EXTERNAL_SOURCE	Reward is generated by external agent.
COMB_SOURCE	Combined internal and external reward.
Relation to performance	
PERFORMANCE	Reward is tied directly to performance.
HEURISTIC	Reward based on intuition of state value.
Time	
IMMEDIATE	Immediate rewards are provided.
DELAYED	Reward is delayed.
Continuity	
DISCRETE	Reward takes on several discrete values.
CONTINUOUS	Reward drawn from continuous interval.
Locality	
LOCAL	Individual agents receive unique rewards.
GLOBAL	All agents receive identical reward signal.
COMB_LOCALITY	Combination of local and global.

Balch's taxonomy of task is given in table 2.3. Six salient features of a task were proposed. By combining these six features it is possible to classify any given problem. For example, the foraging task as described in section 2.2.4 can be classified as: TIME_LIM (assuming a time limit is given); OBJECT_BASED; RESOURCE_LIM; COMP_INT; and SINGLE_AGENT. A 'Criterion' is not chosen as, in this case, the task was not an unlimited time task.

2.4.2 Farinelli

Farinelli *et al.* introduced a taxonomy based upon the level of communication and co-ordination among robots within a team [41]. A hierarchical view of the taxonomy is shown in figure 2.18. The first layer (top) distinguishes between co-operative and non co-operative teams of robots. The second layer represents the level of knowledge members of a team of robots has about one another. The third layer is concerned with the mechanisms used for co-ordination, if any. The final layer is concerned with the architecture communication/co-ordination of the system: that is, a centralised or

Table 2.3: Taxonomy of task.

Feature	Description
Time	
TIME_LIM	Fixed time task.
TIME_MIN	Minimum time task.
TIME_UNLIM	Unlimited time task.
SYNC	Synchronisation required.
Criteria	
CRIT_FINITE	Optimise over finite period.
CRIT_AVG	Average performance over all future.
CRIT_DISC	Discount future performance geometrically.
Subject of action	
OBJECT_BASED	Movement/placement of objects is important.
ROBOT_BASED	Movement/placement of robots is important.
Resource limits	
RESOURCE_LIM	Limited external resources.
ENERGY_MIN	Minimum energy task.
COMP_INT	Competition between team members for resources.
COMP_EXT	Team competes with external agents.
Group movement	
CONVERGENCE	Multiple robots converge.
COVERAGE	Multiple robots disperse.
MOVEMENT_TO	Movement to a position.
MOVEMENT_WHILE	Movement while maintaining position.
Platform capabilities	
SINGLE_AGENT	A single robot can perform the task.
MULTI_AGENT	Multiple robots are required.
DISPERSED	Agents must be dispersed.
SENSOR_COMPLETE	Robots can sense all relevant features.
SENSOR_LIM	World is only partially observable.
COMM	Communication amongst robots is required.

distributed architecture.

A brief description of each category of the taxonomy is given below:

1. *Unaware*: where robots act individually within a group and do not recognise group members.

2. *Aware — not co-ordinated*: where robots can differentiate group members from

the environment but still work individually (typically only simple communication methods are used to avoid interference).

3. *Weakly co-ordinated*: where the group does not use any form of explicit co-ordination. Instead, co-ordination is typically an emergent property of the system.

4. *Strongly co-ordinated — strongly centralised*: group members rely upon a 'leader' robot/agent to co-ordinate tasks.

5. *Strongly co-ordinated — weakly centralised*: groups do not assign a 'leader' robot *a priori*, the assignment is task/environment dependant.

6. *Strongly co-ordinated — distributed*: group members do not rely upon any 'leader' robot/agent; each robot makes its own decisions.

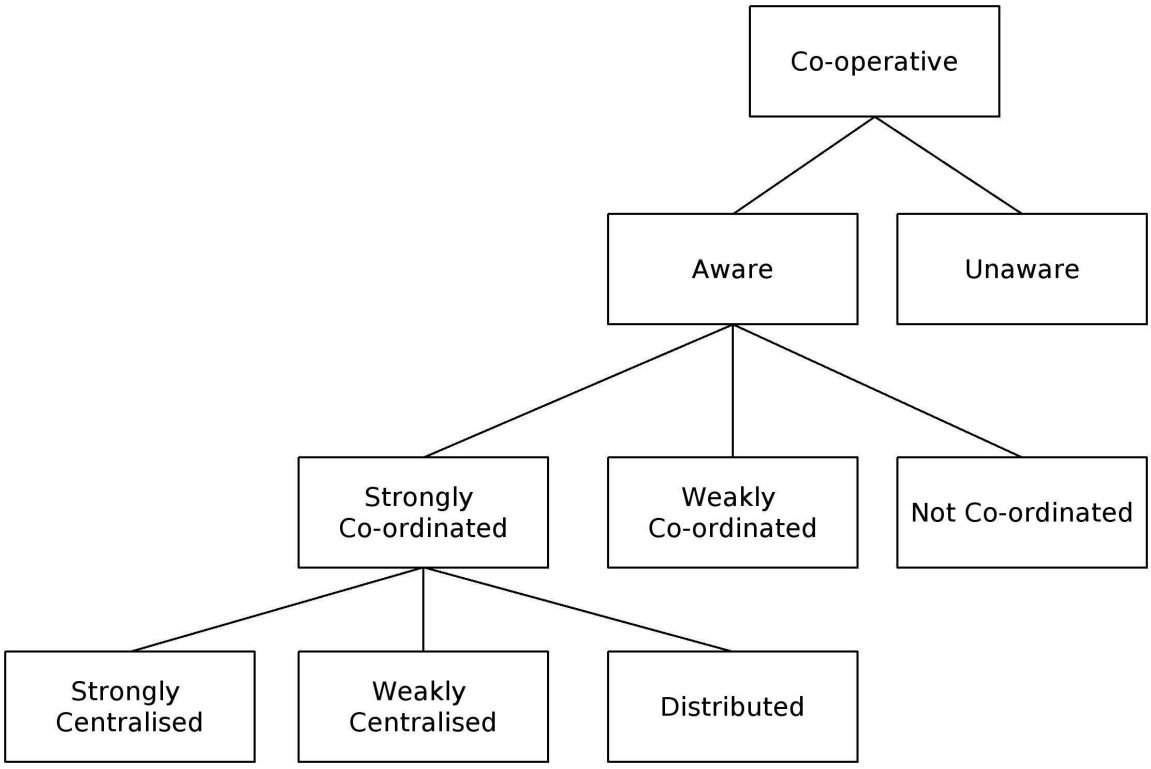


Figure 2.18: Taxonomy of communication and co-ordination. Figure adapted from [41].

2.4.3 Dudek

Dudek *et al.*'s taxonomy was first proposed for swarm robotics [38] and later extended to the more general case of multi-agent robotics [39]. The taxonomy is shown in tables 2.4 and 2.5. The original taxonomy for swarms is highlighted with an asterisk. An example classification based upon Dudek *et al.*'s taxonomy of a robot football system (as described in section 2.2.6) could be as follows: SIZE-LIM; COM-INF; TOP-ADD; BAND-MOTION; ARR-DYN; PROC-TME; and CMP-HOM.

Table 2.4: Taxonomy of Multi-agent robotics (part 1).

Feature	**Description**
Collective reconfigurability*	
ARR-STATIC	Static arrangement. The topology is fixed.
ARR-COMM	Co-ordinated re-arrangement. Re-arrangement between team members that communicate.
ARR-DYN	Dynamic arrangement. The relationship of team members can change arbitrarily.
Processing ability*	
PROC-SUM	Non-linear summation unit.
PROC-FSA	Finite state automaton.
PROC-PDA	Push-down automaton.
PROC-TME	Turing machine equivalent.
Collective composition	
CMP-IDENT	Identical. The collective is made up of homogeneous robots. Both in terms of hardware and software.
CMP-HOM	Homogeneous. The collective is made up of robots with the same hardware.
CMP-HET	Heterogeneous. The collective is made up of robots with differences in hardware.

2.4.4 Gerkey

Gerkey *et al.* propose a taxonomy of Multi-Robot Task Allocation (MRTA) problems [45]. The following three axes were proposed:

- Single-task robots (ST) *vs.* multi-task robots (MT).

Table 2.5: Taxonomy of Multi-agent robotics (part 2).

Feature	Description
Size of the collective*	
SIZE-ALONE	1 robot. The minimal collective.
SIZE-PAIR	2 robots. The simplest group.
SIZE-LIM	Multiple robots. The number of robots is relatively small to the size of the task/environment.
SIZE-INF	There is effectively an infinite number of robots.
Communication range	
COM-NONE	Robots cannot communicate with other robots directly. Indirect communication is still possible.
COM-NEAR	Robots can only communicate with other "nearby" robots.
COM-INF	Robots can communicate with any other robot. Infinite communications range.
Communication topology*	
TOP-BROAD	Broadcast. Robots communicate with all robots in the collective or none.
TOP-ADD	Address. Robots can communicate with arbitrary robots using a unique address.
TOP-TREE	Tree. Robots are linked in a tree structure and can only communicate through this hierarchy.
TOP-GRAPH	Graph. Robots are linked in a graph structure. A more general and robust form of the tree structure.
Communication bandwidth*	
BAND-INF	Communication is free. The cost/overhead of communication can be ignored.
BAND-MOTION	Communication costs are of the same magnitude of the cost moving the robot between locations.
BAND-LOW	Very high cost. Communication costs are much higher than the cost of moving from one location to another.
BAND-ZERO	No communication. Robots are unable to sense each other.

- ST means a robot can only perform one task at a time. MT means a robot can perform multiple tasks at once.

- Single-robot tasks (SR) *vs.* multi-robots tasks (MR).
  - SR means that each task requires only one robot to complete it. MR means that some tasks may require more than one robot.

- Instantaneous assignment (IA) *vs.* time-executed assignment (TA).
  - IA means that planning for future allocations is not possible. TA means that planning of future allocations is possible.

These three axes allow for a possible eight combinations as given below:

- ST-SR-IA:
  - Examples include Parker's L-ALLIANCE architecture [73, 74] and Werger *et al.*'s BLE [95], as discussed in section 2.5.6.

- ST-ST-TA:
  - Examples include Dias *et al.*'s market based controller [37] and Parker's ALLIANCE architecture [72], as described in section 2.5.5 and 2.5.6 respectively.

- ST-MR-IA:
  - Examples include Fua *et al.*'s COBOS architecture [43] and Rekleitis *et al.*'s [80] multi-robot localisation problem, as described in sections 2.5.5 and 2.5.6 respectively.

- ST-MR-TA:
  - As an example; Jennings *et al.*'s [52, 51] multi-robot search and rescue problem is a type of ST-MR-TA problem, as described in section 2.5.6.

- MT-SR-IA and MT-SR-TA:
  - These types of problem are currently uncommon, as it assumes that robots can execute multiple tasks concurrently. This requires highly accurate actuators, which are not typically available.

- MT-MR-IA and MT-MR-TA:
  - Robot football is a good example of both MT-MR-IA and MT-MR-TA problems, the difference in the robot football case being the inclusion of a 'coach' agent within a team to produce future task allocations. For example, CMUnited [92] and Tews *et al.* [88], as described in sections 2.5.4 and 2.6 respectively.

2.4.5 Critical Analysis of Multi-robot Taxonomies

Whilst Balch's taxonomy of task can be used to categorise a wide range of multi-robot tasks, his taxonomy of reward is heavily reinforcement learning orientated. By changing the parameters of the taxonomy to suit other learning methods the taxonomy could be expanded. However, the taxonomy is of no real use to non learning systems.

The papers by Dudek *et al.* and Gerkey *et al.* provide good generalised taxonomies that are able to accommodate the majority (if not all) of multi-robot systems. Gerky *et al.*'s taxonomy of multi-robot task allocation, as the name suggests, is centred around the task that has to be completed (as Balch's taxonomy of task). This is unlike Dudek *et al.*'s taxonomy of multi-agent robotics and Farinelli *et al.*'s taxonomy of communication and co-ordination, which are centred around the multi-robot system.

As the aim of this thesis is to present a new multi-robot system, it would seem appropriate to use a multi-robot taxonomy centred around the multi-robot system rather than the task. Indeed, the task to be completed was not a controlling factor of the design of the new multi-robot system presented in chapter 4. Farinelli *et al.*'s taxonomy has been chosen as the taxonomy to be used throughout the rest of this thesis as it focuses on the communication and co-ordination aspects of the multi-robots system, which is the research area of this thesis. The relative ease of assigning multi-robot system to categories within the system is also attractive.

2.5 Multi-robot Systems

In this section, several multi-robot systems will be discussed. The following sections are categorised according to Farinelli *et al.*'s Multi-robot taxonomy, as discussed

above. This section will conclude with a critical analysis of multi-robot systems.

2.5.1 Unaware Systems

Sugawara *et al.* compared an interacting and a non-interacting system in a task to forage for pucks [87]. Interaction was in the form of robots broadcasting their current position if they found a puck. This was achieved through radiating light (each robot was enabled with a light sensor) for a short period of time, before returning the puck to the desired location. An example of the system is shown in figure 2.19, in which one robot finds a puck and then emits a signal to notify other robots within the area that it has found a puck. The other robot within the system is then attracted to the region in which the first robot found the puck. They showed that, if no interaction took place, the performance of the system was proportional to the number of robots. However, if interaction did occur, the performance of the system was dependant upon the spatial distribution of the pucks: i.e. the closer together the pucks were, the less time it took to collect all of the pucks.

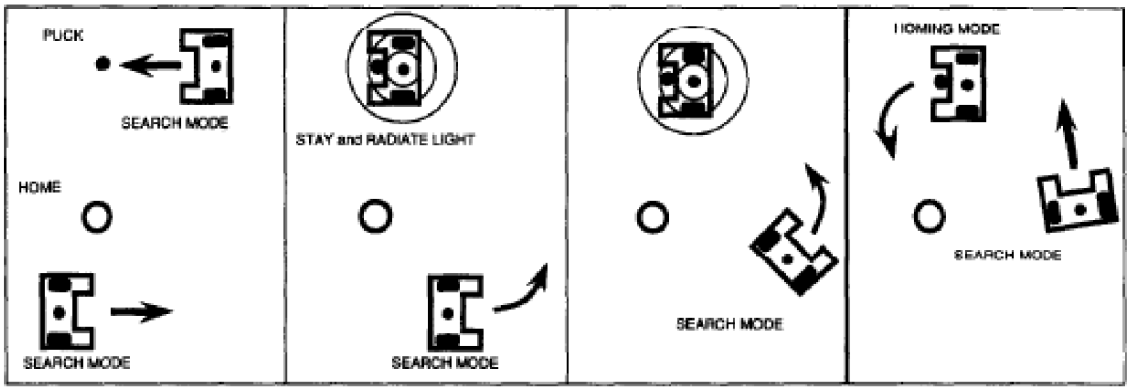


Figure 2.19: One robot discovers a puck and so radiates light for a set period of time. The other robot is attracted to this light. After radiating the light, the first robot takes the puck to a pre-defined location. The figure is taken from [87].

Werger employed a behaviour based approach to robot football in which team members entered into formations without using any explicit communication [94]. It was found that team members entered into offensive and defensive formations based

upon the behaviour of other agents and the position of the ball (no symbolic distinction between opponents and team members was needed).

2.5.2 Aware, Not Co-ordinated Systems

Balch *et al.* [8] introduced social potentials to tackle the formation control problem in simulation. Depending on the formation, each robot had a number of local attachment sites that other robots within the group were attracted to. See figure 2.20 for an example of the attachment sites for a column formation. No global communication protocol was needed — local sensors were used to detect team members and acquire free attachment sites. This made the system highly scalable. Experiments were conducted on groups of 1 to 8 robots. Of the four formations used (diamond, line, column and square), the column formation produced the best performance when traversing an obstacle field.

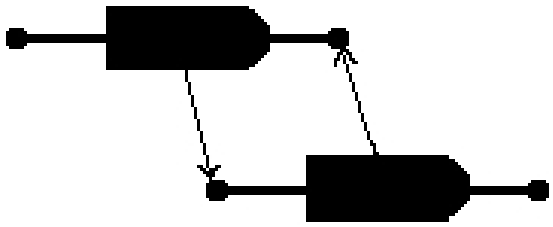


Figure 2.20: Attachment sites for column formation: The arrows signify the motion of the robots, moving into formation. Figure adapted from [8].

In [9], Batalin *et al.* address the problem of deploying mobile sensor networks. They used a behaviour-based architecture, as defined in section 2.3.1. In the first of three systems described, individual robots simply moved in the direction that best improved their coverage. In the second system, robots within a given distance of one another formed coalitions, and decided on subsequent motions based upon relative positions and bearings. In the third system, robots were simply repelled by other robots within their field of view. This non co-ordinated approach outperformed the co-ordinated approach. Both of the systems that could differentiate other robots from obstacles within the environment outperformed the individual system.

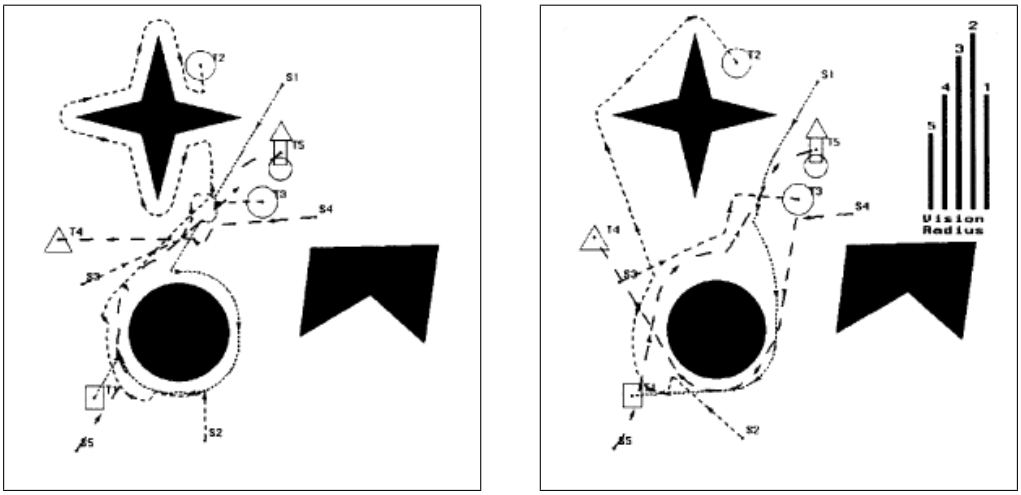
2.5.3 Weakly Co-ordinated Systems

Howard *et al.* used a potential field based approach to deploy a mobile sensor network on a single floor of a simulation of a large hospital [50]. The potential fields were constructed so that each node within the network was repulsed by other nodes and obstacles within the environment. The nodes were also subject to viscous friction to ensure that the network stabilised to a static state. As a result, the algorithm solved the coverage problem for this environment.

Gazi also implemented a potential field based approach in which he proposed the use of 'sliding mode control' [90] for implementing foraging and formation control tasks [44]. The numerical simulated examples showed that the system would form stable swarms and form desired formations.

Cao *et al.* proposed a distributed control approach to multi-robot hunting, which they termed 'local interactions with local co-ordinate systems' (LILCS) [30]. Each robot within the system made its own decisions based upon its current sensor readings. Co-operation emerged from local interactions within the system that may have been beneficial to the task. Results showed that the system could successfully capture evaders in three different simulated environments.

Lumelksy *et al.* [58] introduced the "Cocktail Party Model" for decentralised multi-robot motion control, in an unknown environment. Robots sensed objects within a given range, and differentiated between robots and other obstacles. At all times, the robots knew their current position within the environment and their target position. No explicit communication occurred between robots. Robots followed a straight line path to their desired target until they came across an obstacle. The robots then attempted to follow the boundary of the obstacle until they reached their initial straight line path once more. It was shown that an increased sensor range improved the performance of the system, as it led to robots pro-actively avoiding one another. An example is shown in figure 2.21. In figure 2.21a, each robot has a very limited sensing capability, and so the system is prone to inter-robot collisions. In figure 2.21b, each robot has a much improved sensing capability. The result is a reduced amount of inter-robot collisions.



(a) (b)

Figure 2.21: a) A system with very limited sensing capability. b) A system with varied (but improved) sensing capability, as shown by the bar chart. The figures are taken from [58].

2.5.4 Strongly Co-ordinated, Strongly Centralised Systems

In [67], (described in section 2.3.1), it was assumed that the "leader" agent broadcast its position to the other agents within the system. If the other agents failed to receive the transmission (or indeed if it was never sent), the agents would have no way of maintaining the triangle formation.

The Champions of Robocup '97, CMUnited [92], used a centralised interface computer that was connected to the camera system, which tracked all robots on the field and the ball. It was also connected to each client module, which calculated the action primitives for specific robots within the team, based upon pre-game agreements and the current world state. The interface computer transmitted these actions to individual robots via RF.

In [49], Howard *et al.* tackle the coverage problem, with the added constraint of maintaining line of sight between the robots. The real time simulation experiments showed that the algorithm employed was practical for groups containing up to 50 robots. The line of sight constraint was used, as it was assumed that a Global Positioning System (GPS) was not available. Robots needed to maintain their line of

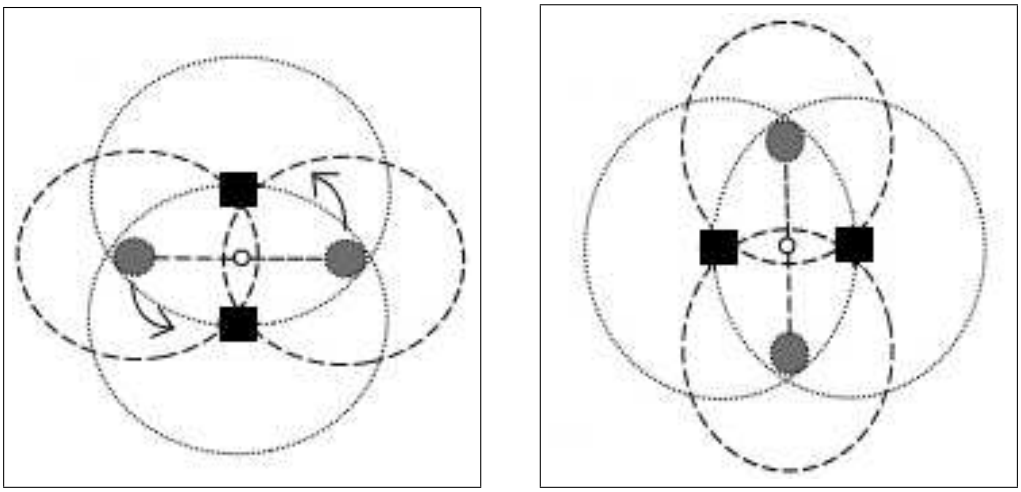
sight in order to use each other as landmarks. A central algorithm cycled through the following processes until all robots had been deployed or the environment had been completely covered:

- *Initialisation*: Robots are assigned one of three states; waiting, active or deployed. All robots are initially in the waiting state apart from the anchor robot which is deployed at the start.
- *Selection*: Sensor data from all of the robots is combined to form a global map of the known environment. The next deployment location is based on analysis of this map.
- *Assignment*: The next waiting robot is assigned the deployment location. If the robot is unable to reach the location, due to other robots obstructing its path, then reassignment of the deployment locations takes place.
- *Execution*: Robots make their way to their deployment location. Robots move sequentially in order to maintain line of sight.

Simmons *et al.* [83] described a technique for co-ordinating heterogeneous robots during an exploration and mapping task. Two problems were considered; creating a single consistent map of the environment, and how to explore the environment in order to create the map in the most effective manner. Each problem was tackled in a similar way. Each robot created its own local map using a laser range finder. A central mapper pooled all of the local maps together to create a single global map. Local mappers helped to reduce errors in the global map through localisation techniques. The central mapper also reduced errors in the global map by iteratively combining local maps. The major assumption was that each robot's calculation of its own position and pose relative to one another, was accurate, and that the communications method had a high bandwidth. During exploration, each robot constructed a bid, describing the expected gain and cost of exploration. A central planner pooled these bids and attempted to assign tasks whilst maximising the global utilisation. During experimentation the system was deployed on three environments (office-like, random and

obstacle free). In the office-like environment, two robots significantly outperformed one robot, whilst the increase in performance between three robots and two robots was insignificant. In the random environment, groups of three robots significantly outperformed groups of two robots. In the obstacle free environment, the three robot groups performed the worst. From these results, a relationship between obstacle frequency and group size can be observed with obstacles helping to limit inter-robot interference.

In [70], Ögren *et al.* tackled the problem of vehicle formation control using artificial potentials and virtual leaders. The artificial potentials defined the interaction forces among neighbouring vehicles, virtual leaders and nearby vehicles. Virtual leaders were used to herd the vehicles into the desired formations and towards the goal location. An example of virtual leaders forcing a formation of vehicles to rotate is given in figure 2.22.



(a) (b)

Figure 2.22: Squares represent vehicles, grey circles are virtual leaders. a) Initial position and orientation. b) Position and orientation after a 90^o rotation. Dotted and dashed circles represent the region of effect for each element within the environment. The figures are adapted from [70].

2.5.5 Strongly Co-ordinated, Weakly Centralised Systems

Reif *et al.* [79] tackled the problem of controlling Very Large Scale Robotic (VLSR) systems. They introduce the concept of *social* potential fields, where the motion of individual robots was related to the resultant force imposed by other components in the system (robots and obstacles). They defined three types of robots in their system:

- *Leader robots* which were not affected by the force laws of the system and whose motion was either pre-planned or human controlled.
- *Landmark robots* were static and not affected by the force laws of the system. They imposed forces on other robots in the system, and so could be used to designate areas of interest/non-interest.
- *Ordinary robots* which were autonomous and their motion was a result of the force laws of the system.

Simulations of the system completing a guard task (similar to the hunting problem described in section 2.2) and a mine-sweeping task (similar in concept to the graze problem described in section 2.2) were conducted.

In [84], Simmons *et al.*, described an approach to co-ordinate three heterogeneous robots in order to solve an assembly problem. Each individual robot's architecture was made up of three layers: a planner, that solved high level goals; an executive that sequenced and monitored tasks; and a behavioural layer which interfaced with sensors and actuators to provide low level command e.g. obstacle avoidance. Each of these layers interacted with those above and below it. In the multi-robot situation each layer could interact with the corresponding layer of another robot. See figure 2.23 for an example of two robots' architectures co-ordinating at individual layers. By enabling the planner layers to interact, global resource utilisation could be improved. This was done by the robots bidding to become the leader of the group. This leader robot negotiated with other robots to form teams and assign tasks to them, as well as monitoring progress and adapting the plan if new needs arised. The robots could also negotiate with other (non leader) robots in order to carry out an assigned task.

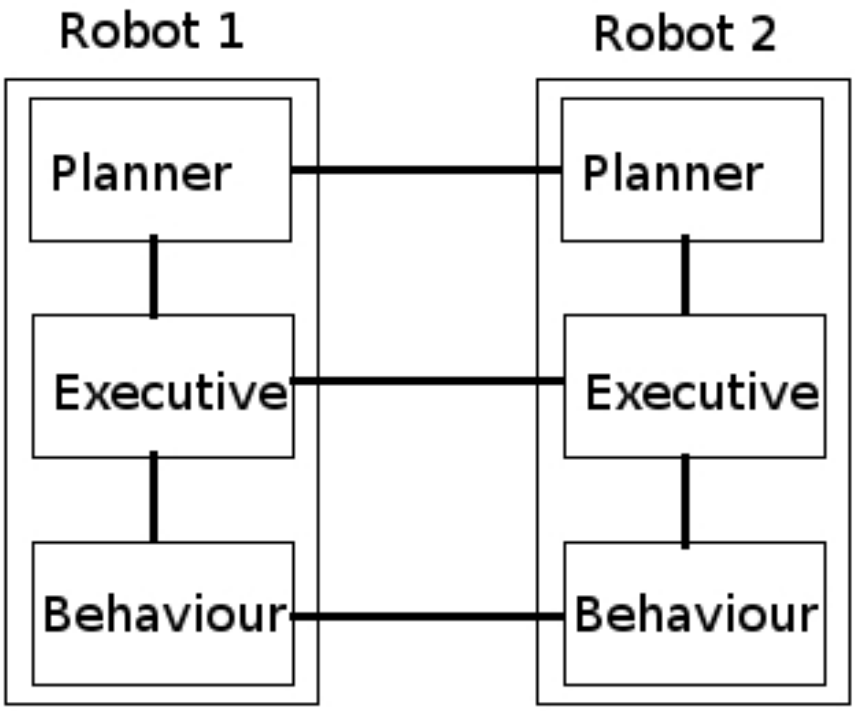


Figure 2.23: Each layer can interact with neighbouring layers, and corresponding layers on other robots. Figure adapted from [84].

Dias *et al.* [37] introduced a market-based controller with opportunistic leaders. Market-based controllers for multi-robot systems were first introduced by Stentz *et al.* [86]; they were based upon the "free market" system from economics. Robots bid and negotiate to carry out tasks. Both co-operation and competition are utilised, to gain the maximum net profit for the "market", whilst also maximising individual robot's personal profits. Dias *et al.* extend this work from single-party, single-task negotiations to multi-party, multi-task negotiations — this enabled the groups to escape local minima in the solution space. Leaders were used within groups to provide better solutions (leaders had access to global state information). A distributed travelling salesman problem was attempted in simulation. The problem was defined as follows:

A group of robots, with different starting locations, in a known environ-

ment, had the task of visiting a number of pre-defined locations within the environment, whilst attempting to minimise the time taken for each location to be visited.

Each robot was given a map of the environment *a priori* and, based on this map, each robot could assess the individual cost of visiting each location. Individual robots bidded to visit each location, whilst trying to maximise their revenue and minimise their costs. The architecture was shown to significantly improve the team solution through a series of 'city-for-revenue' deals between pairs of robots.

Fua *et al.* tackled the problem of multi-robot task allocation (with limited communications range), and, in doing so, introduced the 'Co-operative Back-off Adaptive Scheme' (COBOS) [43]. Initially, each robot started assuming it could solve all tasks. During the experimentation, tasks were generated by a central planner at random, making *a priori* planning impossible. This was an attempt to mimic a dynamic environment. Information about the tasks and robots was broadcast among robots within communications range. If a robot left communications range, the information held about it i.e. state of task being attempted, was stored and only changed once the robot re-entered communications range. Hence, it was unknown whether or not a robot outside communications range had completed a task or had failed. Robots learnt through "experience" whether or not they could perform a task in that they back-off from a task if they estimated the time it would take them to complete the task was longer than the group computed average time. Each task had a number of requirements associated with it. For example, a robot without a gripper would not attempt a task that had the requirement "manipulate object" associated with it. Experiments were conducted under simulation which verified the effectiveness of the proposed scheme.

2.5.6 Strongly Co-ordinated, Distributed Systems

Zlot *et al.* [97] used a market-based approach to solve a multi-robot exploration problem. Unlike the *weakly centralised* approach by Dias *et al.* [37], their approach was entirely distributed i.e. no leader robots were used. Price information was used as a means of low bandwidth communication. The system was deployed on three different

environments: a large indoor open space; an outdoor area with a limited number of static obstacles; and an indoor area with numerous static and dynamic obstacles. Negotiation based strategies were shown to significantly outperform a greedy algorithm based system and a no communication based system in each environment.

In [80], Rekleitis *et al.* tackled the two robot case of the multi-robot exploration problem. Each robot used the relative position of its team mate (each robot had a *robot tracker* sensor, that observed and reported the relative pose of the robot) to update its own estimation of position. This helped to reduce the effect of errors abundant in dead reckoning. To ensure that line of sight between the team members was maintained at all times, the robots moved in sequence (at most one robot was moving at any one time). Experiments in both simulation and on real robots validated the hypothesis that joint exploration and localisation can lead to more robust modeling than odometry alone, due to the reduced error.

In [95], Werger *et al.* continued their previous investigation into port-arbitrated behaviour [93], the abstractions and techniques which Brooks implemented in his Behaviour Language [26], and used to good effect in [62]. They introduced the Broadcast of Local Eligibility (BLE) approach to heterogeneous multi-robot co-ordination. During a task, an individual determines its eligibility to perform a sub-task; this was compared with the best eligibility calculated by other members of the group. The robot with the best eligibility claimed the sub-task and hence inhibited the related behaviours of other members of the group. If the robot completing the sub-task failed, it was freed and was immediately available to be claimed by another robot. No explicit negotiation or recognition took place. Experiments were conducted comparing the BLE system against a standard local subsumption system, a local greedy system (the behaviour with the highest evaluation function controls the robot) and random controller (only the random wander behaviour was implemented). Each system had the task of navigating an environment in an attempt to track a number of targets (similar to the surveillance problem described in section 2.2). Results showed that the BLE significantly outperformed all other systems in this scenario.

Jennings *et al.* presented an algorithm for distributed robotic search and rescue, in both the two robot case [52] and the three robot case [51]. As is shown in figure

2.24, the problem was split into two sub-problems. At first the robots "search" for the target: i.e. the robots move around the environment in a random fashion until they found an object. In the second sub-problem, the robots "rescue" the target. This problem is only activated once an object is found. The robot that discovered the object transmitted a broadcast message, giving its current position. All other robots in the system made their way to the broadcasting robot. Once two robots were at the object they would attempt to manipulate the object to a pre-specified goal location. No explicit transmission of messages or internal states occurred between robots. Instead, each robot could make remote procedure calls on other robots in their team. Robots could retire from a task if they detected a fault i.e. low battery, in which case the robot that was left would attempt to find a substitute robot, by explicitly requesting a new team member.

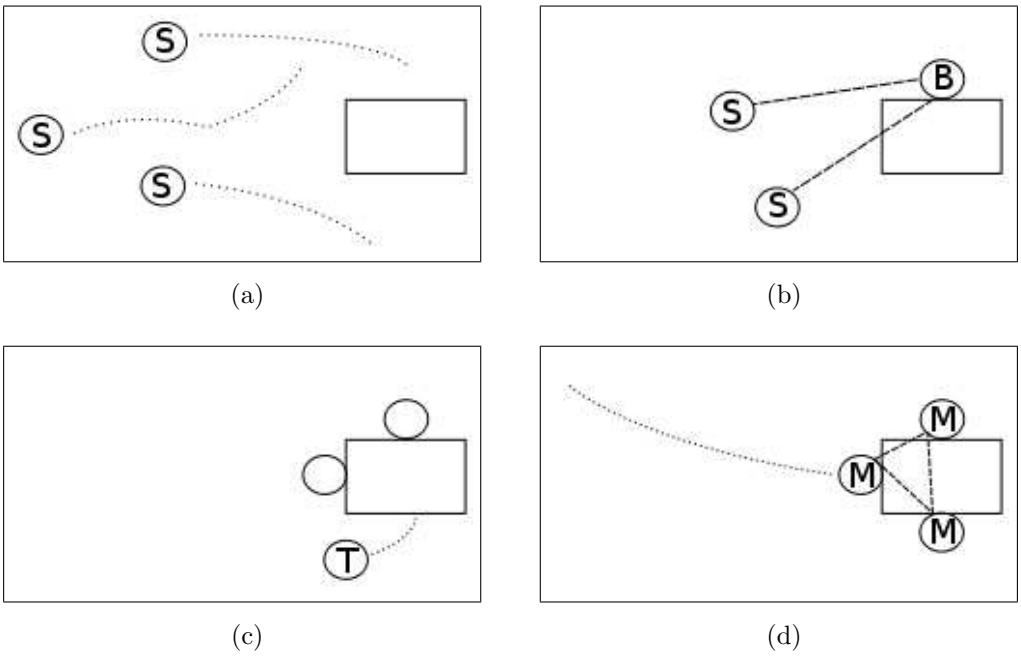


Figure 2.24: Three robots (circles) are required to rescue the target (rectangle). (a) Robots search the environment (S). (b) One robot finds the target and broadcasts (B) a message for assistance to the other robots. (c) Robots make their way to the target location (T). (d) Robots manipulate the target (M) to the desired location. Dashed line represents inter-robot communication. Dotted lines represent desired paths of the robots.

Parker introduced ALLIANCE, a fault tolerant architecture for multiple heterogeneous mobile robots in [72, 74]. The architecture was an extension of the behaviour-based systems described previously in section 2.3.1. A number of high-level behaviour sets compete to perform tasks. The concept of motivational behaviours was introduced as a mechanism to choose between these high-level behaviours. Each motivational behaviour had a number of inputs and one output, as shown in figure 2.25. The output was the activation level of the corresponding behavioural set, which was activated once a pre-defined threshold was passed. All the other behavioural sets in the system were inhibited. The inputs ranged from sensory readings to inter-robot communication (broadcasting of state information). Internal motivations were also a factor on the output. Impatience was used, to encourage individual robots to perform a task that had not been undertaken by any other robot. However, a robot had the ability to override the inhibitory signal from another robot, if a task assigned to that other robot was not being completed to a certain level, i.e. the robot had stalled. Conversely, robots also monitored their own progress. If they did not think they were completing a task to the required level, they would give up and attempt a different task. The system was deployed on a mock hazardous waste cleanup task (similar to the forage problem discussed in section 2.2). The system was observed to perform as designed.

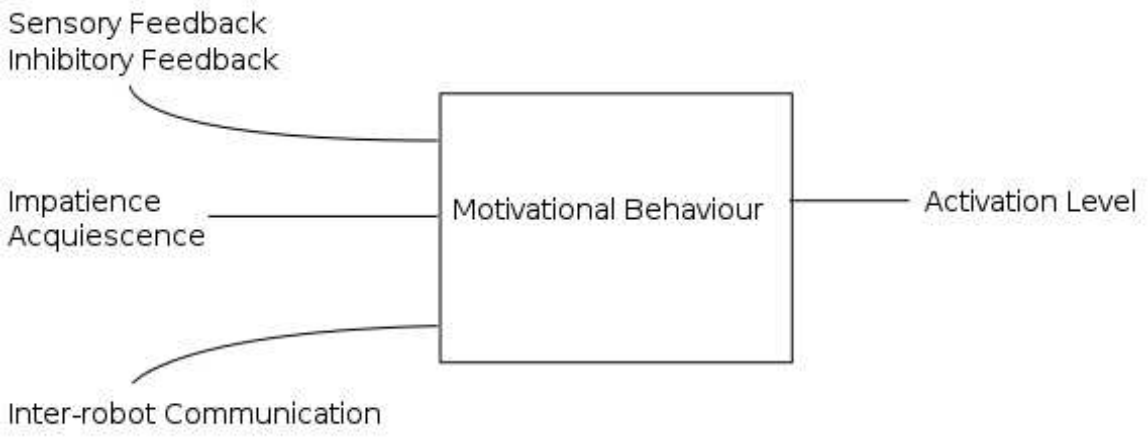


Figure 2.25: The behavioural sets are activated depending upon feedback, communication and state inputs.

L-ALLIANCE [72, 73] is a further extension to the behaviour-based approach. The system improved upon the efficiency of the robotic team's performance in the waste disposal task, whilst maintaining the team's fault tolerance. This was achieved by each robot assigning a metric to how well it could complete a task and how well other robots could complete their current tasks. The metric that was calculated was the mean completion time of the last five tasks. Knowing when robots should become impatient with the performance of teammates or themselves was a major issue in terms of performance. Three strategies were implemented; the first only required the robots to use their own performance measures. The second required the robots to compare their performance against the best robot they knew of within the group. The third strategy required robots to use the performance measures of other robots within the group. In the first strategy, better robots took over the tasks of worse robots only after the worse robot had decided to give up the task after a "fair" attempt. In the second strategy, the better robots took over tasks from worse robots; these worse robots gave up performing a task once a better robot was available. In the final strategy, tasks were completed on a first come, fist served basis. With better robots only taking over a task once the current robot decided it was going to fail.

Three methods of choosing which idle task to complete next were also investigated, these were: longest first; shortest first; and random selection. Simulation experiments were conducted with varying numbers of robots, degrees of robot heterogeneity, and tasks. The longest task first approach was quickly dismissed as it led to robots with the worst ability attempting tasks. The second strategy (let the best robot win) was the most successful strategy throughout the experiments, apart from the cases where the robots are only mildly heterogeneous, and do not assign progress when carrying out a task, and the shortest idle task is attempted first, in which case the third strategy (give robots a fighting chance) performed the best.

2.5.7 Critical Analysis of Multi-robot Systems

As discussed in section 2.5.1, robots that are members of *unaware* systems act as individuals and have no concept of 'the team'. No explicit communication or co-

ordination occurs between members of the system. Robots make decisions based upon the positions of other agents (team members or other elements of the system). In the most general case, this being a decision of "something's there, so I can't go there and must go somewhere else". The solution to a task is not achieved as a group — it is achieved as individuals. Solutions are more or less always sub-optimal. *Unaware* systems are often the benchmark systems used by multi-robot systems researchers to compare the effectiveness of another system's communication and co-ordination against. Indeed, this is the case in this thesis. The non-sharing system described in chapter 4 is of the *unaware* type.

Section 2.5.2 describes a number of *aware, not co-ordinated* systems. A robot in these systems can distinguish other team members from other elements of the environment. However, there is still no explicit communication or co-ordination. These systems act in a similar fashion to the *unaware* systems, but the recognition of team members allows the system to reduce the amount of interference between team members during a task. *Aware, not co-ordinated* systems are reliant upon their 'team member detection' sensor. If this fails, the system reverts to an *unaware* system. False positive and false negative team member recognition can also be an issue, both of which affect the performance of the system in the given task. *Aware, not co-ordinated* systems often appear in simulated environments where the reliability of the 'team member detection' sensor can be guaranteed.

Weakly co-ordinated systems, as discussed in section 2.5.3, share the properties of the above systems in that they do not use any form of explicit co-ordination. However, they do employ the use of implicit co-ordination. Team behaviours are an emergent property of local interactions within the system. *Weakly co-ordinated* systems have the benefit of being simple to design due to the lack of a complicated co-ordination strategy, coupled with the additional benefit of the designer having knowledge of how the system will react in an environment *a priori*. Due to this limited co-ordination, the level of complexity of the task being undertaken is also limited in nature.

Section 2.5.4 describes the *strongly co-ordinated, strongly centralised* type of multi-robot system, in which a group of robots is controlled or led by a designated 'leader' robot or by a remote PC. The 'leader' agent makes sure that the system is always

moving towards a solution and gives the designer of the system yet more assurances of the behaviour of the system in a given environment. If undesirable behaviour is occurring, the 'leader' agent can be used to change this behaviour. The main problem with a *strongly co-ordinated, strongly centralised* system is its centralised nature. If the 'leader' agent fails, the system will, at best, reverting to a *weakly co-ordinated* system. At worst the whole system will fail.

Strongly co-ordinated, weakly centralised systems, as described in section 2.5.5, unlike their *strongly centralised* peers do not assign 'leader' agents *a priori*. Instead, the assignment of 'leaders' is task dependant. Typically, robots within the system use a bidding method to discover which robot is more capable of performing a given task. Robots that are unable to form a bid, perhaps due to systems failure, act as individuals. The cost is, of course, a more complex co-ordination strategy. However, there is the benefit of increased tolerance to failure (especially in homogeneous groups).

Finally, *strongly co-ordinated, distributed* systems, as described in section 2.5.6, do not assign any agent as a 'leader' — each robot within the system makes its own informed decisions. Typically, each robot will have a number of individual goals to achieve as well as a number of team goals. By maximising their individual achievements robots within the team aim to maximise the team's achievements. The cost of such a system is the complex co-ordination strategy coupled with a typically high bandwidth communications protocol. However, allowing each robot to make its own decisions throughout the task increases the fault tolerance and the robustness of the system dramatically, if the robots are presumed homogeneous. If heterogeneous robots are employed, this type of system reduces the amount of wasted resource, as robots are aware of what tasks they can and cannot complete.

2.6 The Potential Field Method

The potential field method is an analogy of the movement of electrically charged particles in free space. Particles of equal signs are repulsed by one another; particles of opposite signs are attracted to one another. Noteworthy robotic systems that have implemented the potential field method, that have not already been discussed above,

will now be discussed. This section will conclude with a discussion of the known limitations of the potential field method, proposed solutions from the literature and close with a critical analysis.

The potential field method was introduced in Khatib's seminal work on obstacle avoidance [54], in which he took what was commonly thought of as a high level planning problem and turned it into a low level real-time control problem. He also envisioned a hybrid class of robotic system, in which a high level planner would generate a global strategy, whilst a low level controller would produce the commands to reach the goals set by the high level planner.

In the robot football domain Tews *et al.* [88] used a centralised system to plan the actions of team members. Only three actions were implemented within the system: *go to destination*; *kick the ball*; and *halt*. The central planner examined the state of the game and made decisions based upon the robot's perspective. Potential fields were used to determine the locations where the robots would carry out these actions. One robot was chosen to kick the ball to a desirable location; the others moved to useful locations. Both of these types of actions were based upon a potential field evaluation of grid locations. The precise action depended upon the state of the game i.e. attacking or defending. The potential field was constructed based upon objects within the physical field. For example, a base field which was biased towards the opposition's goal was used to encourage the robots to attack, and a robot's personal region was used to discourage interference from other team members. Choosing desirable locations to kick the ball to, or move robots to, were calculated by the summation of a number of potential fields. Experiments showed that the system performed better than a system that did not employ any co-ordination.

Also in the robot football domain, Damas *et al.* [36] used a potential field method to enable a robot to dribble the ball. The potential field was "stretched" in the direction of motion, in the direction of the *x-axis* in a non-holonomic robots case. As shown in figure 2.26, a non-holonomic robot is restricted to movement along the *x-axis*. As the ball was an attractive force within the potential field, the robot moved towards it. In order to maintain contact with the ball (dribble) the inertia and friction forces exerted on the ball had to be balanced with the torque generated at

the contact point. The system was successfully implemented on the IsocRob team in the RoboCup middle-size league.

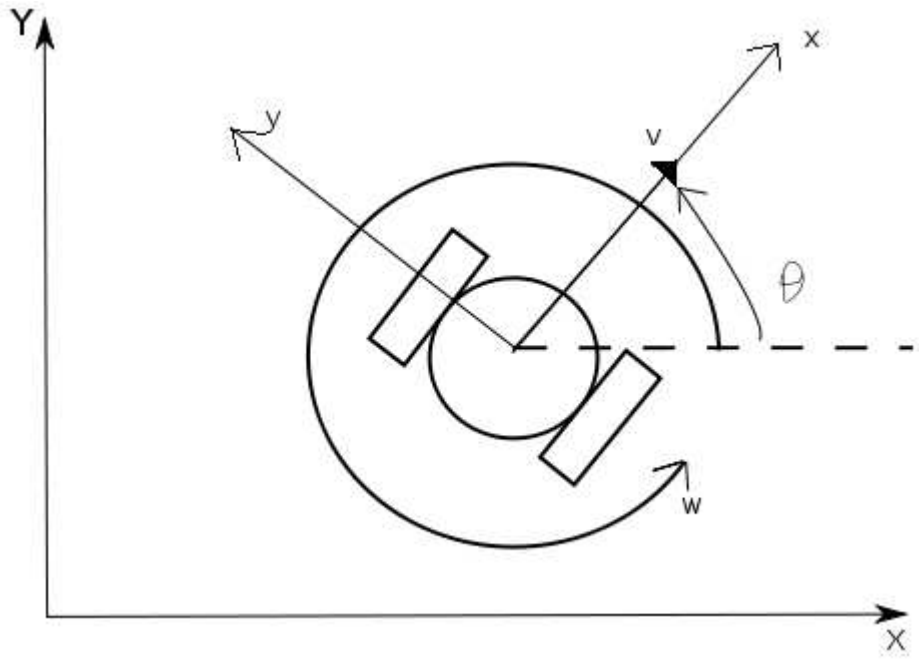


Figure 2.26: The robot cannot freely move along the y-axis, movement along the x-axis is a necessity. Figure adapted from [36].

Pathak *et al.* [76] applied the potential field method to mobile robot path planning. Given a starting location and a goal location a planner algorithm outputted a string of overlapping bubbles joining the start and goal locations. The size of each bubble represented the free space available between obstacles. The shortest path was calculated through these bubbles, and was guaranteed to encapsulate the robot's own current bubble. An example of the bubble path planning is shown in figure 2.27, where a robot safely navigates past two obstacles to a desired location. The robot's own bubble was moved in discrete steps towards the direction provided by the planner, whilst avoiding unexpected obstacles i.e. other moving entities within the environment. Hence, motion was achieved through the switching between two controllers. These controllers were defined as local potential fields, the first attracted the robot to the centre of its bubble, and the second attracted the robot to the desired orientation. The system was implemented on real robots in a laboratory setting in a

exploration type task.

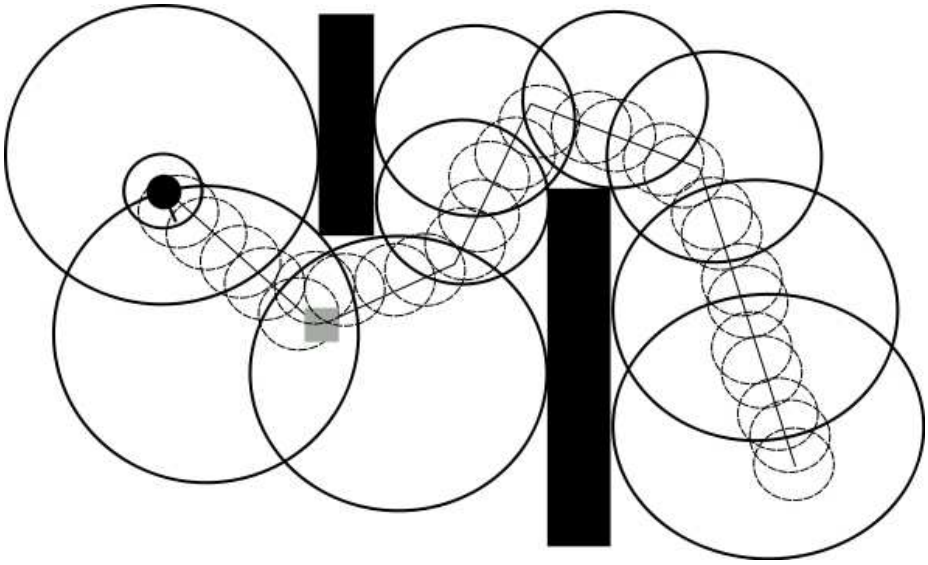


Figure 2.27: Obstacle free path from start location to goal location. The black rectangles are known obstacles, the grey square is an unknown obstacle. The large circles represent the free space between known obstacles, the small circles are steps generated to get to the goal location. Figure adapted from [76].

Zavlanos *et al.* [96] used artificial potential fields to maintain the *connectivity* constraint of a group of twenty mobile agents (represented as a graph). The agents maintained a formation whilst following a leader, and avoiding collisions with one another. The leader agent was used to steer the other agents throughout the simulated environment, and as such it was not affected by the potentials of the other agents. However, it was attracted to a rendezvous point within the environment, to encourage motion. The follower agents were repulsed by other agents, in order to avoid inter agent collisions. A repulsive force also existed between agents and an imaginary obstacle. This imaginary obstacle represented the *connectivity* constraint of the system. An example is shown in figure 2.28, in which each non-leader robot is attracted to the leader robot. Once the non-leader robots are close enough to the leader robot, the potential field governing formation becomes dominant. Finally, once in formation, the group moves towards the desired target. During simulated experiments, agents avoided imaginary obstacles which enabled them to stay in formation.

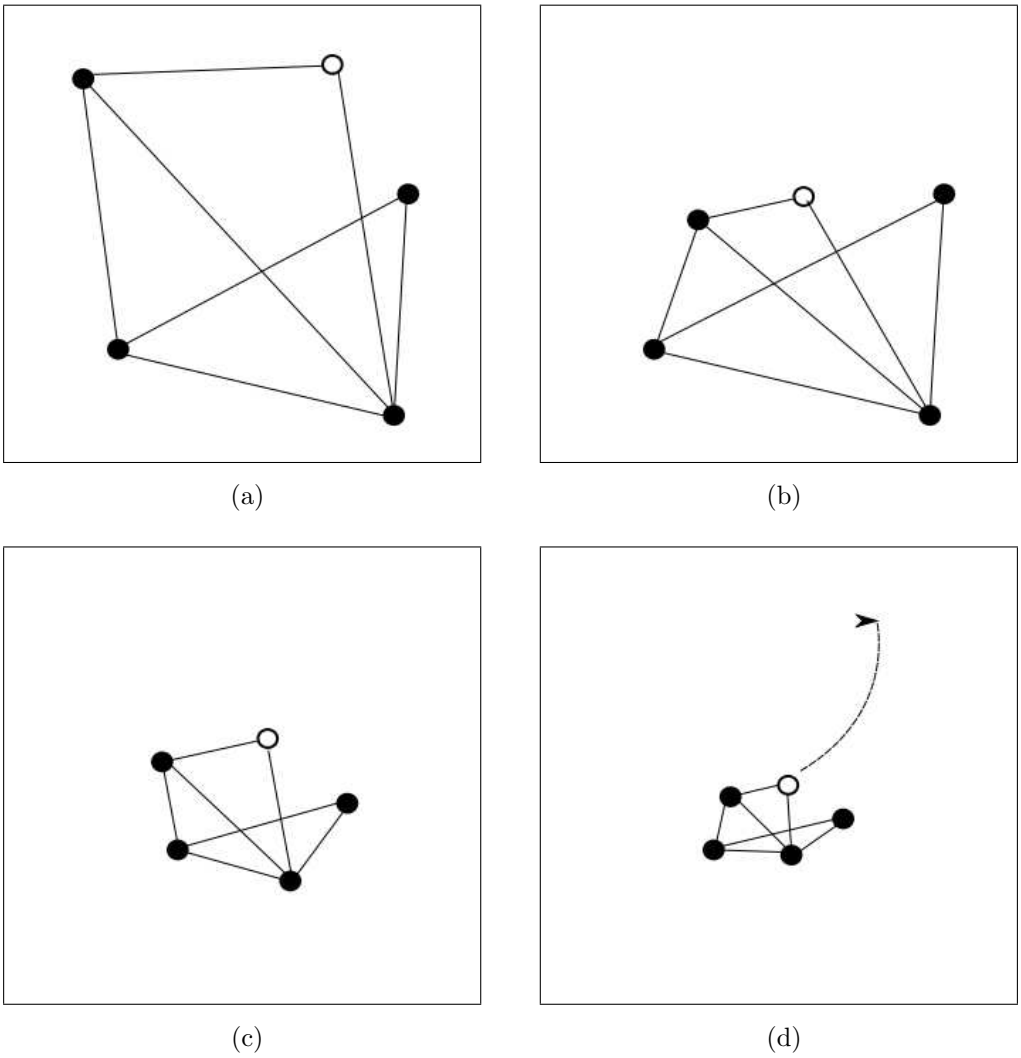


Figure 2.28: The time steps of 5 agents first moving into formation, then moving towards the target as a formation: Note that we have one leader agent (empty circle) and four follower agents (filled circles). The solid lines are the graph connections. The dashed line is the leader robot's path to the rendezvous point (arrow head).

All of the literature above has concentrated on two dimensional potential fields.. The reason for this is that two dimensional potential fields are by far the most commonly used within the robotics community, as the majority of robots are restricted to travel in two dimensions. Potential fields can be constructed in three dimensions for robots can travel in three dimensions. However, the basic premise is the same,

particles of the same sign repel one another and particles of different signs attract one another.

2.6.1 Limitations of the Potential Field Method

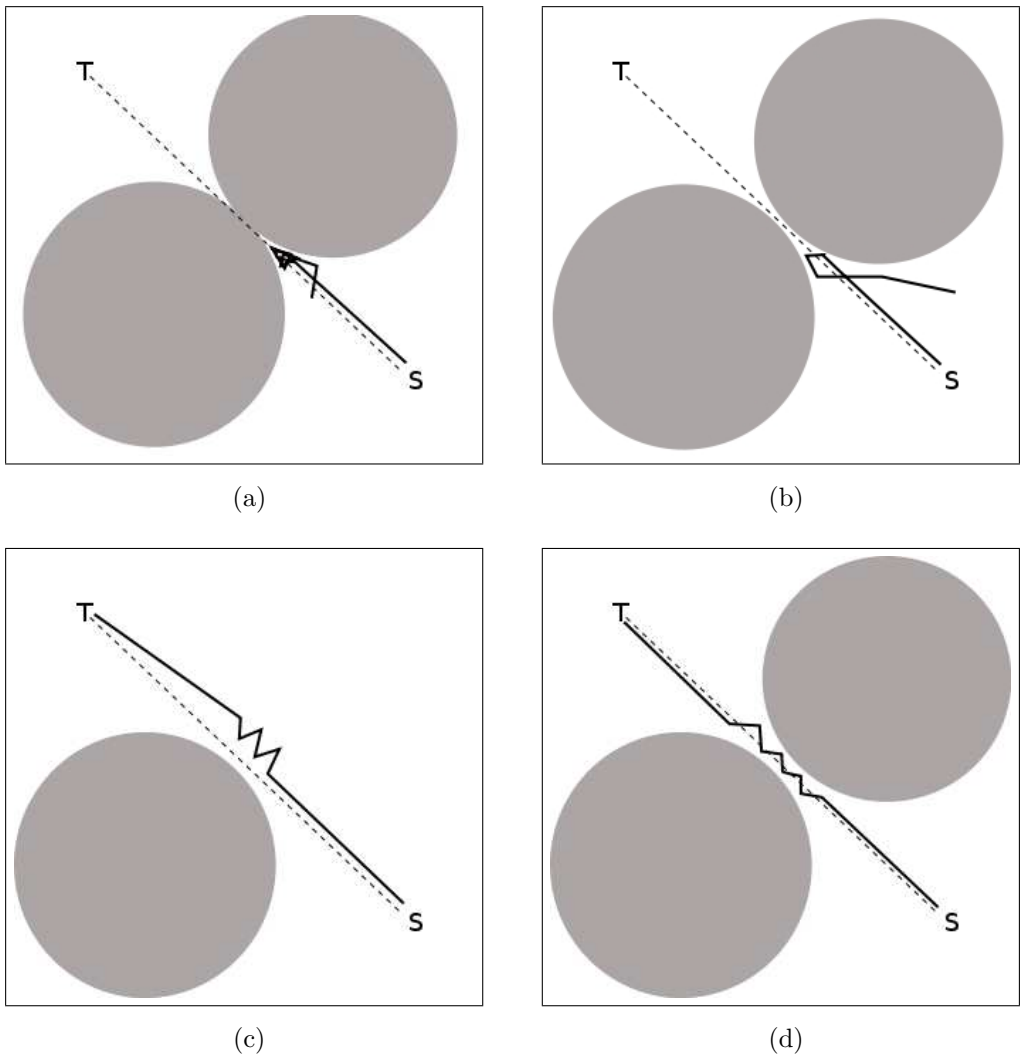


Figure 2.29: (a) Trap. (b) No Passage. (c) Oscillation in presence of obstacle. (d) Oscillation in narrow passage. The grey circles are obstacles. The dashed line represents the straight line path from the start location (S) to the target location (T). The solid line represents the path taken by the robot.

Koren *et al.* [55] identified four major limitations of the potential field method:

1. *Trap Situations*: These are situations in which cyclic behaviour between local minima occurs, as shown in figure 2.29a. Trap situations occur when robots run into dead ends e.g. U shaped obstacles, where the robot's range sensors pick up objects to the front and sides of the robots. The potential field generated in such a case forces the robot into a cyclic behaviour. In these situations, it is often necessary for some global recovery mechanism to intervene, which often results in a sub-optimum solution, but at least the robot is no longer trapped.

2. *No Passage*: Closely spaced obstacles bar passage, as shown in figure 2.29b. No passage situations arise when a robot attempts to travel between two obstacles. However, the obstacles are sufficiently close enough together that the potential field that is generated forces the robot away from the passage, so that in essence, the two objects are treated as one large object. Again the results are sub-optimal.

3. *Oscillations in the presence of obstacles*: This type of oscillation occurs when a robot passes an object — the resultant potential field generated forces unstable motion whilst passing the object. This results in a sub-optimal solution. An example is shown in figure 2.29c

4. *Oscillations in narrow passages*: This type of oscillation occurs when a robot is travelling down a narrow passage. The resultant potential field causes the robot to oscillate from one side to the other. This is a result of the robot being repulsed by the objects on either side of it sequentially, until the robot has escaped the passage. Again, the results are sub-optimal. An example is given in figure 2.29d.

Ren *et al.* [81] proposed a general solution to the inherent oscillation problems (see figures 2.29c and 2.29d) in the potential field method. The modified Newton's method was implemented to alleviate oscillations by approximating the navigation function with a quadratic form, compared to the traditional gradient method which contains only linear information. From observations of simulated experiments, it was clear that the gradient method encountered difficulties. Unlike the modified approach, whose

tolerance level was higher than the gradient method and so could navigate through the narrow passages. The gradient method also produced a much higher failure rate.

Each of Arkin's motor schemas [1] (described in section 2.3.1) output was a velocity vector, this was the point in the potential field of the entire environment in which the robot currently resided. Each schema only had to compute this one point within the potential field in order to create the velocity vector. An additional *noise* schema was incorporated into the system to help the robot avoid oscillations between local maxima and minima. This solution was successfully implement on a group of robots which completed a formation control task [7].

2.6.2 Critical Analysis of the Potential Field Method

As detailed in section 2.6.1, the potential field method has a number of well known limitations. However, the method is still widely used in the robotics community due to its simplicity and elegance. The limitations of the method generally mean that systems that employ it also employ some type of recovery mechanism to escape from local minima traps. The use of these recovery mechanisms and the method's susceptibility to oscillations leads to sub-optimal solutions. The potential field method can be assigned to the reactive robotics architecture described in section 2.3.1. As such, the sub-optimal nature of the solutions is expected. The potential field method therefore takes advantage of the same benefits as any other reactive system in that it has no reliance on information gathered *a priori*.

2.7 Related Work

In this section the literature most related to the system presented in chapter 4 will be discussed.

At the individual robot control level, all of the literature discussed in section 2.6 is related to the control system used by the potential field sharing method described in chapter 4. The major difference is that in the potential field methods from the literature, a vector sum is calculated for the individual robot from a number of potentials fields created by obstacles and the environment, the resultant vector provides the

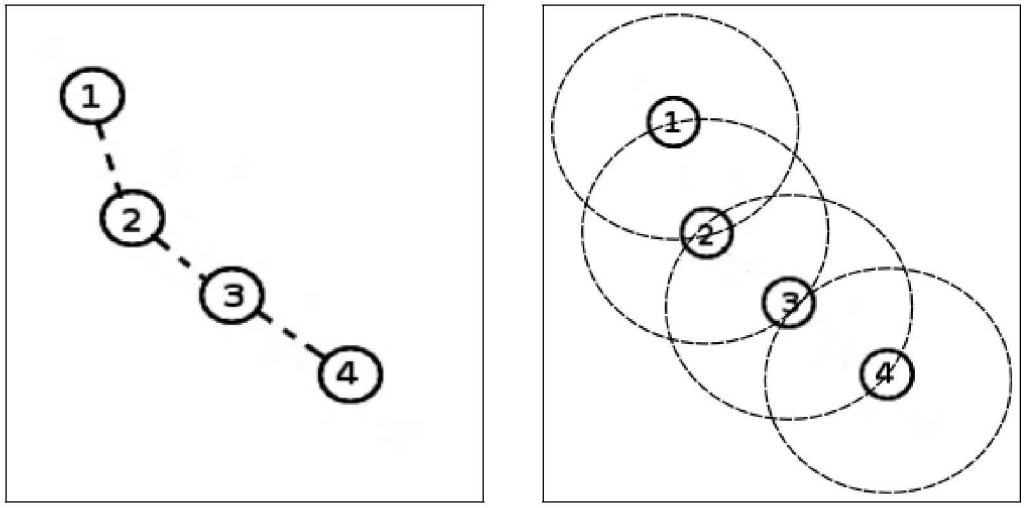
motion dynamics of the robot. In the system described in chapter 4, a resultant force is calculated for each of the ultra-sonic sensors on the robot. The motion dynamics are provided by an action selection method, based upon these resultant forces.

Howard *et al.*'s mobile sensor network (as described in section 2.5.3) solved the coverage problem through the emergent property of the system [50]. This system is another potential field based approach. Robots were repulsed by one another and other obstacles within the environment. The same can be said about the shared potential field method implemented in this thesis. However, it is to a lesser extent, because in the potential field sharing system, teammate recognition is not possible. Section 8.5 describes how the system could be extended to allow such recognition, in which case the behaviour of the two systems could be more alike.

In the potential field sharing system described in chapter 4, the concept of local groups is introduced. This process of sharing information within a finite set of robots is similar in concept to "dynamic robot networks" in Clark *et al.* (see section 2.3.3) but instead of sharing trajectories, potential field information is shared. Another noteworthy difference is that whereas Clark *et al.*'s system forms non-overlapping networks of robots, the potential field sharing method produces groups of overlapping robots. Figure 2.30 gives an example. Whereas in 'dynamic robot networks', robots 1 to 4 form an non-overlapping network, in the potential field sharing method, 4 overlapping local groups are created: robots 1 and 2; robots 1, 2 and 3; robots 2, 3 and 4; and finally robots 3 and 4.

Robots in Sugawara *et al.*'s system (as described in section 2.5.1) only interact once a puck is discovered [87]. The sharing potential field system (see chapter 4) is similar in a fashion, as interaction only occurs between robots when they enter one another's local group radius. In Sugawara *et al.*'s approach, the interaction causes the 'search' robots to move in the direction of the discovered puck, whereas in the potential field sharing method robots are also encouraged in a direction, but in this case in the direction of least resistance. It is also believed that the performance of the potential field system, as with the system in Sugawra *et al.*, is related to the puck/target distribution.

Vail *et al.* also implement co-ordination though "shared potential fields" in the



(a) (b)

Figure 2.30: a) Robots 1 to 4 form a dynamic robot network. The dashed lines represent the network. b) 4 local groups are created: group 1 — robots 1 and 2, group 2 — robots 1, 2 and 3, group 3 — robots 2, 3 and 4, group 4 — robots 3 and 4. The dashed circles represent the local group radius used to define local group members.

CMPack team that competed in the RoboCup Sony legged league, and indeed won the competition in 2002 [91, 82]. However, the system is significantly different from the one presented in this thesis. Each robot maintains its own local world model that contains the robot's position and heading, the ball's location, teammate positions, and opponent positions. The robot's position and heading is updated by its own localisation module; the opponents positions are updated solely by the robot's own vision module. Teammate positions are updated solely from the shared world model, and the ball's location is updated from a combination of the vision module and the shared world model. The shared world model is completely distributed, with each robot having its own local copy; each copy is updated by continuous broadcast information from teammates. Again, Vail *et al.*'s system differs significantly here, as the broadcast information consists of positional information (for the robot and ball) and estimates of the uncertainty in those positions, unlike in the system presented in this thesis, which, in effect, broadcasts pose and potential field information. Also, in Vail *et al.*'s system, every robot within the system receives the broadcast information,

unlike in the system presented in chapter 4 in which broadcast information is limited to local groups. Where the systems are similar is that both contain local and shared models of the world, that are distributed throughout the system.

2.8 Summary

In this chapter, a number of common robotic problems were defined, ranging from the low level path planning problem to the high level robot football problem. An overview of the three major robotic architectures has been given. Reactive systems are not given any knowledge of the environment *a priori* and react to the environment through the information gathered by sensors in real-time. Global behaviours are often an emergent property of the system. Deliberative systems are given as much information as possible prior to the task being executed. Near optimal solutions for sub tasks are computed *a prior* and executed sequentially throughout the task. Hybrid systems attempt to gain the benefits from both reactive (robustness to unexpected events) and deliberative (near optimal solutions) systems, whilst minimising the known problems associated with each system. A critical analysis of the robotic architectures was also conducted.

A number of multi-robot taxonomies were discussed and critically analysed. Farinelli *et al.*'s taxonomy was chosen as the taxonomy that would be used to categorise the multi-robot systems within the literature review, as well as the new multi-robot system presented in chapter 4. This taxonomy splits various robotic systems into sub-categories based upon the level of communication and co-ordination within the systems. The most simple (in terms of the level of communication and co-ordination) category is *unaware*; in these systems, robots are totally unaware that they are within a team of robots, and no explicit communication or co-ordination of any kind occurs between team members. These systems rely upon implicit communication such as changes to the environment made by one robot affecting the decisions made by another robot. The next step up is, *aware-not co-ordinated* systems, although aware of the existence of other team members, they again rely upon implicit forms of communication. Like their *unaware* counterpart's behaviours they are often an emergent

property of the system. *Weakly co-ordinated* systems are aware of team members, and as the name suggests, do perform some low level explicit communication; usually the communication is limited to task relevant situations. For example, a robot may broadcast the message "I have found a puck" in a forage type scenario. This would enable the other robots in the team to either move to the robots location in search of other pucks (if they assume a low puck distribution within the environment), or stop the task all together if all pucks have been found. The last three sub-categories are all *strongly co-ordinated* — explicit communication is used heavily throughout the assigned task. They do, however, differ in terms of architecture: *strongly centralised*, *weakly centralised* and *distributed* — the difference being where the decisions are made. In centralised systems, a remote PC or robot "leader" makes the decision on how to solve a given task; if the whole task is solved by this central system it is said to be a strongly centralised system. If only a small amount of the decision process is taken at the centre, then the system is said to be weakly centralised. As the name suggests, a distributed system has no central controller and all decisions are made by the system members on an individual basis.

A number of robotic systems that employed the potential field method were also discussed. The potential field method uses the concept of artificial forces which are produced by all entities within an environment. When two entities produce a force with the same sign, they are repulsed by one another; if the signs differ they are attracted to one another. This concept allows mobile robots to explore an environment whilst avoiding obstacles. The concept is analogous with electrostatic forces and magnetism. Known limitations of the potential field method and solutions available in the literature were discussed. A critical analysis of the potential field method was presented, which concluded that despite the known limitations of the system, the simplicity and elegance of the system makes it a popular choice amongst researchers.

A detailed review of the of other systems in the literature that are related to the system presented in chapter 4 is presented. The innovations made in this thesis are highlighted. These innovations include the use of shared potential fields as both a method for communication and co-ordination.

Chapter 3

Robotic Hardware and Software

3.1 Introduction

In this chapter, the type of robot used throughout the project outlined within this thesis will be introduced — the Miabot Pro. Brief technical specifications will be given for each device used in our experimentation. The laboratory setup will also be described. This includes the overhead camera system used to track the Miabots throughout the environment. The software developed to control the Miabot Pro will be introduced. The software architecture, in which a plug-in for the Miabot Pro was implemented, will be briefly described. Configurations and settings used throughout the experimental study will also be briefly described. The current system architecture will be outlined and possible future architecture(s) discussed. The chapter will conclude with a summary of the experimental setup used throughout this thesis.

3.2 Miabot Hardware

3.2.1 Base Module

What follows is only a brief introduction to the Miabot Pro class of robots. More information can be found at http://www.merlinrobotics.co.uk.

As shown in figure 3.1, the base module of the Miabot Pro consists of a differential steering drive, a processor board and a Bluetooth communications board. The differential steering drive has an optical resolution of $0.04mm$ and a reported maximum

speed of $3.5m/s$ (approximately $1m/s$ in a straight line). The processor board has $64kb$ of programmable memory and supports up to eight IO devices. The Bluetooth communications board transmits in $2.4GHz$ range at $11.5kb/s$. The large and accumulative errors within the Miabots encoders makes them an unreliable source of positional information and so an overhead tracking system is used to position the Miabots within the environment, as described in section 3.2.4.

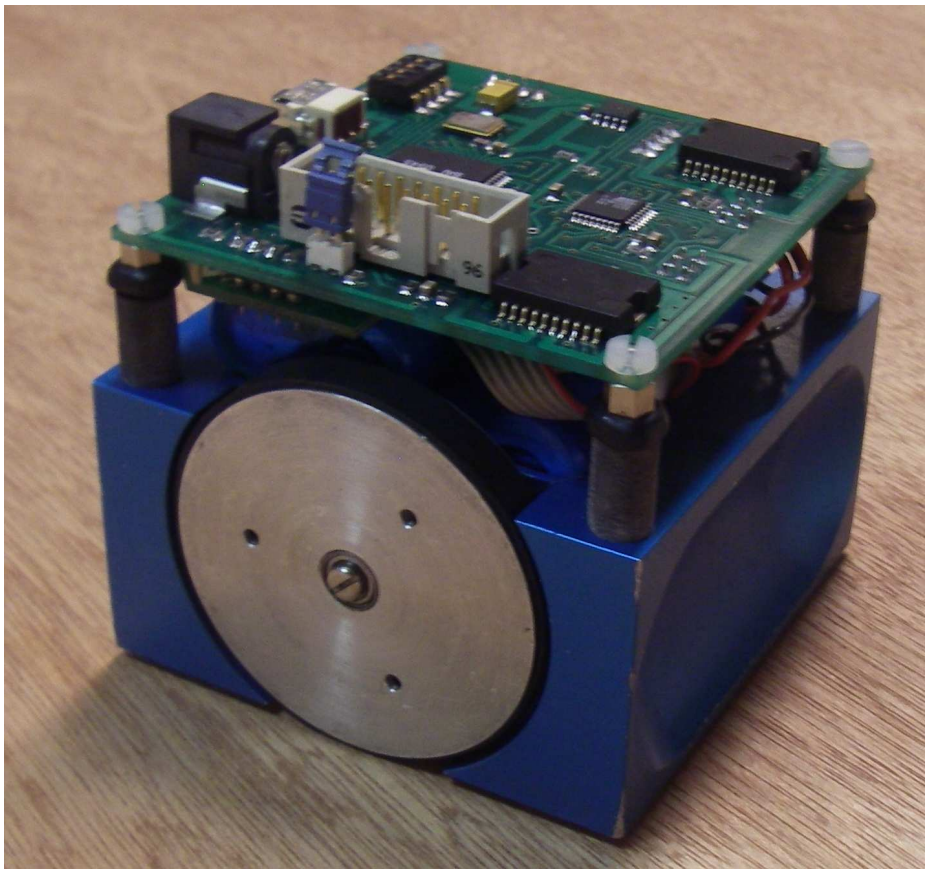


Figure 3.1: The base module contains a differential steering drive and a bluetooth communications module.

All devices transmit and receive information through a single communications bus on the processor board. Devices are assigned to one of two categories:

- *master*: Devices which subsume control over the bus from slave devices.
- *slave*: Devices which differ control of the bus to a master device.

The differential drive is a *slave* to this single bus.

3.2.2 Ultra-sonic Range Finders

As shown in figure 3.2, the ultra-sonic sensor array supports 8 individual sensors at 45^o intervals, giving a 360^o field-of-view (FOV). Each individual sensor has a FOV of approximately 30^o, leading to small gaps of approximately 15^o between each sensor. The affect of these blind spots on sensor coverage decreases the further away obstacles are within the environment. If obstacles are of small enough dimensions, they may not be detected by the ultra-sonic array. The probability of such an oversight increases the closer such an obstacle is to the array. In the experiments within this thesis, all obstacles within the environments are of big enough dimensions as to be detectable within the entire range of the ultra-sonic array. The sensors have a reported range of $6m$, although only ranges of $2.5m$ have been observed due to environmental conditions. The ultra-sonic sensor module is a *slave* to the single bus on the processor board.

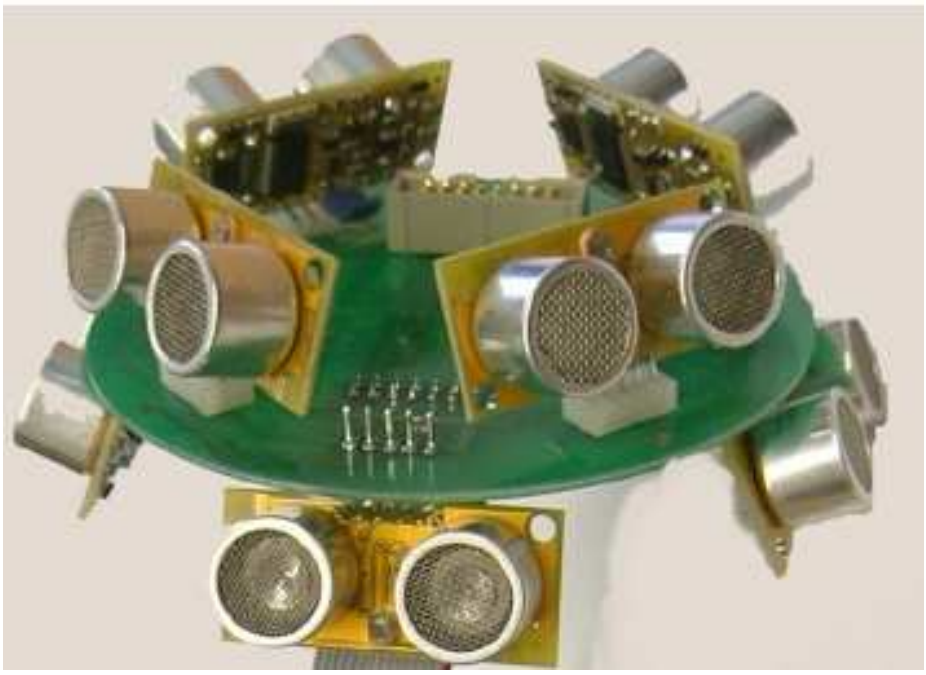


Figure 3.2: Contains eight separate transceiver and receiver pairs, giving approximately a 360 degree view.

3.2.3 Blob-finder

As shown in figure 3.3, the blob-finder module has its own processor board, which includes its own embedded blob-finding algorithm. The blob-finder tracks up to 8 different, RGB defined, blob values at $30fps$ (frames per second). The tracked image resolution is 88×144 pixels. The blob-finder module is the *master* of the single bus on the Miabot processor board. This can be an issue if blob data is abundant within the environment, as the blob-finder module will effectively lock the bus, not allowing possibly critical information to be communicated (e.g. ultra-sonic data). As such, the environment is kept as blob free as possible. Only a single blob is defined to the module and this only occurs once within the environment (of course, it is not wise to assume that this will always be the case, i.e. light reflections, but the effect can be minimised).

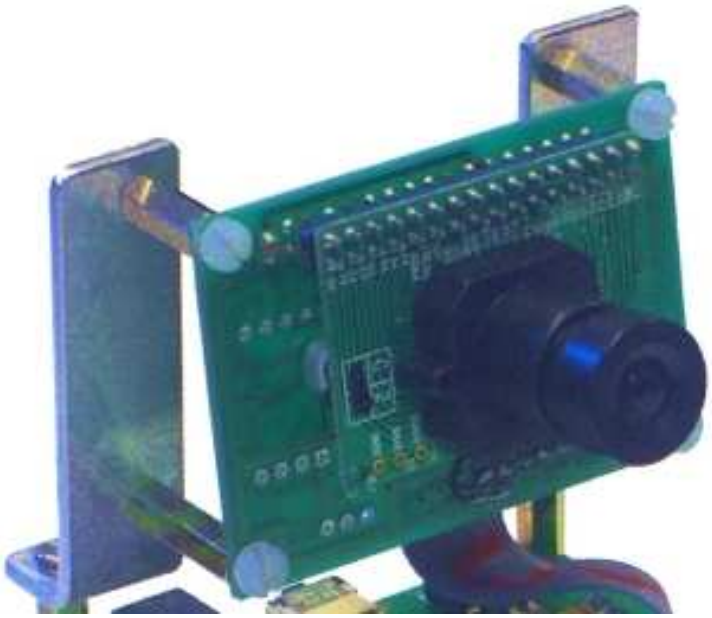


Figure 3.3: Can track up to eight different RGB defined blobs at a time.

3.2.4 Robot Village

The Miabot tracking system (supplied by the Miabot manufacturer), shown in figure 3.4, consists of two webcams with an image resolution of 640×480 pixels. Each of these connects to its own PC via a fire-wire connection. Each of these PCs runs a local blob server, which track the robots in each camera frame. Each robot is crowned with a blob patch, as shown in figure 3.5. By finding the centres of the 3 blobs, the system can estimate the centre of the Miabot (the centre point between points 1 and 3 in the example) and by the use of trigonometry it can estimate the orientation of the Miabot (the angle highlighted in the example). One of these PCs also has a global blob server which takes the local co-ordinates from each local blob servers (via a TCP/IP network) and constructs a global co-ordinate for each robot. As is shown in figure 3.6, the *x-axis* of the local blob servers overlap. This is intentional, as this overlapping avoids any blind spots between camera frames. However, the global co-ordination system needs to take into account this additional translation. The distance of the overlap is known *a priori* to all experiments.

As can be seen in figure 3.7 the Player server and clients described in section 3.3 resided on another PC that communicates to the Miabots through a Bluetooth router, which allows TCP/IP communication over the Bluetooth RF-COMM (virtual serial communications) port. The PC also used TCP/IP to read the global blob servers socket to gain access to position information.

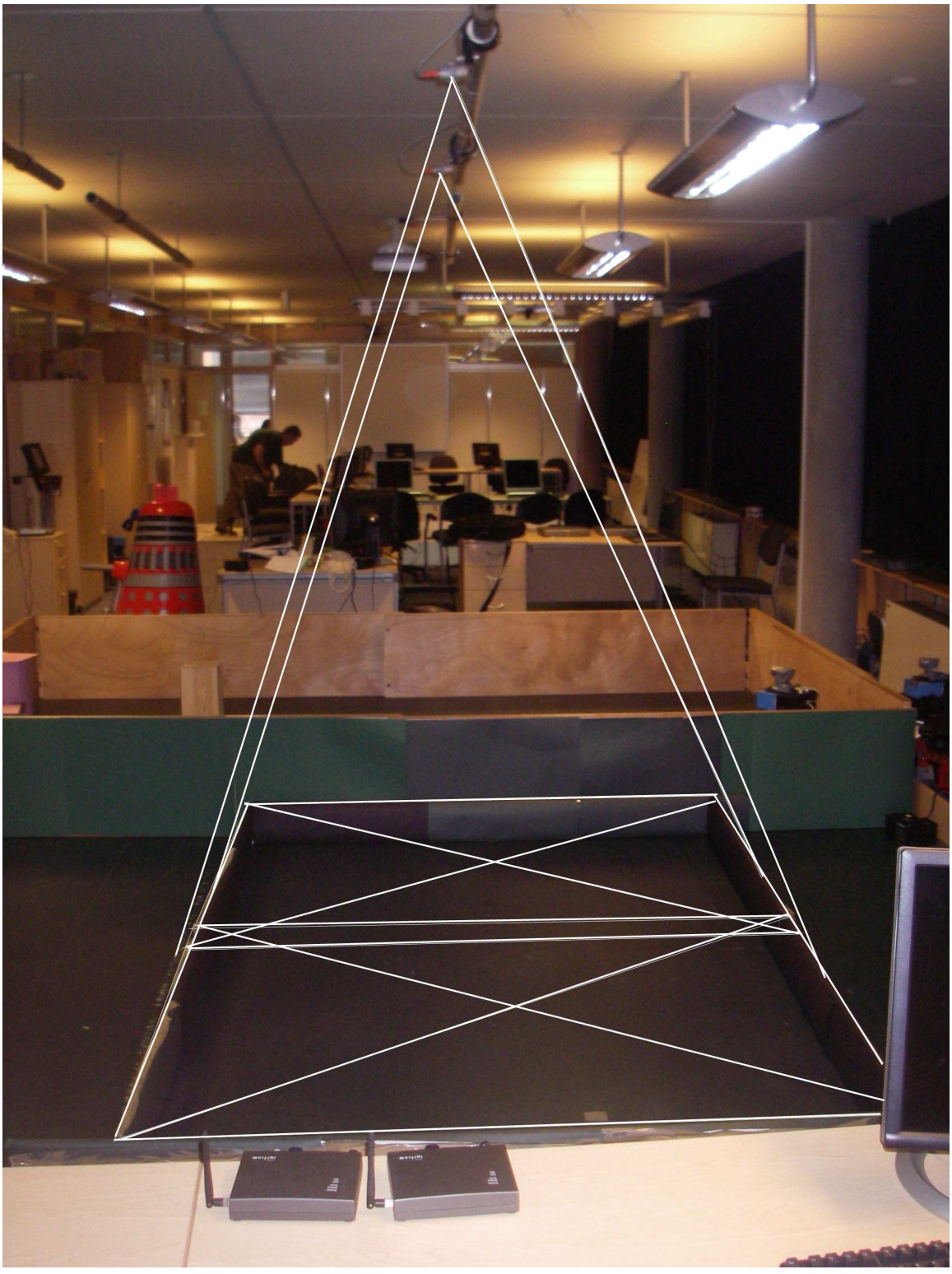


Figure 3.4: The approximate coverage area of each camera is shown with white lines.

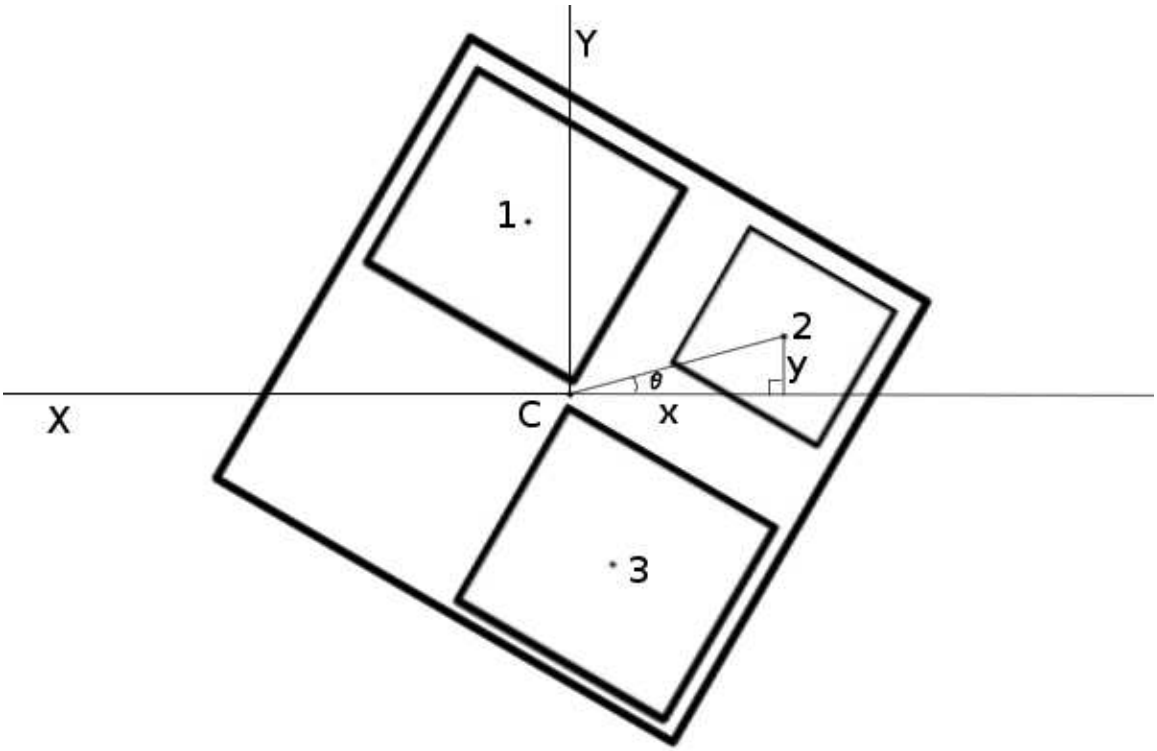


Figure 3.5: By finding the centre of each of the three blobs, the x and y position of the Miabot (at point C) can be discovered, as well as the θ orientation.

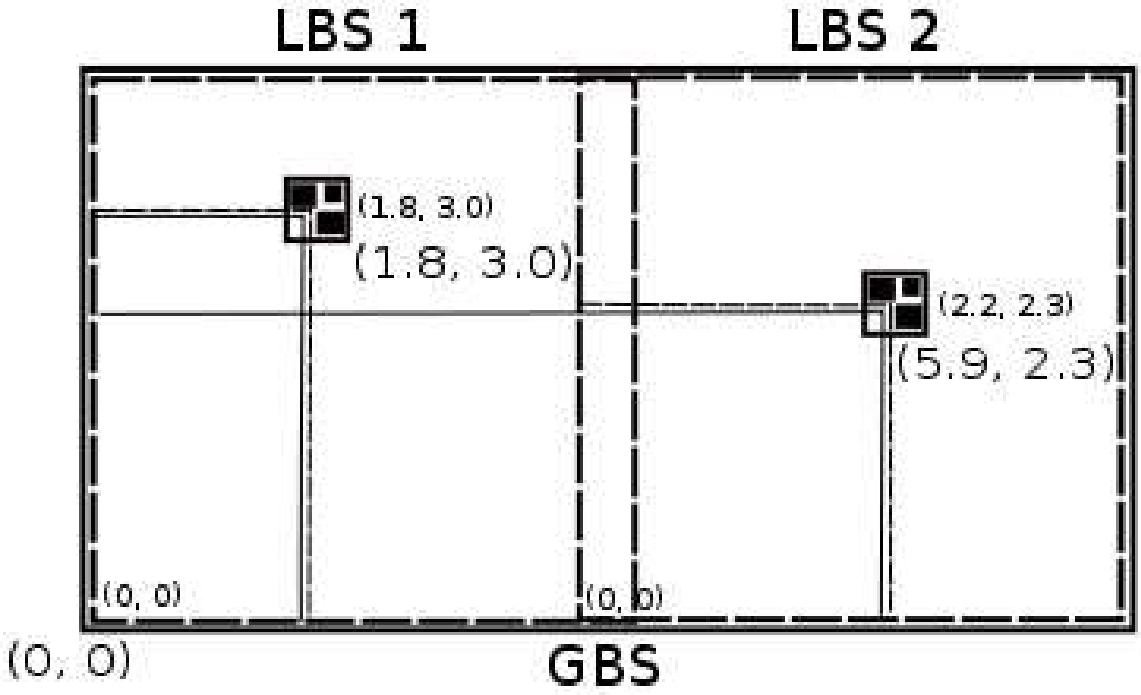


Figure 3.6: The co-ordinate systems from both of the local blob servers (LBS1 and LBS2) are joined together by the global blob server (GBS) to create a global co-ordinate system.

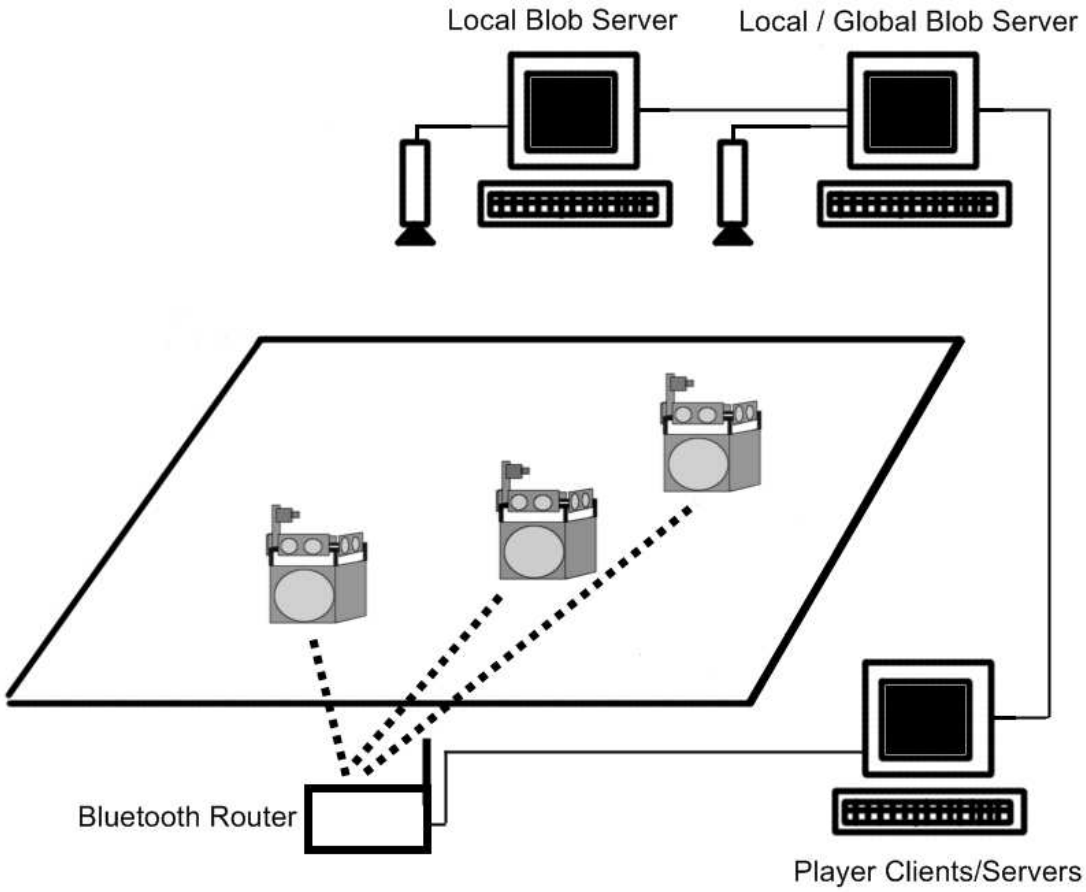


Figure 3.7: Two cameras are connected to two PCs via firewire. Both have local blob servers running, one has a global blob server. A third PC has the Player server/clients running and communicates to the Miabots via a bluetooth router. The whole system is a TCP/IP network.

3.3 Player/Stage

3.3.1 Introduction

This is only a brief introduction to the Player/Stage architecture — more information about the project can be found at http://playerstage.sourceforge.net/.

Figure 3.8 shows the basic Player/Stage architecture. The most important part of the architecture is the Player Server. It acts as the middleware. The Player server enables end users to write client applications, without any prior knowledge of the low level robotics, to communicate with either a 2d simulator (Stage) or a number

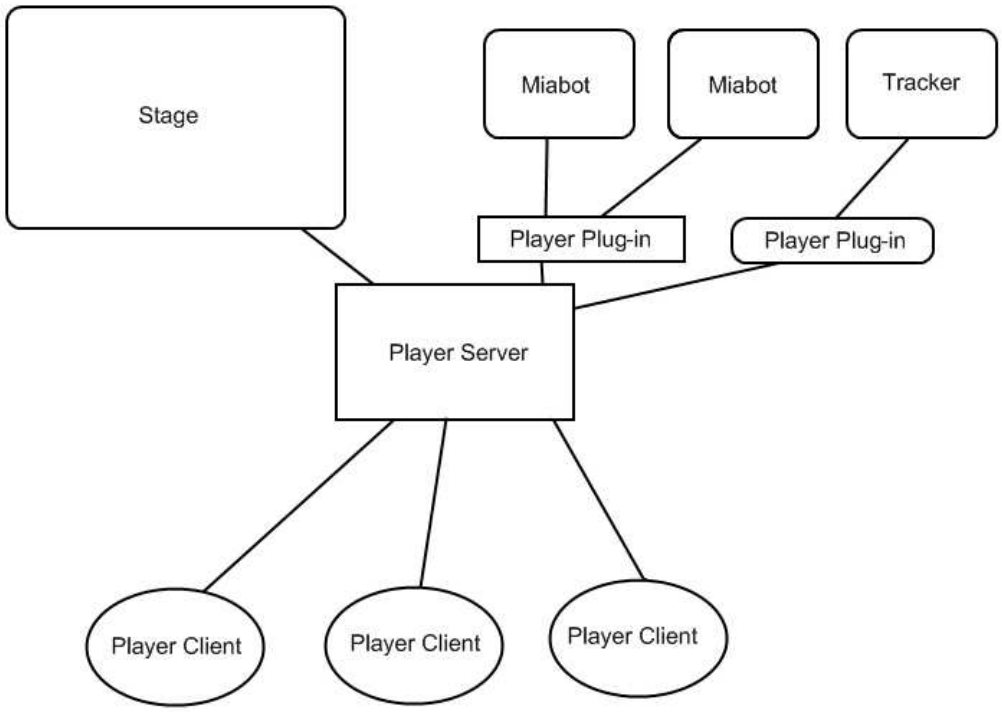


Figure 3.8: The player server acts as the middleware between the clients and the robots/simulation.

of known devices (a robot or individual a sensor/actuator). The Player clients use abstract device definitions that allow the same client to control any type of device known to the Player server. For example, instead of writing code to retrieve range data from a specific ultra-sonic sensor, the client's code uses a global 'get sonar range data' function. The Player server then looks to see what specific device is involved and communicates to it using the necessary low level commands, returning to the client the requested data. As new devices are developed all of the time, the developers of the Player/Stage architecture have implemented a plug-in scheme which allows users to add knowledge of devices to the Player server. When the server is queried about a device it does not recognise, it looks up its plug-in table to find the appropriate device. Similarly with Stage, if a user wants to simulate a new device they simply have to create a Stage model of the device. A Stage model is an abstract description of the

device based upon terms Player/Stage recognises. See section 3.3.3 as an example. As the Player/Stage architecture uses TCP/IP as its communication protocol, it can be distributed over a robotic system in a number of ways:

1. *Autonomous*: A server and client embodied on each robot.
2. *Remote*: A server embodied on each robot. A client(s) embodied on a separate computer.
3. *Local*: A server(s) and a client(s) embodied on a separate computer, with the robot(s) treated as a device(s) of the computer.

For the laboratory experimental setup in this thesis, a local approach is taken, as the Miabots do not have enough on-board memory to run a Player server or client. A server per Miabot was run on a separate computer, which communicated to the Miabots through the TCP/IP network. The computer also ran individual clients for each Miabot. In the simulation experiments, the same individual clients were used to connect to the Stage simulator (one Player server).

In the following sub-sections, both the Miabot plug-in (section 3.3.4) and the Tracker plug-in (section 3.3.5) were developed by the author. The wavefront plug-in (section 3.3.6) was developed by the developers of the Player architecture, and the Nearness Diagram (ND) plug-in (section 3.3.6) was develop by Minguez *et al.* as discussed in chapter 2.

3.3.2 Stage Validity

Gerkey *et al.* point out that no guarantee is given that the experiments run under stage are directly comparable with experiments run in the laboratory [46]. They point out that a client written for Stage will work on a real robot with little or no modification and vice-versa. This was confirmed in experiments carried out in this research as described in chapters 5 and 6.

As simulation was used in this project as a fast prototyping method (as discussed in chapter 5) a test of Stage's validity was conducted. These tests showed that, given

the same controller (the non-sharing system proposed in chapter 4), a simulated Miabot's actions were similar to those of a real Miabot's. Figure 3.9 shows the path taken by a simulated Miabot in a Stage representation of the real arena used in chapter 6. Starting at location S, the robot would follow the wall of the arena until forced to rotate by an oncoming wall; the robot rotated towards the direction of least resistance and continued following the wall. If, however, the robot moved towards the wall at a certain angle it would cause the robot to collide with the wall whilst attempting to turn away from it. This was similar to the real world, although in the real world collisions could be more frequent due to bad ultra-sonic echoes as described in chapter 6. After a collision, whereas in simulation the Miabot will stall and come to a permanent halt, the behaviour of the Miabot in the real world is unpredictable.

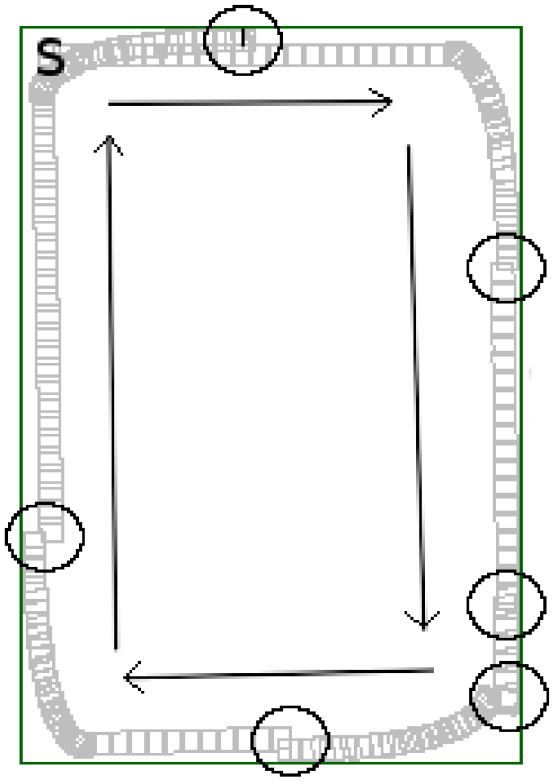


Figure 3.9: A simulated Miabot using the non-sharing controller described in chapter 4, traverses a simulation of an empty laboratory arena, as described in chapter 6. Circles represent points of collision.

3.3.3 Stage Miabot Model

The Stage Miabot model is split into three parts: the sonar, the blob-finder and the base robot. Below is an example of the Miabot sonar definitions. The number of sensors in the array, the pose of these sensors in relation to the centre of the Miabot, the field-of-view of the individual sensors and the size of the individual sensors are defined.

```
# The Miabot sonar array
define miabot_sonar ranger
(
  # number of sensors in array
  scount 8
  # define the pose of each transducer
  spose[0] [0.0375 0 0]
  spose[1] [-0.0375 0 180]
  spose[2] [0 -0.0375 270]
  spose[3] [0 0.0375 90]
  spose[4] [0.0375 -0.0375 315]
  spose[5] [-0.0375 0.0375 135]
  spose[6] [0.0375 0.0375 45]
  spose[7] [-0.0375 -0.0375 225]
  # define the field of view of each transducer
  sview [0.03 1.0 30]
  # define the size of each transducer
  ssize [0.015 0.032]
)
```

Below are the Miabot blob-finder definitions. The number of different colours the device can track is defined, along with the specific colours. The range, field-of-view and size of the camera are also defined, along with the size of the tracking frame.

```
# The Miabot blob-finder
define miaptz ptz
(
  blob-finder
  (
    # 2 colour definitions
    channel_count 2
    channels ["blue" "yellow"]
```

```
    range_max 1.0
    # only fov
    ptz [0 0 30.0]
    image [88 144]
    size [0.028 0.04]
   )
)
```

Finally, the base Miabot definition is given below. The dimensions of the chassis, the initial location (in odometry terms) and the mass of the robot are given. The type of drive system is defined, in this case differential. The type of localisation is also defined, in this case odometry. Finally, the base definition is told what sensors it is equipped with, in this case the sonar and blob-finder sensors defined above.

```
# A Miabot in standard configuration
define miabot position
(
  # actual size
  size [0.075 0.075]
  origin [0 0 0]
  # estimated mass in KG
  mass 0.567
  # differential steering model
  drive "diff"
  localization "odom"
  # use the sonar array defined above
  miabot_sonar()
  #use the blob-finder defined above
  miaptz()
)
```

3.3.4 Player Miabot Plug-in

This is only a brief introduction to the Miabot Plug-in — more information can be found at:
http://www.asap.cs.nott.ac.uk/∼robots/wiki/index.php/Player/Stage/Gazebo.

The job of the Miabot plug-in is two-fold. Firstly, to translate the abstract commands/requests from the Player server into low level Miabot commands/requests.

When the Player server connects to the Miabot for the first time, the Miabot is initialised using a configuration file, an example of which is given below. The configuration file tells the Player what devices the Miabot supports (position2d, sonar, blob-finder) and what devices it requires (tracker — this is the plug-in for the overhead camera system, see appendix 3.3.5). The Miabots communicate via Bluetooth. However, in the laboratory Bluetooth routers are deployed, which the Miabots connect to on start-up. The router assigns each Miabot a port number, and hence the configuration file provides the name of the router and the port number of the Miabot to allow TCP/IP communications over bluetooth. A number of sonar settings are also initialised, the ping mask sets which ultra-sonic sensors in the array are powered. The ping rate sets the time in milliseconds between each sonar ping. It is also possible to buffer the last 'good' sonar reading. This feature is only used when using the Nearness Diagram (ND) algorithm (used within the hybrid system described in chapter 7) to override the emergency stop procedure that occurs when the ultra-sonic sensors return zero. The Bluetooth socket wait time is also set; this is the time that the plug-in will wait on a socket waiting for a response from the Miabot. The camera settings set the initial camera parameters and the RGB value of the colour to be tracked. Secondly, the plug-in manages the Bluetooth connection to the Miabot.

```
driver
(
  name "miabot"
  plugin "miabot.so"
  requires ["tracker:::position2d:2"]
  provides ["odometry:::position2d:0"
        "sonar:0"
        "blobfinder:0"]
  # are you using the overhead tracking system
  tracker 1
  # are you using a dongle (1) or the router (0)
  type 0
  # router settings
  router "village1"
  port 5000
  # sonar settings
  ping 255
```

```
  ping_rate 200
  buffer_sonar 1
  # socket settings
  bt_wait 260000
  # camera settings
  set_cam 1
  auto_white 1
  auto_adj 1
  light_filt 0
  num_blobs 1
  redmin0 160
  greenmin0 64
  bluemin0 16
  redmax0 192
  greenmax0 80
  bluemax0 48
)
```

Player server commands are simply translated into the corresponding Miabot commands. A number of examples are shown in table 3.1:

Table 3.1: Player to Miabot command translations.

Player Abstract Command	**Miabot Command**
Start Sonar	$[Sn]$ (n is the ping rate in ms)
Start Blob-finder	$[iA0 = 0, 45, 54, 0D]$
Stop Motor	$[s]$
Set left and right wheel velocities to $10cm/s$	$[= 50, 50]$

When the Player server requests data it simply references the appropriate device data structure in the plug-in. In order to store this data, however, the plug-in listens on the appropriate Miabot socket waiting for a response to a previous command. When it captures a response, it first looks up what request the response corresponds to, by checking the response header. It is then able to translate the response into the appropriate Player data structure. Table 3.2 shows some Player server requests and the related Miabot responses.

The Miabot responds to all requests with packets (denoted by $< .. >$) of hexadecimal values. All sonar related packets have the header, *SEn:*, where n is the address of

Table 3.2: Miabot response translations.

Player Abstract Request	**Miabot Response**	**Miabot Translation**
Sonar Data	$< SE0 : 7F002F >$	0: 0.47
	$< SE2 : 8A0023 >$	1: 0.35
	$< SE4 : 640026 >$	2: 0.38
	$< SE6 : 9E001C >$	3: 0.28
	$< SE8 : 6B001E >$	4: 0.3
	$< SEA : 5400AA >$	5: 1.7
	$< SEC : 2F0017 >$	6: 0.23
	$< SEE : 2C0017 >$	7: 0.23
Blob-finder Data	$< R0A010049 >$	Count:1
	$< R6E5C71FF >$	blob 0:
		id: 0
		area: 80
		X: 82
		Y: 111
		Left: 73
		Right: 92
		Top: 110
		Bottom: 113

the sonar. When translating sonar data the Miabot can ignore the first hexadecimal value after the header, as this relates to the photodiode. The second hexadecimal value is only used when the range reading is above $2.55m$ which is possible in theory, although never seen in practice. The third hexadecimal value is the range result up to $2.55m$. The result is converted into a decimal (giving a result in centimetres) then divided by ten to give a result in meters. All blob-finder related packets have the header, *R*. After stripping all blob-finder headers and non blob-finder related packets out (e.g. sonar packets), if the first hexadecimal value after the header is *0A*, which denotes the start of blob data, then the plug-in grabs all packets until finding the hexadecimal value, *FF*, which denotes the end of blob data. The second hexadecimal value represents the number of blobs detected. The third hexadecimal value represents the colour ID of that blob (passed to the Miabot in the configuration file). The fourth, fifth, sixth and seventh hexadecimal values, represent the left, top, right and bottom of the bounding box of the blob respectively. The centre point (X and Y

values) and the area of the bounding box can be calculated from the bounding box values already obtained.

3.3.5 Player Tracker Plug-in

The manufacturers of the Miabot also provided an overhead camera tracking system; more information can be found in section 3.2.4. In order to use it within the Player architecture, a plug-in was developed. Unlike the Miabot plug-in which supported multiple devices, the tracker plug-in only needed to support a position2d device to provide the x and y co-ordinates and the orientation of a Miabot within our laboratory setup. Like the Miabot plug-in, the tracker plug-in was initialised with a configuration file, an example is given below:

```
driver
(
  name "tracker"
  plugin "tracker.so"
  provides ["tracker:::position2d:2"]
  global_server "grishnakh"
  global_port "9000"
  socket_wait 260000
  x_conv 1.62
  y_conv 2.46
  x_res 1200
  y_res 2600
)
```

As with the Miabot plug-in, the tracker plug-in supports a position2d device and communicates via TCP/IP. The overhead tracking system is made up of two local blob servers (one for each camera) and a global blob server. The local blob servers track the Miabots within their individual camera frames and broadcast their co-ordinates. The global blob server reads the co-ordinates from both local blob servers and transforms the co-ordinates into a global co-ordinate frame. As such, the plug-in only needs to know the name and port of the global blob server. The dimensions of the area covered by the global co-ordinate system are also provided along with the global pixel resolution.

When the Miabot plug-in requests the position of a Miabot (assigned a unique id on initialisation) the tracker plug-in requests the current position from the global blob server. The global blob server responds with the x and y co-ordinates of the Miabot (in pixels) along with its orientation (in degrees). The plug-in converts the co-ordinates into metres and the orientation into radians. The origin of the global co-ordinate system is in the top left hand corner, whereas the origin of the Player co-ordinate system is in the centre of the global co-ordinate system. The plug-in makes the necessary translation. The orientation systems of the global co-ordinate system and Player also differ; again the plug-in makes the required adjustments. Once the positional data is in the Player data structure, the tracker plug-in responds to the Miabot plug-in request.

3.3.6 Hybrid System: Player Configuration

In chapter 7, the sharing system is compared against a hybrid system. This system is made up of the *wavefront* and *nd* drivers provided in Player. The configuration and settings for these drivers follow.

Wavefront Driver

The Wavefront driver is the Player frontend to the wavefront propagation algorithm, described in detail in section 7.2.1. The algorithm plots a path from the robot's current location to a desired goal location.

Like the Miabot plug-in, the Wavefront driver provides and requires a number of devices. The planner device allows the robot to store goal locations and way-point locations. The "output" device is the robot to be controlled. The "input" device is the source of current pose information (e.g. the tracker plug-in). The map device is an image representation of the environment being explored. The value of 'safety_dist' is the minimum distance to a known obstacle for which a path will be planned. 'max_radius' is the radius around the robot within which obstacles are taken into consideration when planning motion; the lower the value the faster the computational time, the higher the value the more optimal the path. 'angle_epsilon' and 'distance_epsilon'

are the threshold values used to trigger goal completion. 'replan_dist_thresh' and 'replan_min_time' are the distances travelled or time passed in-order to activate the computation of a new path, whichever occurs first. 'dist_penalty' is a weight used in the computation of a solution, the higher the value the greater the punishment for cutting corners.

```
driver
(
  name "wavefront"
  provides ["6665:planner:0"]
  requires ["output::6665:position2d:1" "input::6665:position2d:0"
            "6665:map:0"]
  safety_dist 0.03
  max_radius 0.2
  distance_epsilon 0.10
  angle_epsilon 20
  replan_dist_thresh 0.2
  replan_min_time 10
  dist_penalty 10
)
```

More detail of the Wavefront driver and the parameters can be found in the Player manual at http://playerstage.sourceforge.net/doc/Player-2.0.0/player/.

Nearness Diagram Driver

The Nearness Diagram driver is the Player frontend to the ND motion control algorithm, described in detail in section 2.3.1.

Again, the Nearness Diagram driver provides and requires a number of devices. The position2d device provides the necessary commands to the robot to avoid obstacles whilst moving towards a goal location. The "input" device is the current source of pose information. The "output" device is the robot to be controlled. The "sonar" device is the source of sensory information used to avoid obstacles. The 'max_speed' and 'min_speed' parameters set the maximum and minimum translational and rotational velocities for the robot, respectively. 'goal_tol' sets the distance and rotational threshold values to trigger goal completion. 'wait_on_stall' tells the robot to

recover from stall until proceeding. 'rotate_stuck_time', 'translate_stuck_time', 'translate_stuck_dist' and 'translate_stuck_angle' are the times or distances the robot has to travel (or have travelled) in order to avoid "giving up". 'avoid_dist' is distance at which the robot will begin avoiding obstacles. 'safety_dist' is the minimum distance the robot will move next to an obstacle. 'sonar_bad_transducers' is used to disable the rear ultra-sonic sensors when calculating the obstacle density in the robot security zone.

```
driver
(
  name "nd"
  provides ["position2d:1"]
  requires ["output:::position2d:0" "input:::position2d:0" "sonar:0"]
  max_speed [0.1 10]
  min_speed [0.02 5]
  goal_tol [0.075 15]
  wait_on_stall 1
  rotate_stuck_time 30.0
  translate_stuck_time 30.0
  translate_stuck_dist 0.05
  translate_stuck_angle 5
  avoid_dist 0.30
  safety_dist 0.03
  # rear sonars not needed
  sonar_bad_transducers [1 6 7]
)
```

More detail on the Nearness Diagram driver and the parameters used can be found in the Player manual at http://playerstage.sourceforge.net/doc/Player-2.0.0/player/.

3.4 System Architecture

Throughout this thesis, all robots are controlled through the Player architecture, as described in section 3.3. The architecture provides a number of client libraries. The C++ library was chosen to develop the clients used throughout this thesis. Each robot (simulated or real) has a client associated with it. In the simulation, these

connect to a server that provides the simulated environment. In the real world, each robot has a server associated to it, which the relevant client connects to.

Communication between clients is achieved through the reading and writing of shared files. The reasons for implementing communications in this manner are two-fold:

1. As all of the client programs are run from the same computer, this method simplified the communications process.

2. The Miabot robots used in our laboratory experimentation have Bluetooth communications modules. Bluetooth uses a virtual serial connection. As such, a device can only be connected to one other device at any one time. As the Player server for each Miabot is hosted on a remote PC, the single connection for each Miabot must be with this PC; otherwise the Miabots would be uncontrollable.

Each client has write access to its own communications file; eand has read access to every communications file within the multi-robot group. As only one client has write access to any given file there is no danger of multiple clients trying to write to the same file at the same time. The only drawback of the system is that when a robot reads a file no guarantee is given that the file is up to date. This is not a major issue as, due to the communications lag within the system, all sensory information is "old", and so the increased lag is considered to be insignificant.

The communications file contains two pieces of information. The first piece of information is the "current" position and orientation of the robot, the position is used by the robot reading the file to see if this robot is within its local group radius. If so, the orientation is used to discover which part of the potential field is being shared. More information can be found in chapter 4. The second piece of information is the robot's "current" potential field. This information is read only if the robot is sharing potential field information. The communications file is simply a tab separated file which is parsed by other robots within the system to read the relevant information. An example is shown in table 3.3 (see section 4.2.1 for more information on the potential field construction).

Table 3.3: Miabot Communication File: Where x, y and θ are the co-ordinates (in metres) and orientation (in radians) of the robot respectively, and 0-7 are the generated forces for the eight ultra-sonic readings (the potential field).

x	y	θ	0	1	2	3	4	5	6	7
0.1	0.34	1.2	11.1	4.7	3.3	11.1	11.1	11.1	11.1	1.2

As can been seen from the above description and in figure 3.10, currently the multi-robot system is implemented on a centralised network. However, as has been discussed, this is due to limitations with the current hardware being employed. It is hoped that, in the future, the multi-robot system can be implemented as a fault-tolerant distributed system. In concept, the system is already distributed as each client/server pair is an individual agent that makes its own decisions. To make the system physically distributed, the Bluetooth communications module would need to be replaced by a broadcast capable medium e.g. wireless Ethernet. Consequently, instead of writing a communications file, the data in table 3.3 would be constructed as a TCP/IP packet. The memory capacity of the Miabots would also need to be dramatically increased in order to be able to run an embedded Linux operating system running the Player server and client.

3.5 Summary

In this chapter, the Miabot Pro and other hardware used in the set of experiments presented in this thesis have been described and their limitations discussed. The Miabot Pro consists of a base module, which contains a differential steering drive, a processor board, and a Bluetooth communications board. The base module has the capacity to support an additional eight I/O devices. However, the base module only contains a single communications bus which each device must use in order to transmit/receive information. Additional devices are treated as either *masters* or *slaves* of the bus. As only the blob-finder module is a master of the bus, in our experiments, the only time any issue arises is when blob information is abundant within the environment, causing the blob-finder module in effect to permanently lock the bus, not allowing other, perhaps, critical information to be communicated (i.e.

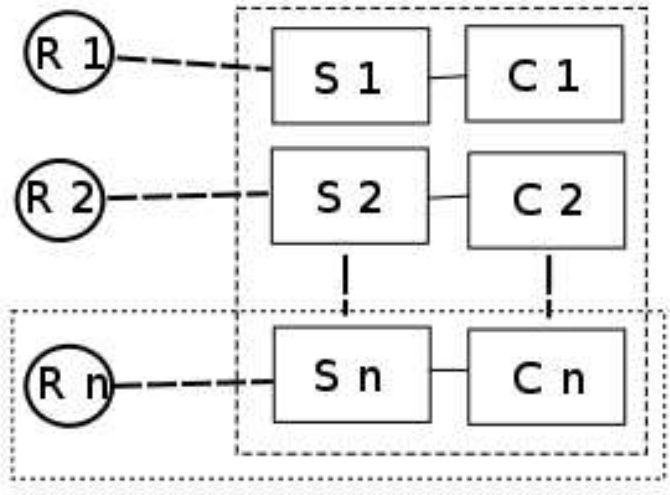


Figure 3.10: The physical systems architecture is centralised, each robot (R) is controlled by a server (S) and client (C) hosed on a remote PC (dashed rectangle). In concept, the system is distributed (dotted rectangle), with each robot making it's own decisions.

ultra-sonic data). The chance, of situations of high blob count is reduced only by defining a single blob to be tracked and by damping the effect of light reflections within the environment. The ultra-sonic module has 15^o blind spots every 45^o. The closer to the Miabot and the smaller the objects are, the more likely it is that the objects will not be detected. However, in the experiments conducted in this thesis, all objects are large enough to be detectable at all ranges. Due to the large and accumulative error within the Miabot encoders, a global tracking system is used to position the Miabots within the experimental environments. The global tracking system consists of 2 local blob systems (covering the two halves of the environment) and a global blob system, which consolidates the 2 local co-ordinate systems into a single global co-ordinate system.

The software architecture used to control the physical/simulated Miabots and

the global tracking system has also been discussed. The 2d simulation model of the Miabot Pro was described, including models for the ultra-sonic array and the blob-finder. A plug-in module for the Player robot controller that allowed control of the Miabots at an abstract level was also discussed; configuration options and a general overview of its internal workings were described. A plug-in for the overhead camera system was also discussed. The configuration options for the wavefront and ND drivers (used by the hybrid system detailed in chapter 7), already implemented within the Player architecture, were also discussed.

The current and future architecture of the multi-robot system was discussed. Due to hardware limitations, the system is currently deployed as a centralised system. However, the system has been designed to be distributed and it would be of interest to deploy the system as a distributed architecture.

Chapter 4

A Potential Field Sharing Multi-robot System

4.1 Introduction

Although there has been much work in both the *unaware* and *weakly co-ordinated* categories of multi-robot systems (described in chapter 2), the mixture of the two employed in the multi-robot system detailed in this chapter represents a novel co-ordination architecture. The system is *unaware* at the global level, yet *weakly co-ordinated* at the local level. It is believed that this is beneficial to the system as co-ordination only occurs when needed. However, this co-ordination is not necessarily needed to complete an assigned task. In this chapter, the multi-robot system will be discussed in detail and concluded with a summary.

4.2 Sharing Potential Fields

The outline of the processes involved in the new multi-robot system is as follows: Individual robots construct potential fields from available sensor data (in the current system only ultra-sonic data is used), see figure 4.1 A-B. Robots that are assigned to the same group then calculate local group intersections and share relevant potential field information, figure 4.1 C. Each individual robot then creates a combined potential field using the shared information (figure 4.1 D) and then makes the relevant

action selection, (figure 4.1 E). The non-sharing system that is used as a control for the experiments in chapters 5 and 6 only contains the processes A, B and E in figure 4.1.

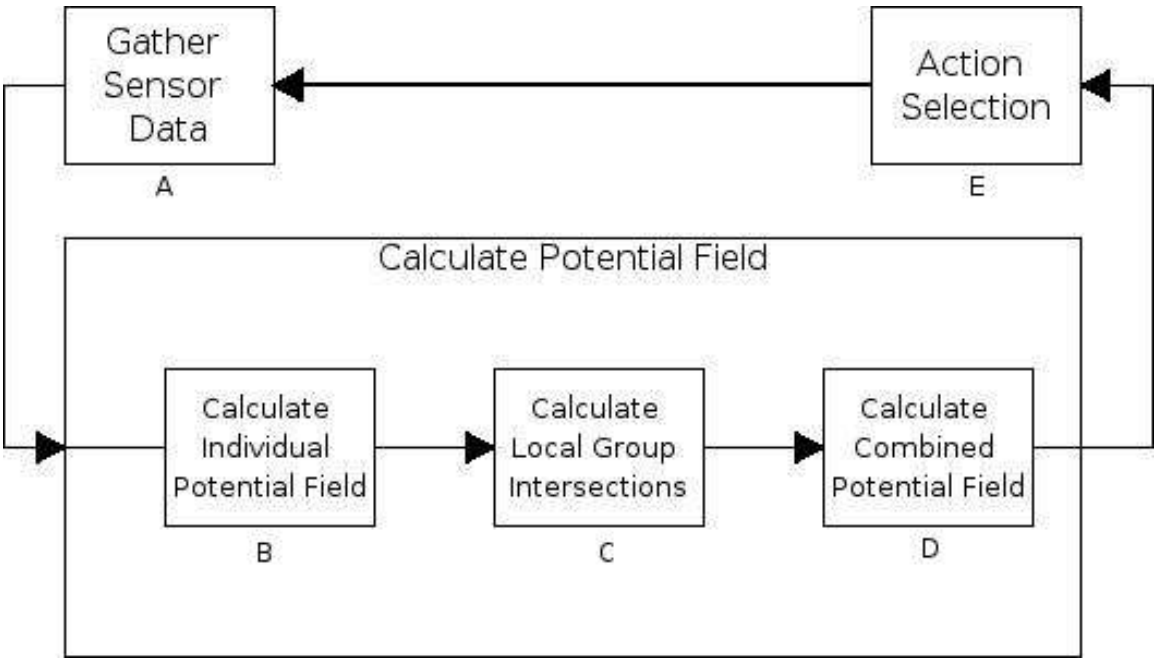


Figure 4.1: Given some sensory input, individual potential fields are created. Robots within local groups share information and create shared potential fields. Action selection is based upon these potential fields.

4.2.1 Individual Potential Field

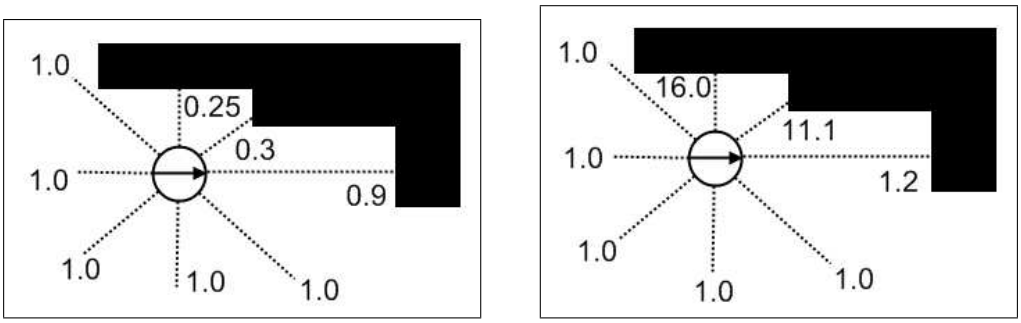
The basis of our multi-robot system is Coulomb's law of electrostatic force, as given by

$$F = k_C \frac{q_1 q_2}{r^2}, \tag{4.1}$$

where F is the force, q_1 represents the unit charge of the robot and q_2 the unit charge of an obstacle detected by an ultra-sonic sensor. For simplicity, all obstacles and robots are given a unit charge of 1. r is the sensor reading from the relevant ultra-sonic sensor, the distance of the closest object. k_C is an electrostatic constant which is ignored (set to 1). Therefore, the calculation is now the inverse square law:

$$F = \frac{1}{r^2} \tag{4.2}$$

A small reading from the ultra-sonic sensor results in a high force. The reason why Coulomb's law is simplified is that, as only the ultra-sonic sensors are used to create the potential field, it is not possible to distinguish between objects. As such, all objects have to be assigned the same unit value. Therefore, the unit charge of the robot is also set to the same unit value, in order to get the appropriate behaviour (attraction to regions of low resistance). The electrostatic constant is ignored as it is just a constant and has no direct effect in this application. An example of the potential field calculations is given in figure 4.2, where the detection of objects which are close by results in high repulsive forces. Distant objects have a low repulsive force. As the system was designed with the Miabot Pro class of robot in mind, eight forces are calculated per robot corresponding to the fact that each robot has eight ultra-sonic sensors — see section 6.2 for the full robot specification.



(a) (b)

Figure 4.2: (a) Example ultra-sonic readings. (b) Resultant forces.

4.2.2 Local Group Interactions

In this system, local groups of robots are formed which share information. Local groups are formed by sequentially iterating through the robots within the environment and assigning other robots to their group if they are within a given distance of that robot. In the simulated experiments, a distance of $2m$ is used (double the range of

the ultra-sonic sensors). In the laboratory experiments, a distance of $70cm$ is used (double the boresight reflection[1] of the ultra-sonic sensors). An example of local groups is shown in figure 4.3, where robot 1 assigns robots 2 and 3 to its group, robot 2 assigns robots 1 and 3 to its group, robot 3 assigns robots 1, 2 and 4 to its group, robot 3 is also assigned by robot 4 to its group, and finally robot 5 is on a group of its own.

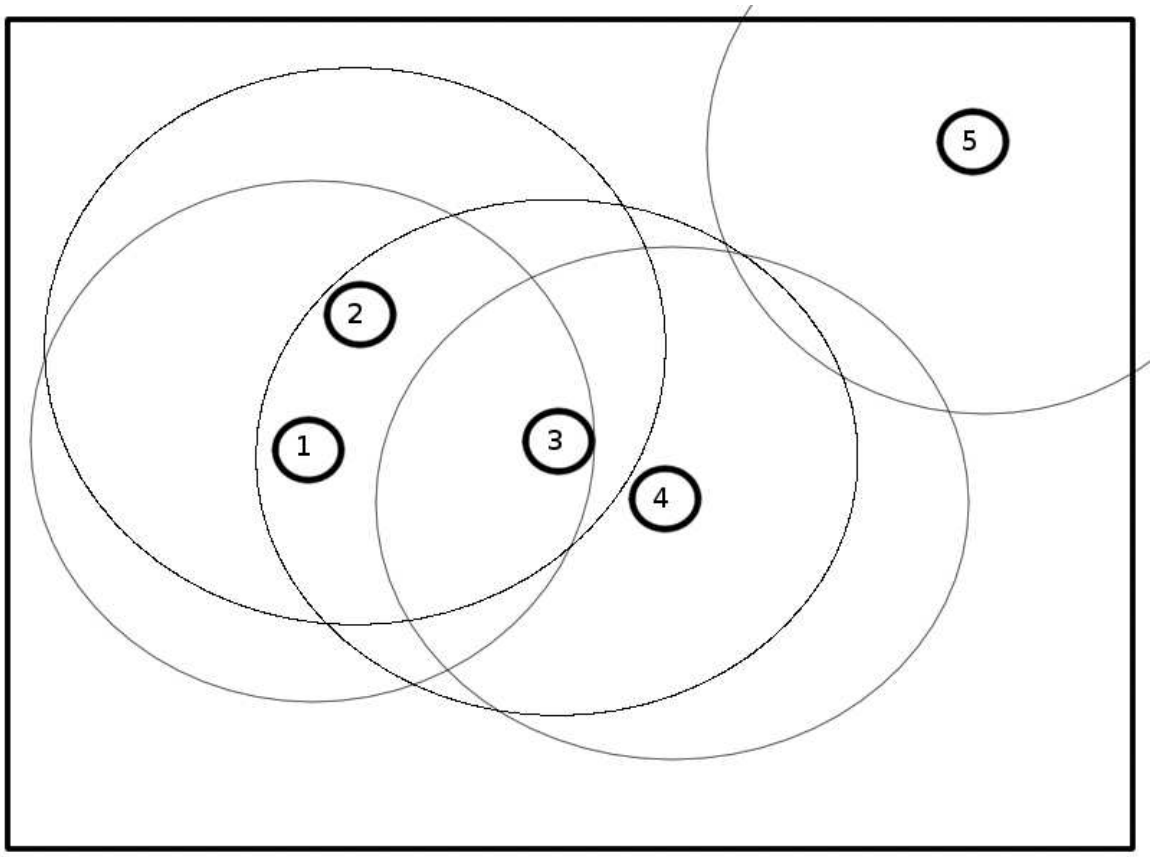


Figure 4.3: Local groups: group 1 and 2 (robots 1, 2 and 3), group 3 (robots 1, 2, 3 and 4), group 4 (robots 3 and 4) and group 5 (robot 5).

Euclidean distances have been used to define local groups, in order to simulate likely communication range limitations. In the robotics laboratory at the University of Nottingham, the maximum area that the robots can explore is approximately

[1]The boresight of an ultra-sonic sensor is the angle to which it is pointing. Boresight reflection refers to the response received by the ultra-sonic sensor from the ground.

$5m \times 3m$ — this is well within the communications range of the Bluetooth modules embedded in the robots used during experimentation (Class 2 — $100m$ range). No barriers to communication exist within the environment. In a real world application, it would be expected that the communications range of the robots would be heavily affected by the environment within which they are situated. The reliability of the communications system would also vary depending on the environment. This is one of the reasons why the motion of the individual robots is not reliant upon receiving information from local group members. In the real world, if no communication was received from group members, an individual robot would use its own potential field information to make action selection choices. Indeed, the action selection process does not distinguish between individual and shared potential fields.

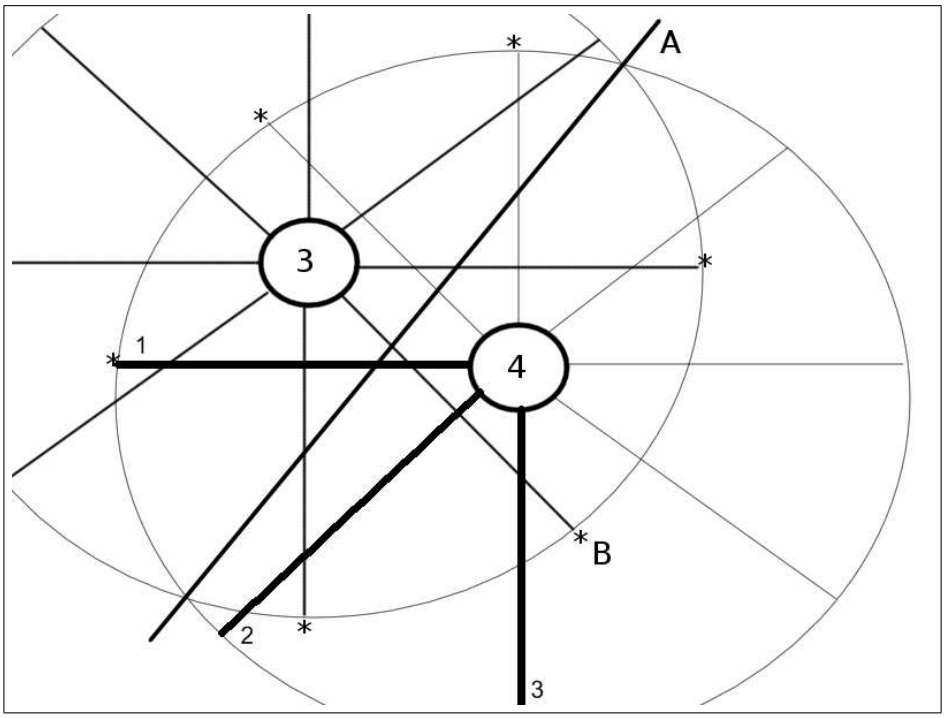
4.2.3 Combined Potential Field

Once local groups have been assigned, potential field information is shared. In order to explain this as clearly as possible an example, as shown in figure 4.4, will be described in detail:

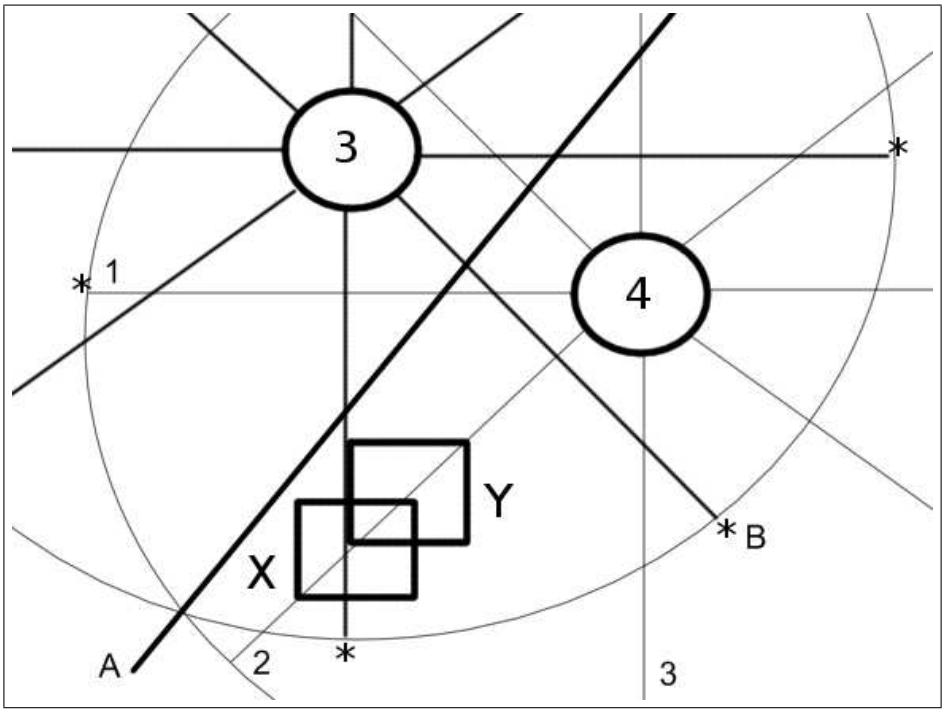
1. The robots are modelled within 2-dimensional space. Ultra-sonic ranges are represented as lines on a plane and the sensory limits of a robot are represented as circles on a plane.

2. Each robot is within each others' local group. All ultra-sonic ranges that intersect the line A (the radical line — the line that intersects the two points of intersection between the two circles) are available to share information. These lines are marked with a $*$ in figure 4.4a.

3. Information is shared between these ultra-sonic ranges and any of the other robots ultra-sonic ranges that they intersect. To simplify the example, we will use line B (figure 4.4a). It shares information with 3 ultra-sonic ranges (thick lines). In such cases, calculations are completed on intersections sequentially. In the example, the force of B would be compared with the force of 1, then the force of 2, then 3.



(a)



(b)

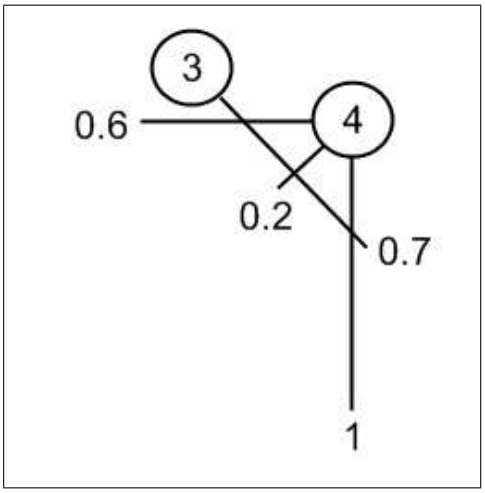
Figure 4.4: (a) A 2-dimensional model of the robots: lines represent the ultra-sonic ranges. Line A is the radial line of the two circles representing the sensory limits of the robots. (b) Obstacle Detection: Robot 3 detects obstacle X, robot 4 detects obstacle Y.

4.2.4 Optimistic and Pessimistic

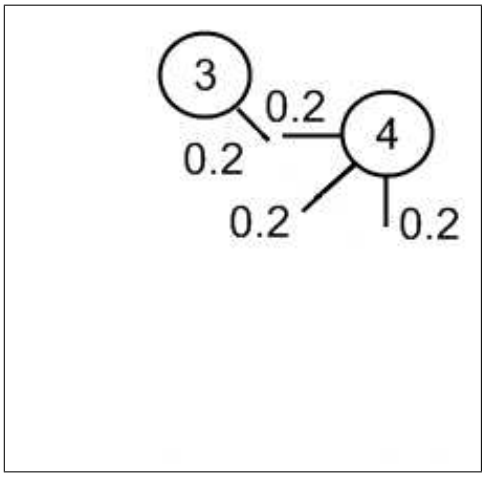
Two versions of the potential field sharing multi-robot system have been implemented — referred to as *pessimistic* and *optimistic* in terms of sensor noise. In the pessimistic system, when two lines intersect their forces are compared and the highest value is used. In the optimistic system, the lowest value is used. In our example (figure 4.4b), robot 3's ultra-sonic sensor detects the obstacle X. However, robot 4 detects obstacle Y. In the pessimistic system, robot 4's force value, being the greater of the two, is used by robots 3 and 4. Hence, robot 3 would detect an obstacle that is closer than would have been previously possible. For example, if the lines are initialised with the following values: B = 0.7, 1 = 0.6, 2 = 0.2, 3 = 1, as shown in figure 4.5a. After robot 3 makes it's comparisons, the values would be: B = 0.2, 1 = 0.6, 2 = 0.2, 3 = 1 and after robot 4's comparisons, the values would be: B = 0.2, 1 = 0.2, 2 = 0.2, 3 = 0.2, as shown in figure 4.5b. However, if robot 4 made its comparisons first, a different result would occur: B = 0.7, 1 = 0.6, 2 = 0.2, 3 = 0.7, after robot 4's pass, then B = 0.2, 1 = 0.6, 2 = 0.2, 3 = 0.7 after robot 3's pass, as shown in figure 4.5c.

Conversely, in the optimistic system, robots 3 and 4 would use the smaller force value from robot 3 and so robot 4 would detect an obstacle further away than previously. For example, if the initial values previously stated are used again. After robot 3 makes its comparisons the values would be: B = 1, 1 = 0.6, 2 = 0.2, 3 = 1 and after robot 4's comparisons the values would be: B = 1, 1 = 1, 2 = 1, 3 = 1, as shown in figure 4.5d. However, if robot 4 made its comparisons first, a different result would occur: B = 0.7, 1 = 0.7, 2 = 0.7, 3 = 1 after robot 4's pass, then B = 1, 1 = 0.7, 2 = 0.7, 3 = 1 after robot 3's, as shown in figure 4.5e.

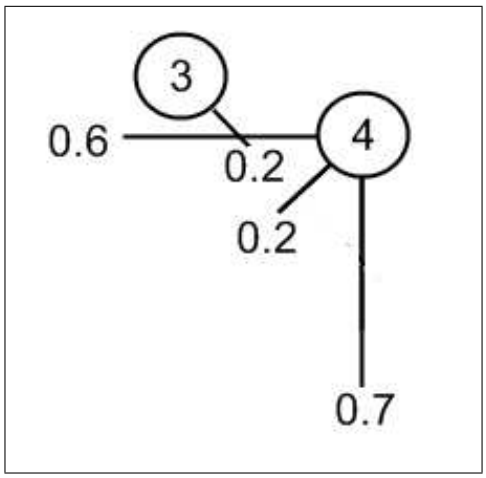
As has been demonstrated in both sets of examples, the sequence of comparisons can make a significant difference to the resulting potential field, the difference being the scale to which the system is pessimistic or optimistic. For example, when comparing the pessimistic systems, it can be seen that the system in figure 4.5b is more pessimistic (in terms of sensor noise) than the system in figure 4.5c. Conversely, when comparing the optimistic systems it can be seen that the system in figure 4.5d is more optimistic (in terms of sensor noise) than the system in figure 4.5e.



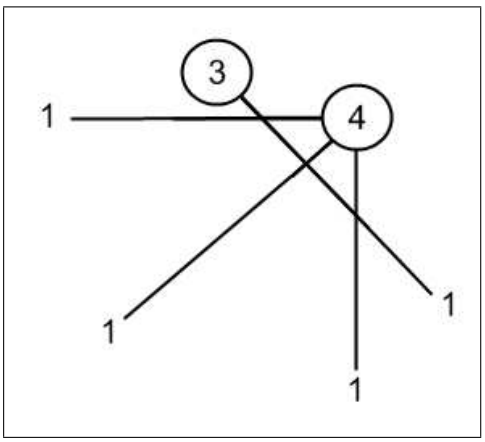
(a)



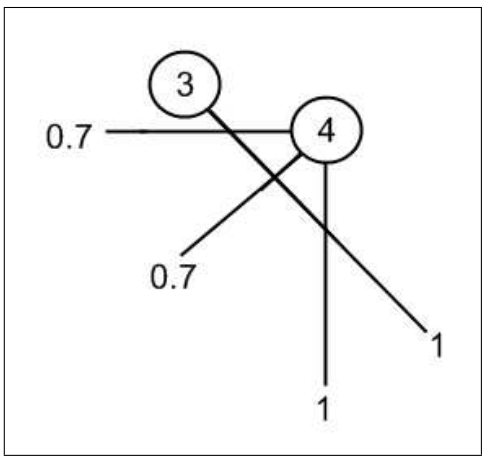
(b)



(c)



(d)



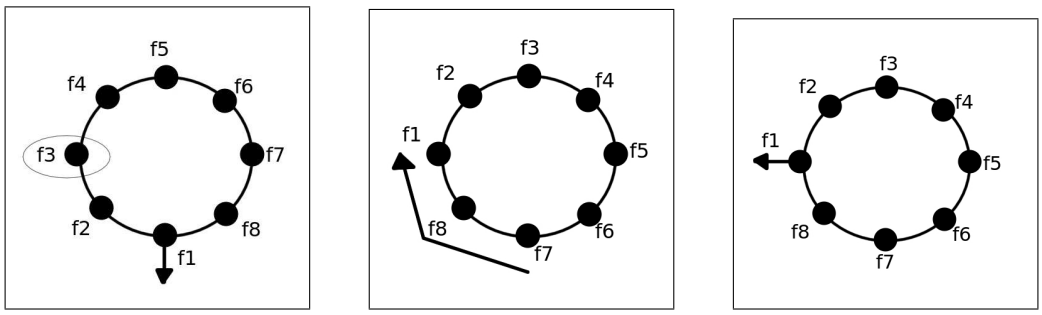
(e)

Figure 4.5: (a) The initial potential fields. (b) Very pessimistic. (c) Normal pessimism. (d) Very optimistic. (e) Normal optimism.

The desired advantage of the pessimistic system is that it will be less vulnerable to false negatives (not detecting obstacles that are there). However, it will be more susceptible to false positives (detecting obstacles that are not there). The desired advantage of the optimistic system is that it will be less vulnerable to false positives. However, it will be more susceptible to false negatives.

4.2.5 Action Selection

Once the combined potential field has been calculated (or not in the case of the non-sharing control), the minimum force f_{min} is discovered, as shown in figure 4.6a, where f_3 is assigned the lowest force. The robot has a default forward motion within the environment unless it comes across an obstacle in its path. A force value of 25 or less is used, even though this is $20cm$ from an obstacle. This is due to lag within the communications system, meaning that the robot will not avoid the obstacle until a distance of about $5cm$. In which case the robot rotates towards f_{min}, as shown in figure 4.6b, where the robot rotates towards the initial orientation of f_3. Once the forward orientation of the robot equals the direction of f_{min}, the robot resumes its forward motion, as shown figure 4.6c. Using this algorithm, robots move away from areas of high charge (obstacles) and towards areas of low charge (open spaces).



(a) (b) (c)

Figure 4.6: (a) The robot calculates the minimum force, f3, (b) The robot rotates towards the minimum force, (c) The robot moves forwards (towards the minimum force).

4.3 Limitations of proposed potential field method

As described in section 2.6.1, the potential field method has a number of limitations. How the system presented in this thesis differs in relation to these limitations will now be discussed.

1. *Trap Situations*: This is still a limitation within this system. It is also worth noting that currently no global trap recovery is employed. Hence, once a robot enters a U shaped object it is not guaranteed that it will ever escape.

2. *No passage*: As the system employs a default forward motion to all robots until they meet an object being directly in their path, it is not affected by this limitation directly. However, it is indirectly affected when the ultra-sonic sensors do not differentiate between objects. An example is shown in figure 4.7a, where the robot (the square) is unable to distinguish between the two objects (grey circles) and hence creates a ghost object (the shaded region).

3. *Oscillation in the presence of obstacles*: Again the system is not directly affected by this limitation. Only if the ultra-sonic sensors produce bad echo data is the system affected. That is, the sensed distance to an object keeps changing, so as to make the robot believe that it would be advantageous to turn towards the object. An example is shown in figure 4.7b, where the robot has a bad sensor reading (the dashed triangle) which results in a sub-optimal path being taken (bold line). The straight line is the desired optimal path.

4. *Oscillation in narrow passages*: If a robot meets the passage head on, as in the *No passage* case, it is not directly affected by this limitation. However, if a robot enters a passage at an angle, then oscillation can occur until the robot's orientation matches the orientation of the passage or the robot exits the passage. An example is shown in figure 4.7c, where the dashed squares and triangles are the position of the robot and its forward ultra-sonic range in future time steps.

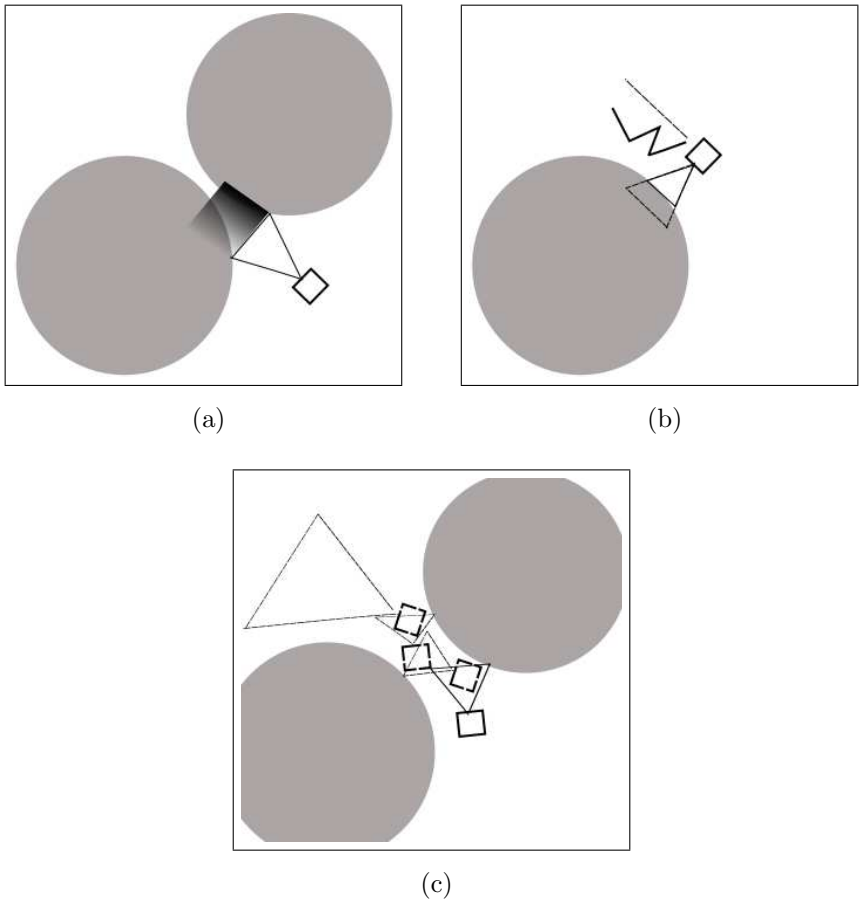


Figure 4.7: (a) No Passage. The shaded region is where the ultra-sonic sensor "thinks" an obstacle exists. (b) Oscillation in the presence of an obstacle. The bold line is the path taken when intermittent bad ultra-sonic echoes occur. (c) Oscillation in a narrow passage.

4.4 Place in Farinelli's Multi-robot Taxonomy

Unlike previous multi-robot systems, the potential field sharing system presented in this chapter does not fit neatly into one category of the Farinelli's multi-robot taxonomy (as discussed in section 2.4.2), as the system possesses different characteristics at different levels. Each robot within the multi-robot system can move freely between the different levels of co-ordination throughout the given task.

The multi-robot system presented in this chapter is *unaware* at the global level. The relevant section of the multi-robot taxonomy hierarchy is shown in figure 4.8. The global level is everything outside of the local group radius of an individual robot.

At this level, robots are not assigned to local groups. The only source of sensory input to the potential fields is the ultra-sonic array, which only provides range readings to the nearest object. It is not known what this object is, as robots have no concept of team members or indeed other robots outside of their local groups. A blob-finder module (as described in chapters 5 and 6) is used to detect the target. However, it is not used for team member recognition. As shown in figure 4.9 groups A, B and C are *unaware* systems.

At the local level, the system is considered *weakly co-ordinated*. The relevant section of the multi-robot taxonomy hierarchy is shown in figure 4.8. The local level is everything within the local group radius of an individual robot. At this level, robots are assigned to local groups — the members co-ordinate implicitly through the use of shared potential fields. However, no explicit co-ordination occurs as robots do not have the ability to distinguish between team members from other objects within the environment as previously discussed. As shown in figure 4.9, robots 1 and 2 are *weakly co-ordinated*, as are robots 3 and 4, whilst robot 5 reverts to single-robot system, as depicted in the multi-robot taxonomy hierarchy shown in figure 4.8.

It is worth noting that the system could converge to an entirely *weakly co-ordinated* system. This is more likely within small environments or in cases where the local group radius has been set to be arbitrarily large. Conversely, the system could also disperse to an entirely *unaware* system. This is more likely in large environments or in cases where the local group radius has been set to be arbitrarily small. It is believed that the hybrid of the two system types will be beneficial to the system, as co-ordination only occurs when needed. However, this co-ordination is not a necessity to task completion.

It is also envisaged that more tightly coupled tasks such as search and rescue may require a higher level of co-ordination than the *weakly co-ordinated* local level can provide. In such cases, the local level could be substituted with a *strongly co-ordinated* system, without disrupting the global level. As such, it would still be possible to design a system with low levels of communication and co-ordination during the "search" aspect of the task, only relying upon communication and co-ordination when necessary. That is, the "rescue" sub-task.

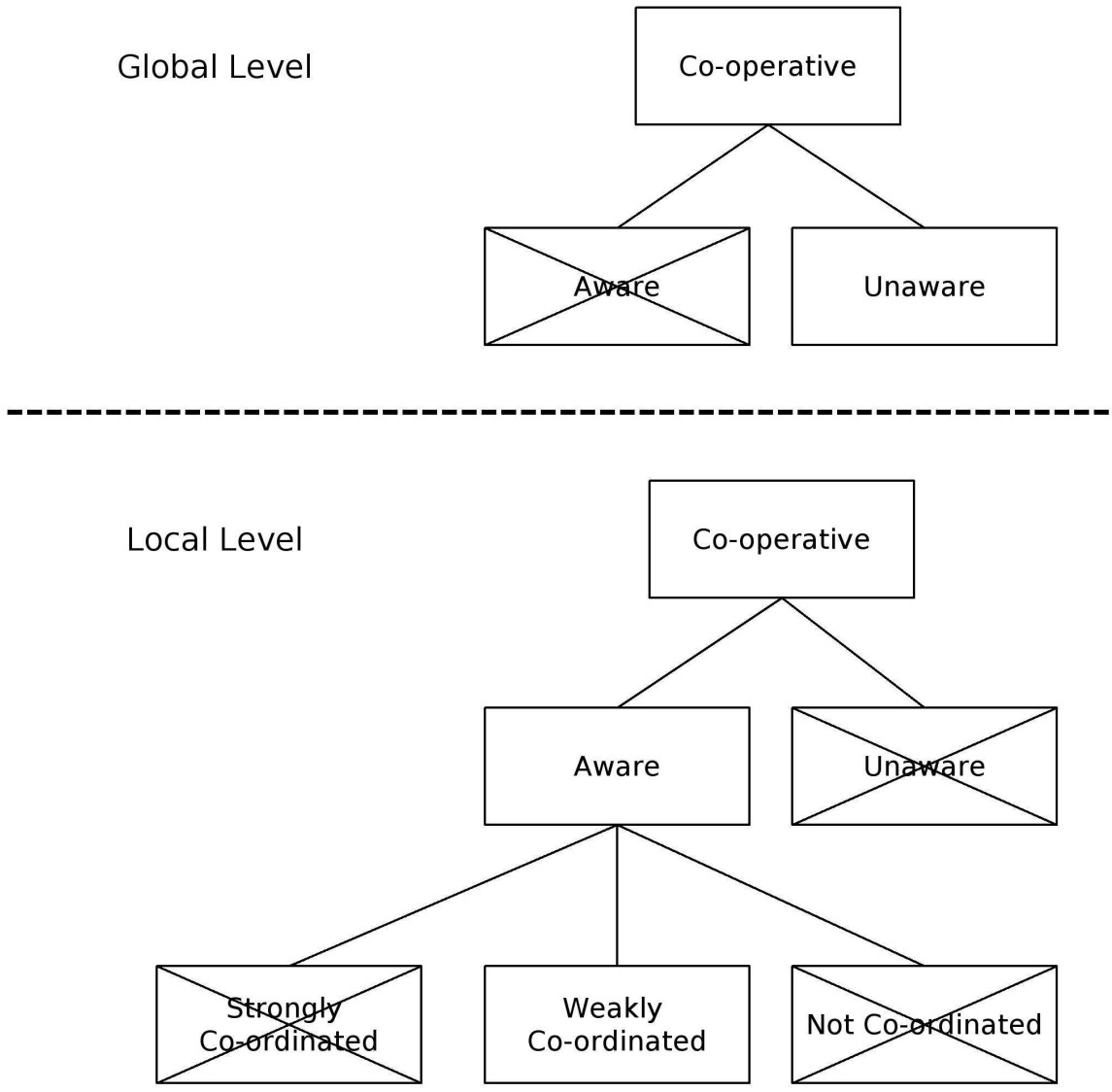


Figure 4.8: Hierarchical view of the sharing potential field method's place within Farinelli's taxonomy. Crossed out categories are not implemented.

4.5 Summary

In this chapter, a new multi-robot system has been described in detail. Potential field information is shared between members of local groups to implicitly co-ordinate the behaviour of the robots. Individual potential fields are calculated for all robots using an inverse square law; robots within an arbitrary radius from one another are

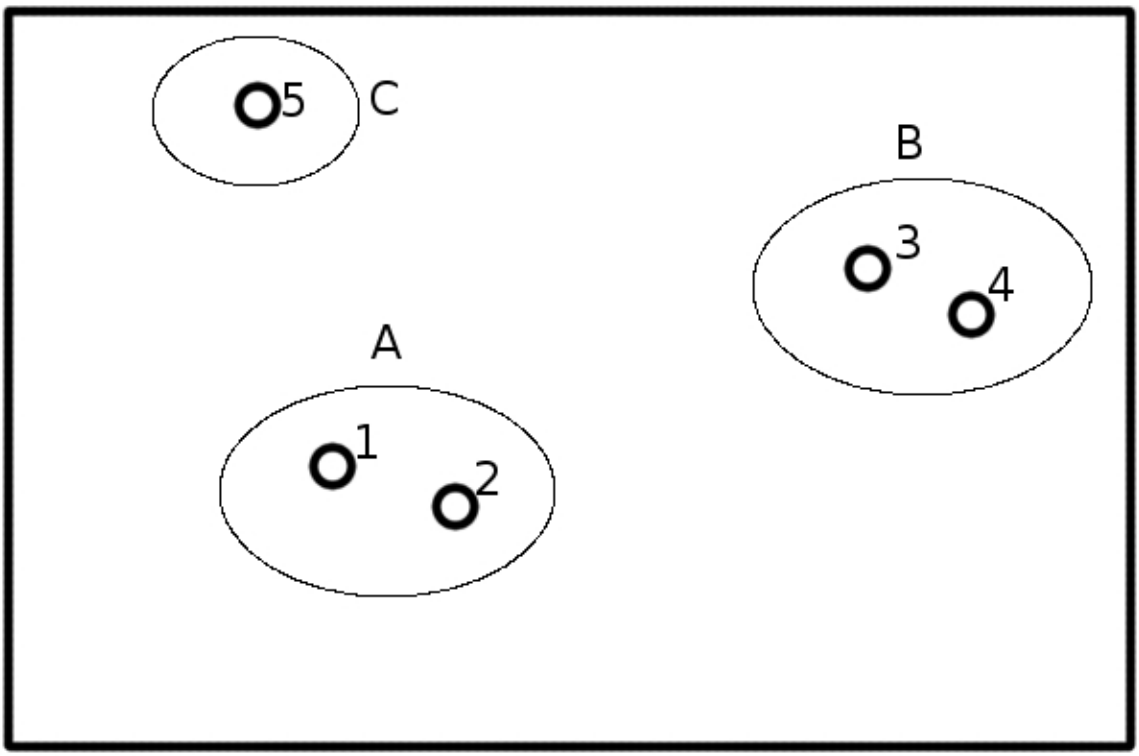


Figure 4.9: Groups A, B and C are all *unaware* of each other. Robots 1 and 2 are *weakly co-ordinated*. Robots 3 and 4 are *weakly co-ordinated*. Robot 5 reverts to non-sharing behaviour.

assigned to the same local group. Robots within these local groups share information. Two varieties of the system are presented: (1) the pessimistic system in which higher forces overwrite lower forces (and has the desired advantage of being less vulnerable to false negatives with regards to sensor noise), and (2) the optimistic system in which lower forces overwrite higher forces (which has the desired advantage of being less vulnerable to false positives).

A comparison to the traditional potential field method was conducted in terms of susceptibility of the multi-robot system presented to the known limitations of the potential field method. It is shown that although the system is still susceptible to *trap situations* and *oscillations in narrow passages*, it is not directly susceptible to *no passage* and *oscillations in the presence of obstacles*. However, the more noisy the sensor information, the more affected by the latter limitations the system becomes.

A discussion on where the multi-robot system belongs within the multi-robot

taxonomy clearly shows that the system is *unaware* at the global level, yet *weakly co-ordinated* at the local level. This is beneficial to the system as co-ordination occurs when needed, but the system does not rely upon this co-ordination to complete a task.

The novel multi-robot system, presented in this chapter, has helped this project meet one of its goals given in chapter 1.

- **To design and implement a new type of multi-robot system based upon Farinelli's multi-robot taxonomy (as described in section 2.4.2).**

  The two level architecture, described in this chapter, is novel. It allows the system to benefit from team co-ordination, but not be dependant upon it to make decisions. Making the system more fault tolerant with respect to the loss of robots or communication failures.

Chapter 5

Simulated Search Problems

5.1 Introduction

Before attempting to run the system on real robots, it was decided to run a number of experiments in simulation. In his early work, Brooks was highly critical of developing artificial intelligent robots through "toy" worlds (simulations or simple block worlds) [24]. He argued that no matter what the intention of the researcher, these worlds have been designed specifically for that robot. He also argued that if a robot is to survive in a real environment it must learn in that environment. However, in certain cases, such as this project, when it was not practical to prototype the system on real robots for various logistical reasons, simulation can play an important role as long as the developer of the system does not lose track of the end goal of implementing the system on real robots in the real world.

The robots were given the task of navigating through a number of simulated environments to find a target(s), within a time limit. This task is akin to the 'search' sub-task of the search and rescue problem defined in section 2.2. In this chapter, the simulation is described, both the environment and the robots. The statistical methods used to measure the performance of the systems are then discussed. As well as the two sharing potential field variants, a non-sharing system is employed as a control. Three experiments are described and their results shown — first the single target search problem is tackled, then the multiple target search problem. The final set of experiments investigates the effect of sensor noise on the performance of the

potential field sharing systems, as well increasing the group size and environment size. The chapter ends with a summary.

5.2 Simulated Environment

The system was prototyped using Stage; a 2-dimensional real-time simulator provided with the Player architecture as discussed in chapter 3. Four environments, containing obstacles and targets were created. In order to maintain impartiality, the number of obstacles, the placement of the obstacles and targets within the environment and the starting location of the robots were all generated randomly (C rand function, current time used as the seed). The simulation environments shown in figures 5.1, 5.2 and 5.3 had dimensions of $5m \times 3m$. The simulated environment in 5.4 had a dimension of $10m \times 6m$.

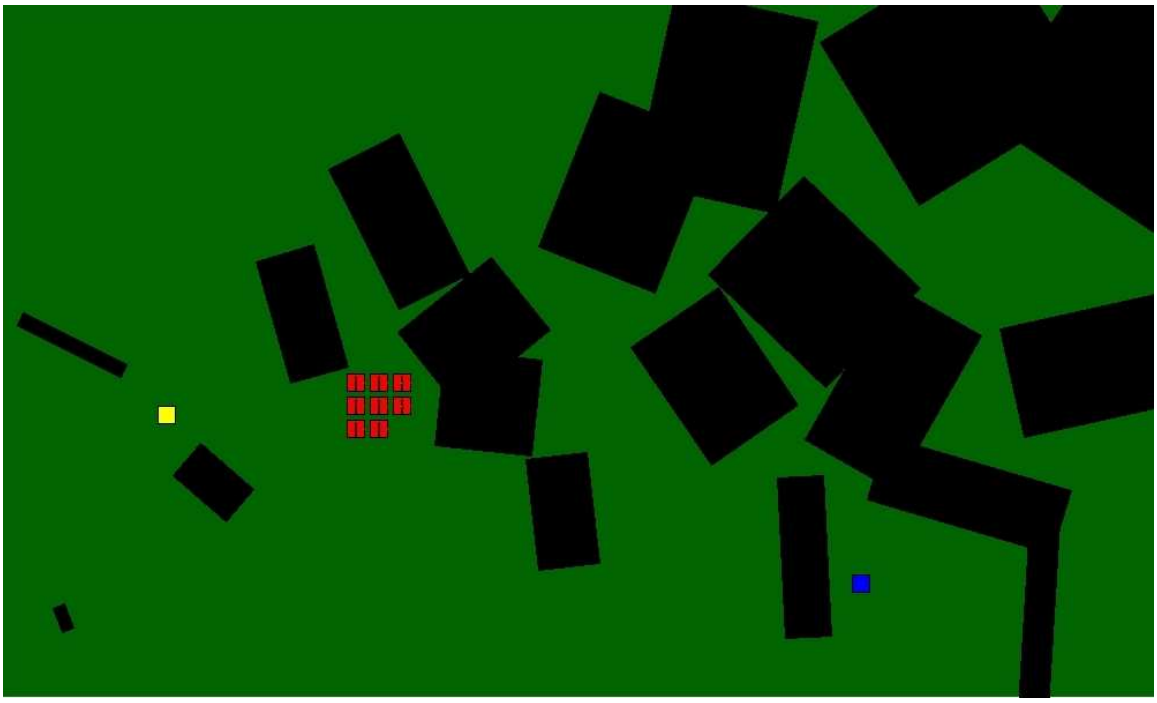


Figure 5.1: Environment 1. It consists of 8 robots (red squares), 2 targets (yellow and blue squares) and 19 obstacles (black rectangles)

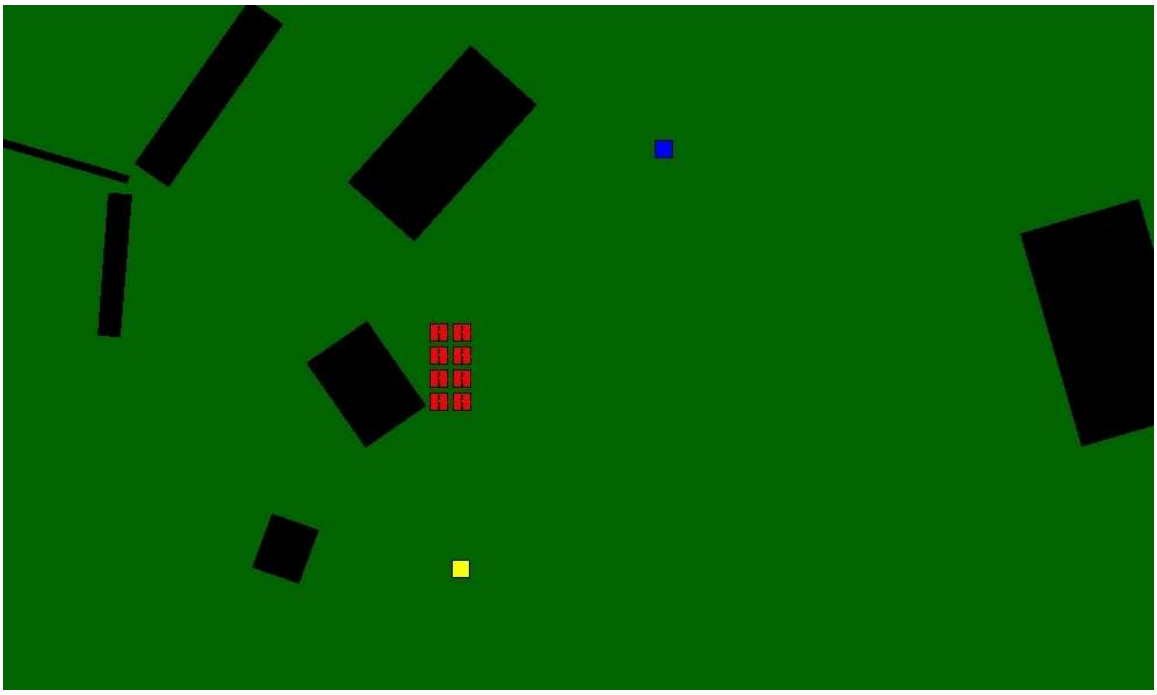


Figure 5.2: Environment 2. It consists of 8 robots (red squares), 2 targets (yellow and blue squares) and 7 obstacles (black rectangles)

The simulated Miabots contained the position2d, sonar and blob-finder interfaces (as defined by the Player robot controller). The parameters of these interfaces were set as close to the real devices (that would be used in future experiments) as was possible within the simulation environment. A summary of the interfaces is given below. The complete Stage model definition file is given in section 3.3.3.

- *Position2d*: The Miabot was modelled as a non-holonomic (the robot cannot move in arbitrary directions) robot with a differential drive (different velocities must be sent to each wheel to turn the robot).

- *Sonar*: Each of the 8 ultra-sonic sensors from the Miabot were modelled in simulation with a minimum range of $3cm$ and a maximum range of $1m$ (limited to $1m$ as in the laboratory environment the real sensors rarely give readings above $1m$), with a field-of-view (FOV) of 30^o. This gives the robot a total FOV of almost 360^o. Small gaps of 15^o exist between each individual sensor. See figure 5.5a.

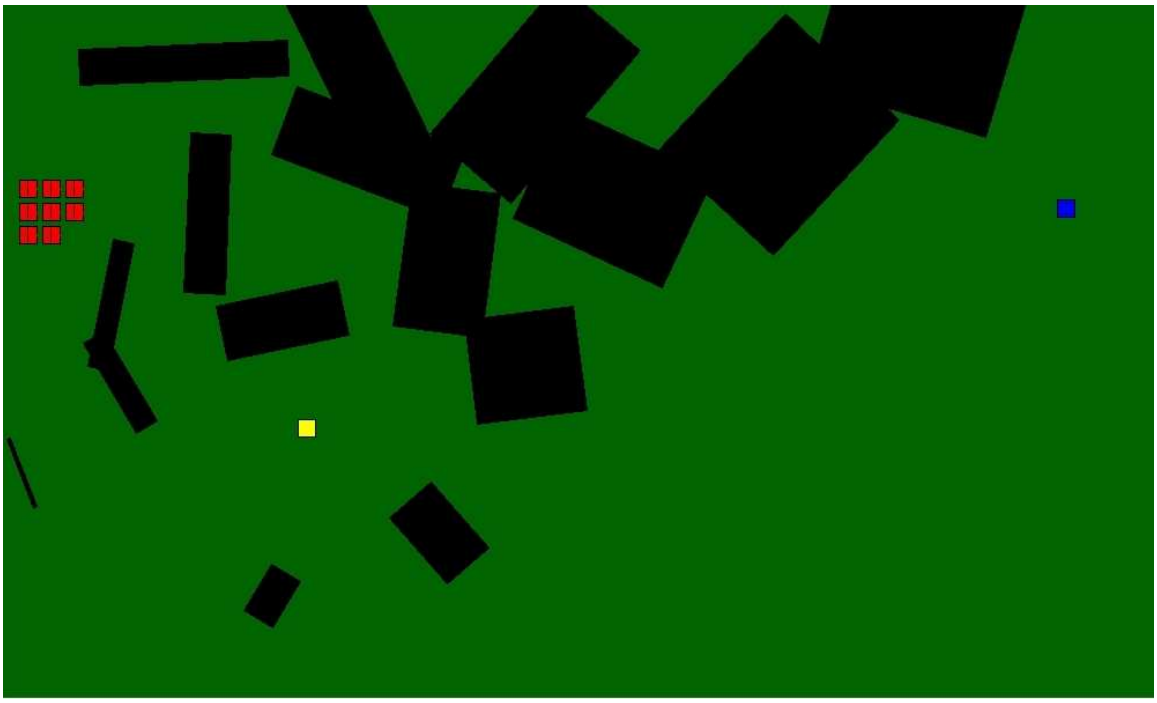


Figure 5.3: Environment 3. It consists of 8 robots (red squares), 2 targets (yellow and blue squares) and 16 obstacles (black rectangles)

- *Blob-finder*: The blob-finder was modelled in the simulation with a range of $1m$ (limited to mimic the real camera image going out of focus) and a FOV of 30^o, with a fixed forward orientation with regards to the robot. The blob-finder could track blue and yellow objects. See figure 5.5b.

During these experiments, the robot's motion was limited to either *move forwards* or *rotate*. The robot's forward motion (S) was relative to the force calculated for the forward ultra-sonic sensor (F), $S = 1 - (\frac{F}{10})$, which gave a maximum speed of approximately $1m/s$. Hence, the closer the robot came to an obstacle, the slower the velocity became, and vice-versa. It rotated at approximately $0.025rad/s$, with a forward motion of approximately $0.5m/s$ (to avoid oscillations). The blob-finder was used to detect the colour of the target. Once the target(s) were verified, the robot(s) came to a permanent halt.

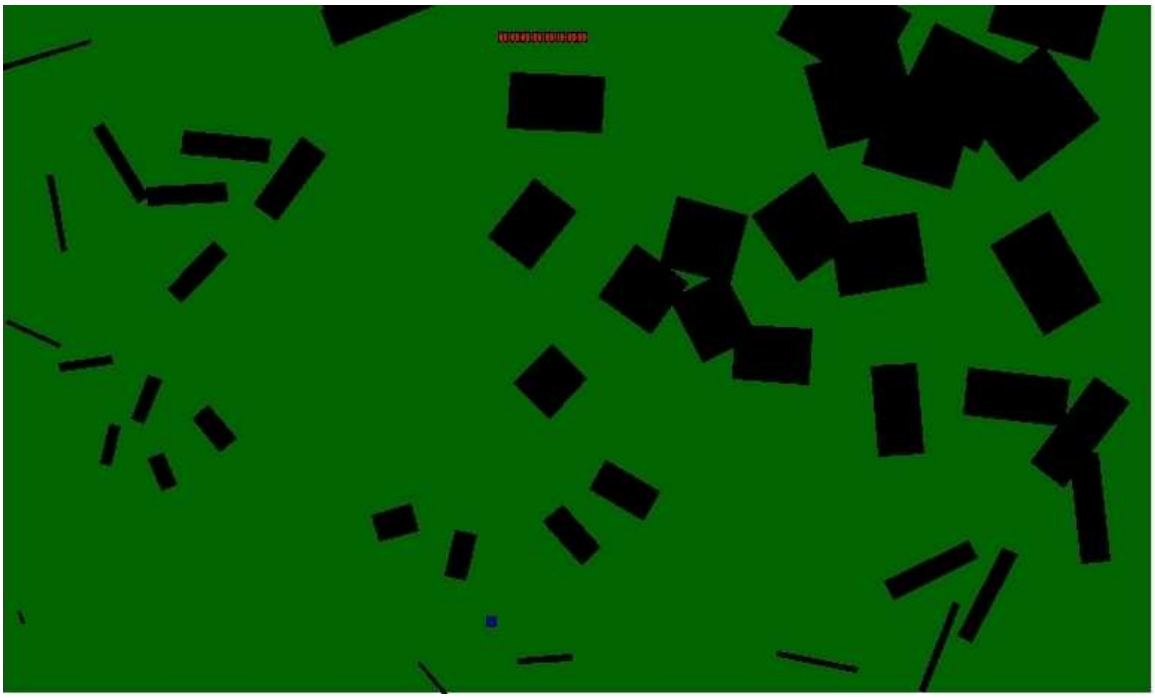
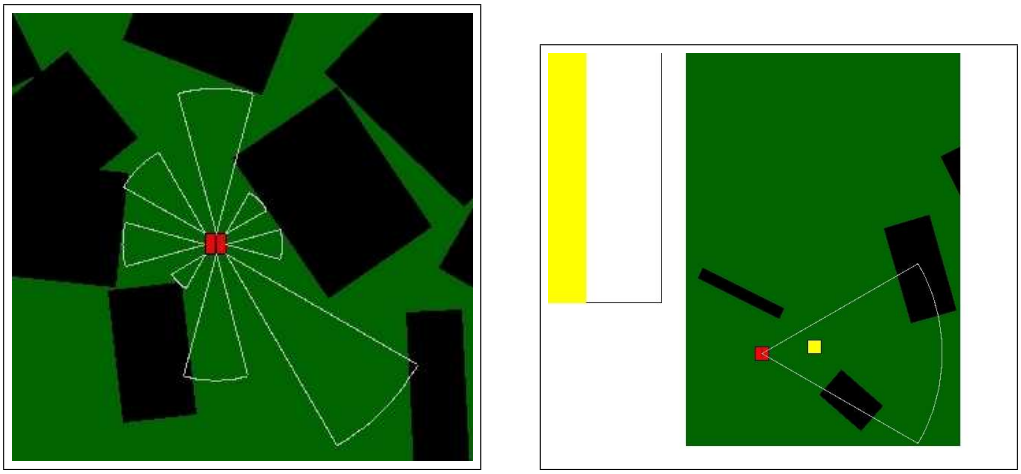


Figure 5.4: Environment 4. It consists of 8 robots (red squares), 1 target (blue square) and 45 obstacles (black rectangles)



(a) (b)

Figure 5.5: Simulated Miabot with (a) sonar and (b) blob-finder.

5.2.1 Noiseless Simulations

It is important to note that during the single and multi-target search tasks no noise was simulated in either the robots (e.g. odometry errors) or the environment (e.g. bad ultra-sonic reflections). In the real world, an ultra-sonic is a very noisy device that is affected by numerous aspects of the environment (e.g. shape of objects within the environment or the height of the transmitter from the ground and the frequencies used). In the simulation, ultra-sonic readings are correct at all times and boresight reflections never occur. Noise is introduced into the ultra-sonic readings during the third set of experiments (section 5.6) to investigate its effect on the performance of the potential field sharing systems. In the simulation, all objects within the environment are single uniform colours; in the real world, objects are multi-coloured and non-uniform and the blob-finder is heavily affected by the lighting conditions of the environment. In this noiseless simulation, once the blob-finder detects 1 pixel of the correct target RGB value, the target is considered to be discovered. Finally, errors in odometric readings are frequent and accumulative, due to wheel slippage and the inevitability that one of the motors will be more dominant than the other. However, in our simulation this is not the case and so the odometric readings are used to position the robots within the environment, relative from the starting location of robot 1, i.e. $(0, 0)$. An implicit form of noise is communications lag. The real robots use Bluetooth to communicate and all sensors are connected to a single bus on the robot. Both of these represent a possible point of slow throughput. Again, this real world phenomenon was not modelled in the simulation, although some lag may have occurred in the Player/Stage architecture (essentially a TCP/IP network), although not to the same extent as on the real robots.

It was decided to have a noiseless simulations in the first two sets of experiments in order to encourage fast (in that it was not necessary to build an accurate simulation from scratch) prototyping of the system, as one of the goals of this project was to implement the system on real robots in a laboratory setting. This methodology is common in robotics research (see [7]).

5.3 Statistical Analysis

Two statistical tests were chosen to analyse the data collected from our experiments. The Kruskal-Wallis test was chosen, as it is useful in detecting a difference in the medians of distributions. The Friedman test was chosen to detect the existence of association between characteristics of a population. Our performance metric was the time taken for the task to be completed. If the task was not completed within three hundred seconds, the task was assumed to have failed, and a result of three hundred seconds was recorded.

5.3.1 Kruskal-Wallis Rank Sum Test

The Kruskal-Wallis rank sum test involves ranking all of the times provided from each population. The mean of the sum of ranks for each was taken and the significant differences noted. The null hypothesis (h_0) and alternative hypothesis (h_1) were as follows:

- h_0: The medians of the k populations do not differ.
- h_1: At least two of the medians differ.

As detailed in [47], the Kruskal-Wallis H test statistic was produced whilst correcting for ranking ties. If the corrected H value was greater than the selected P-value (0.1) (this value is relatively large, due to the small sample size used throughout the experimentation in order to encourage the detection of significant results), then h_0 could be rejected in favour of h_1. If the differences between the means of ranks did not satisfy (5.1) and was negative, this meant that the first group had a significantly smaller completion time than the second group. However, if it was positive, it could be concluded that the second group had a significantly smaller completion time. This is because a larger mean difference relates to a longer completion time.

$$|\overline{R_i} - \overline{R_j}| \leq z\sqrt{\frac{k(N+1)}{6}}, \tag{5.1}$$

where k is the number of samples (3), N is the total sample size (60) — the sample size was small due to the time taken to complete the experimentation in real-time (approximately fours hours per sample). Finally, z is the *critical z value* for a level of significance of 0.1 (2.1 to 1 d.p.)

5.3.2 Friedman Rank Sum Test

The Friedman test involves ranking all of the samples provided. The sum of ranks for each was taken and the significant differences noted. This was repeated for all three environments. The null hypothesis (h_0) and alternative hypothesis (h_1) were set as follows:

- h_0: The number of R in the k populations has no effect.
- h_1: The number of R has an effect.

where R is the property of the system that is being investigated. As detailed in [47], the Friedman Q test statistic was calculated. If the Q value was greater than the selected P-value (0.1) (as previously stated this is a relatively high value due to the small sample size), then h_0 could be rejected in favour of h_1. If the difference between column sums did not satisfy (5.2) and was negative, this meant that the first group had a significantly larger completion time than the second group. If positive, it can be concluded that the first group had a significantly smaller completion time.

$$|R_i - R_j| \leq z\sqrt{\frac{kn(n+1)}{6}}, \tag{5.2}$$

where k is the size of a sample (20) — the sample size was small due to the practicalities of conducting the experimentation in real-time. n is the number of samples (7) and z is the *critical z value* for a level of significance of 0.1 (2.8 to 1 d.p.).

5.4 Single Target Search

In the single target experiment [12], the robots explored the environments shown in figures 5.1-5.3 with the exception that the yellow target was not included. Each

system (non-sharing, pessimistic and optimistic) completed the task in groups of two to eight robots over three different environments. Each group completed twenty runs. The mean completion times for each system, within each environment, are shown in table 5.1. The fastest time for a category is shown in **bold**. Full results are given in appendix A.1.

Table 5.1: Mean completion (seconds) for each system in each environment for 2-8 robots, to 1 d.p. The standard deviation is given in brackets, to 1 d.p.

	2	3	4	5
Environment 1				
non	300.0 (0.0)	250.7 (89.1)	255.9 (81.0)	225.1 (107.8)
pes	**203.4** (103.8)	211.1 (109.5)	172.6 (95.5)	133.9 (106.3)
opt	262.5 (74.3)	**198.8** (115.6)	**150.2** (110.0)	
Environment 2				
non	153.8 (123.0)	129.6 (104.3)	70.1 (84.9)	31.7 (24.0)
pes	**77.2** (102.6)	55.6 (62.4)	43.2 (62.0)	49.9 (67.8)
opt	132.1 (122.1)	**43.1** (23.1)	**31.7** (17.9)	**26.3** (19.8)
Environment 3				
non	**52.2** (6.2)	**91.2** (90.2)	69.6 (23.7)	**73.6** (54.7)
pes	115.5 (94.8)	141.8 (106.9)	**67.9** (11.5)	78.8 (53.0)
opt	115.5 (94.8)	128.5 (102.5)	70.5 (14.1)	105.7 (85.6)

	6	7	8
Environment 1			
non	205.2 (106.5)	273.0 (56.3)	288.2 (44.3)
pes	121.6 (107.3)	124.3 (105.9)	120.6 (95.3)
opt	**98.0** (89.0)	**110.2** (98.5)	**116.0** (98.6)
Environment 2			
non	32.4 (34.2)	33.2 (25.1)	29.9 (16.6)
pes	**8.0** (4.5)	**4.3** (1.0)	11.9 (11.3)
opt	11.4 (13.6)	**4.3** (0.6)	**9.5** (7.3)
Environment 3			
non	**62.7** (14.0)	112.7 (97.1)	134.4 (112.5)
pes	64.7 (9.7)	133.5 (112.3)	**95.9** (88.7)
opt	66.7 (13.3)	**106.6** (85.3)	110.0 (98.0)

By looking at table 5.1 it can be seen that in environment 1, the optimistic system performs the best in all but one category (2 robots). In environment 2, again the optimistic system performs best but, this time, in only six of the categories. In

environment 3, the non-sharing system performs best in four of the categories.

5.4.1 Comparison Across Systems

Table 5.2: Significant differences between the non-sharing ($R1$), pessimistic ($R2$) and optimistic ($R3$) systems (to 1 d.p.)

	$R12$	$R13$	$R23$
Two Robots			
env 1	15.5	*7.0*	*-8.6*
env 2	-16.6	*-7.4*	*9.2*
env 3	-27.9	-27.7	*0.3*
Three Robots			
env 1	*6.4*	*8.8*	*2.4*
env 2	15.7	19.2	*3.6*
env 3	-18.4	-14.7	*3.8*
Four Robots			
env 1	13.2	18.3	*5.2*
env 2	*6.1*	*8.1*	*2.1*
env 3	*-2.8*	*-6.9*	*-4.1*
Five Robots			
env 1	*11.7*	13.1	*1.4*
env 2	*-0.4*	*5.3*	*5.7*
env 3	*-9.0*	-13.3	*-4.3*
Six Robots			
env 1	14.3	17.6	*3.3*
env 2	23.6	22.0	*-1.6*
env 3	*-5.1*	*-5.6*	*-0.5*
Seven Robots			
env 1	20.4	22.6	*2.1*
env 2	30.4	29.5	*-1.0*
env 3	*-2.0*	*-4.0*	*-2.0*
Eight Robots			
env 1	22.2	25.2	*3.1*
env 2	22.6	26.7	*-4.1*
env 3	*4.4*	*0.8*	*-3.6*

Using the Kruskal-Wallis test described in the previous section, the following results are obtained. Table 5.2 shows the significant differences between the means of ranks for the non-sharing system ($R1$), pessimistic system ($R2$) and the optimistic

system ($R3$). Non-significant differences are shown in *italics*. As only values above 11.8 (to 1 d.p.) are significant, it can be seen that the pessimistic system performs better than the non-sharing system in groups of five or more in environments 1 and 2. This is clearly shown in figures 5.6 and 5.7, where points between the horizontal lines are insignificant, and points above the highest horizontal line represent instances when the sharing systems out-performed the non-sharing system. It can be seen that the optimistic system performs better than the non-sharing system, in groups of six or more in environments 1 and 2. For groups of less than six, the results are mixed. Again, this is clearly shown in figures 5.6 and 5.7. The non-sharing system performs significantly better in environment 3, for group sizes of up to five. This is clearly shown in figure 5.8, where points between the horizontal lines are insignificant and points below the lowest horizontal line represent instances when the non-sharing system out-performed the sharing systems. Beyond that, the differences become insignificant. It can also be observed that no significant difference between the pessimistic and optimistic systems performance occurred.

5.4.2 Comparison Across Size

Table 5.3: Significant differences between the numbers of robots, for the non-sharing system. (to 1 d.p.)

	2	3	4	5	6	7	8
Environment 2							
ind2	NA	3.0	**42.0**	**58.5**	**68.0**	**61.5**	**54.0**
ind3	-3.0	NA	**39.0**	**55.5**	**65.0**	**58.5**	**51.0**
Environment 3							
ind3	NA	-19.5	**-46.0**	-34.5	-32.5	**-55.0**	**-47.0**

To investigate what effect the number of robots had on performance, the Friedman rank sum test was conducted. Table 5.3 shows significant differences between the pairs of column sums for the non-sharing system. Significant differences are shown in **bold**. As only values above 38.6 (to 1 d.p.) are significant, it can be seen that in environment 2, the non-sharing system performs better with four or more agents.

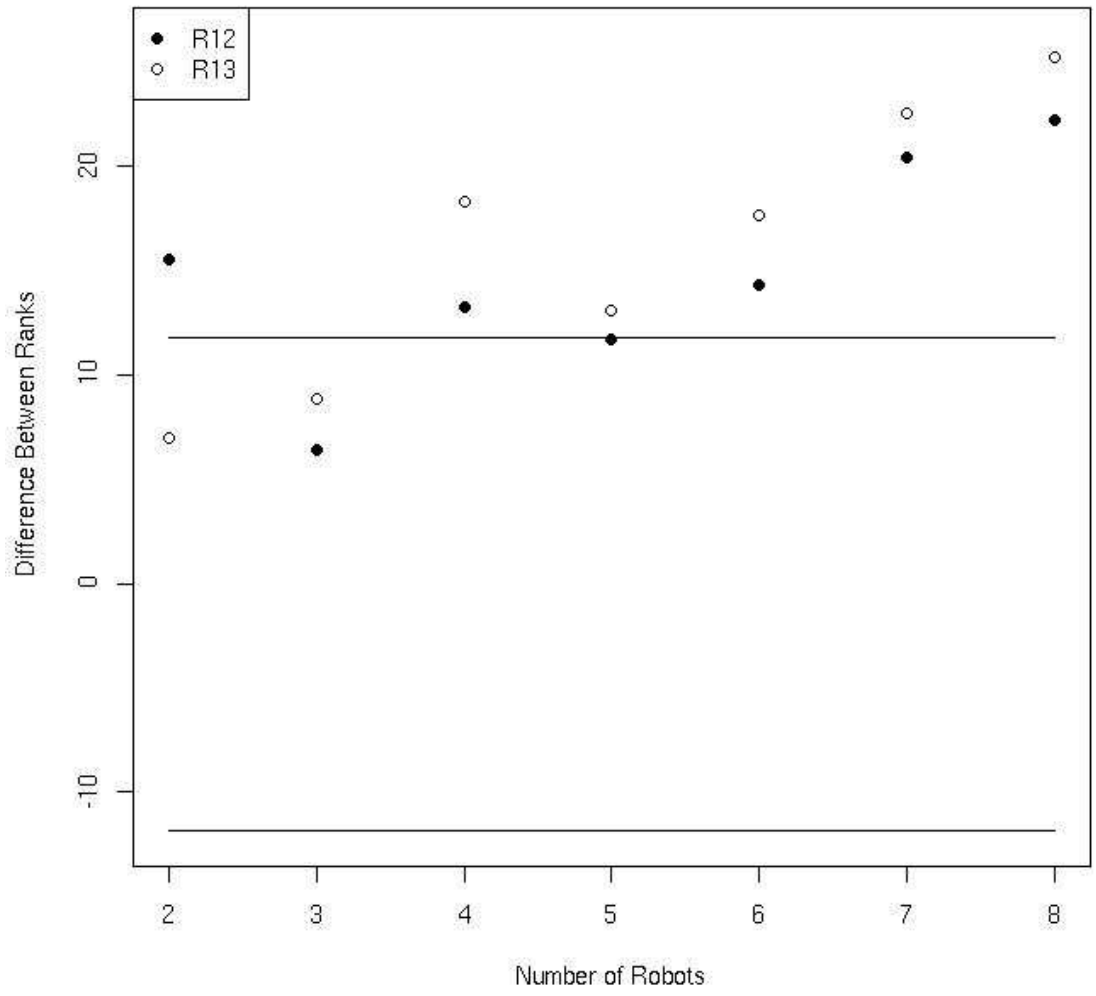


Figure 5.6: Differences between ranks for environment 1 (single target): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

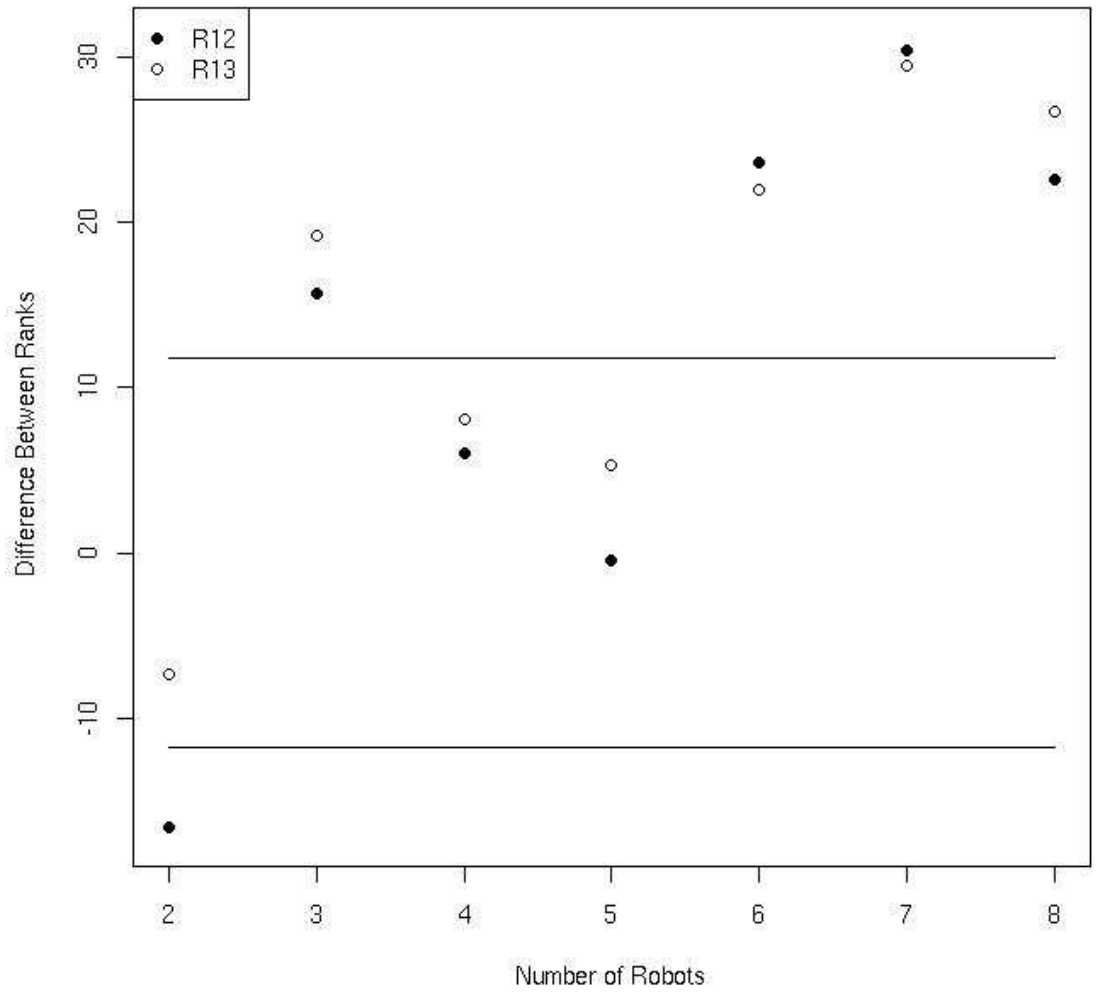


Figure 5.7: Differences between ranks for environment 2 (single target): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

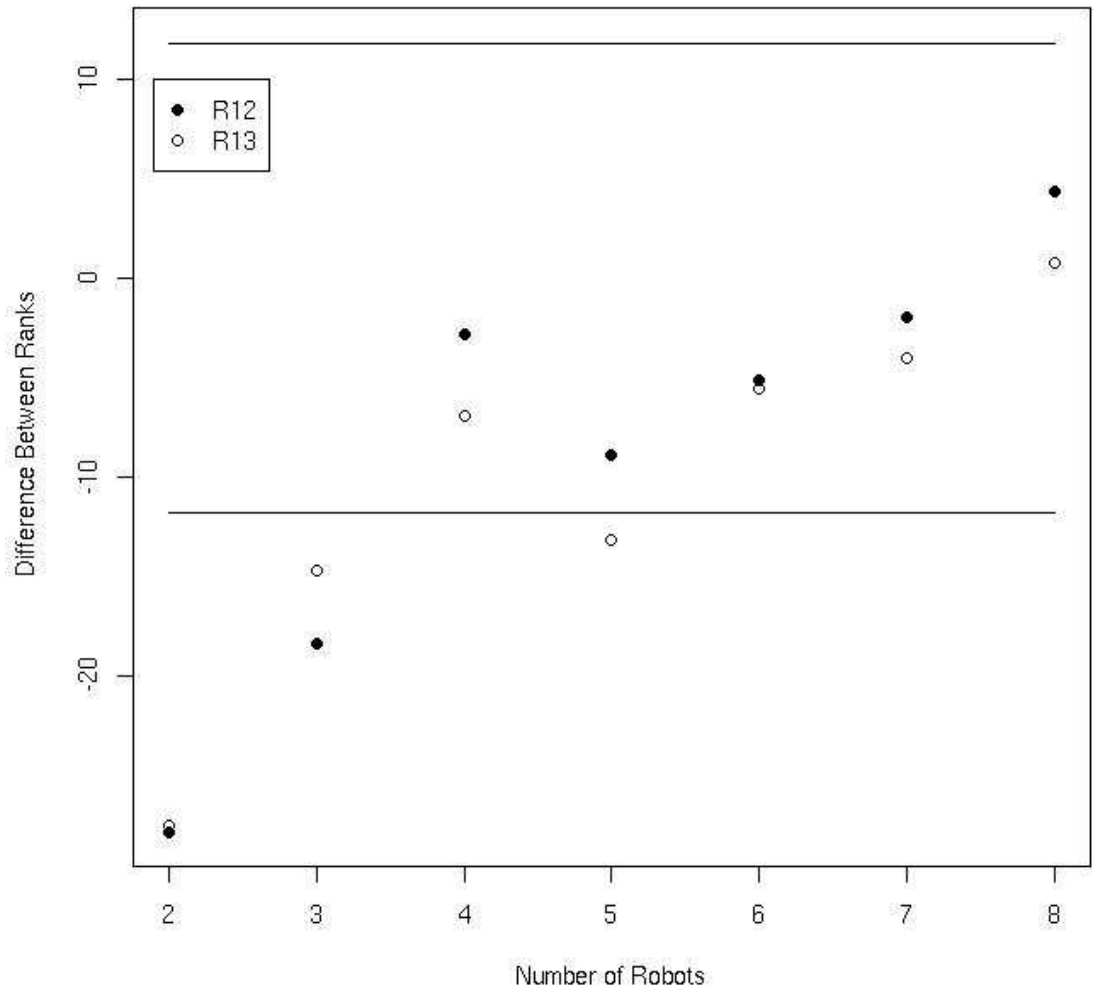


Figure 5.8: Differences between ranks for environment 3 (single target): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

Table 5.4: Significant differences between the numbers of robots, for the pessimistic system. (to 1 d.p)

	2	3	4	5	6	7	8
Environment 1							
pes2	NA	1.5	12.0	36.0	**41.5**	33.5	36.0
pes3	-1.5	NA	10.5	24.5	**40.0**	32.0	34.5
Environment 2							
pes2	NA	-8.0	6.5	15.5	**52.0**	**83.0**	**54.0**
pes3	8.0	NA	14.5	23.5	**60.0**	**91.0**	**62.0**
pes4	-6.5	-14.5	NA	9.0	**45.5**	**76.5**	**47.5**
pes5	-15.5	-23.5	-9.0	NA	36.5	**67.5**	38.5
Environment 3							
pes3	18.5	NA	36.0	36.0	**45.5**	28.5	**45.5**

Table 5.5: Significant differences between the numbers of robots, for the optimistic system. (to 1 d.p.)

	2	3	4	5	6	7	8
Environment 1							
opt2	NA	19.0	38.5	**47.5**	**61.5**	**56.0**	**57.5**
opt3	-19.0	NA	19.5	28.5	**42.5**	37.0	38.5
Environment 2							
opt2	NA	5.5	18.0	34.5	**61.0**	**89.0**	**72.0**
opt3	-5.5	NA	12.5	29.0	**55.5**	**83.5**	**66.5**
opt4	-18.0	-12.5	N/A	16.5	**43.0**	**71.0**	**54.0**

However, for environment 3, the non-sharing system performs better with two agents only — this is clearly shown in figure 5.10. It can also be observed that group size had no significant impact in environment 1. Table 5.4 shows the same data for the pessimistic system (significant differences are shown in **bold**). It appears to perform better with six or more agents in all three environments (which is in line with the results from table 5.2). These results are shown in figures 5.9, 5.10 and 5.11. Table 5.5 again shows the same data for the optimistic system (significant differences are shown in **bold**), which performs better in environments 1 and 2 with five or more agents. These results are clearly shown in figures 5.9 and 5.10. There were no significant differences in environment 3, as shown in figure 5.11.

5.4.3 Discussion of Single Target Results

From the results in this section, it can be observed that both the pessimistic and optimistic systems perform better than the non-sharing system (when in groups of 6 or more). It can also be observed that both the pessimistic and optimistic systems perform better in larger group sizes (6 or more and 5 or more respectively). These results are as expected, as the presence of more robots implies a higher probability that the robots can take advantage of potential field sharing, instead of reverting to the non-sharing behaviour. The second observation is interesting as it shows that a system less inclined to avoid obstacles performs better with a smaller group size than a system that is inclined to avoid obstacles. This also makes sense as the smaller the group size, the less likely it is that more (previously unseen) obstacles will be discovered. The observation that the performance of the pessimistic and optimistic systems did not differ significantly is also interesting as this implies that the ability to detect more obstacles has the same benefit as the ability to ignore more sensor noise. This result, however, is not a surprise as little or no noise was present within the simulation.

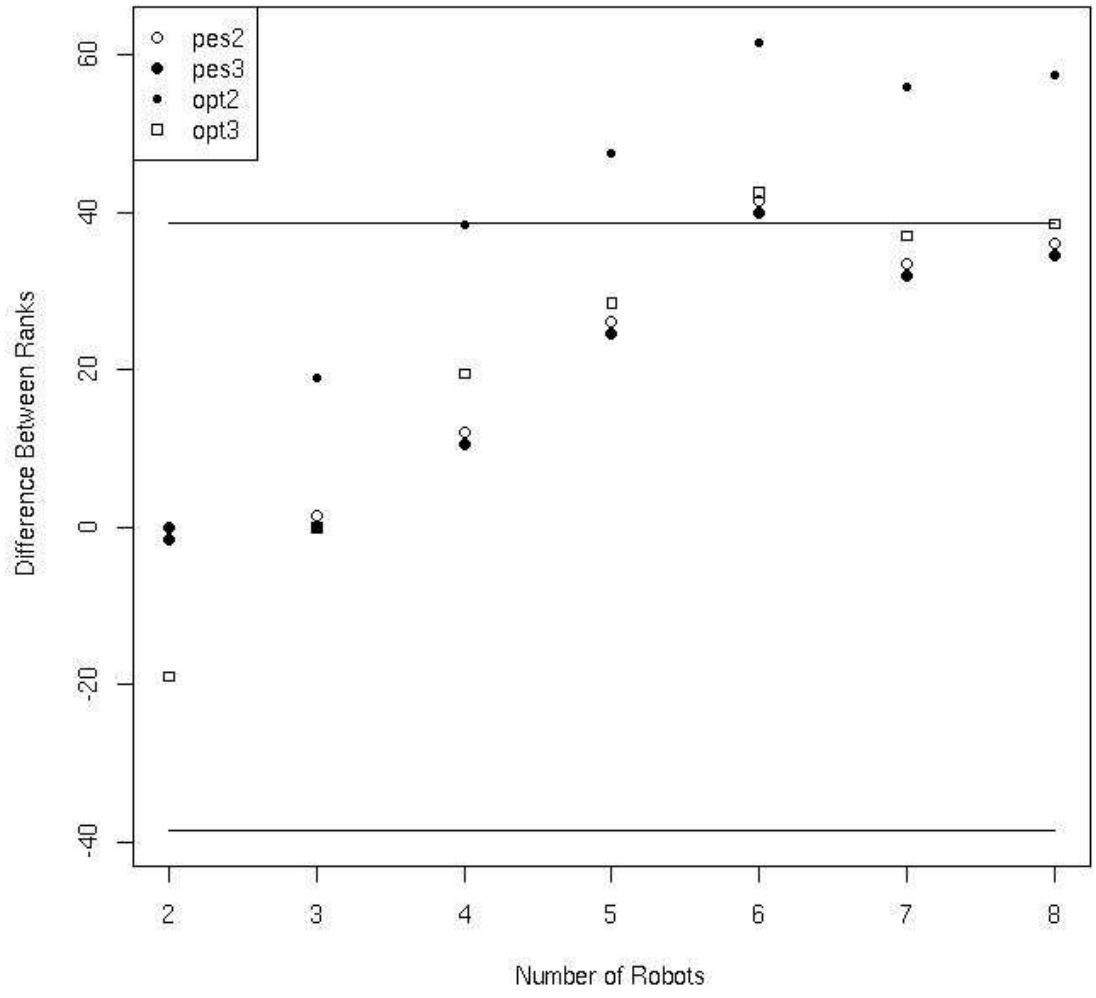


Figure 5.9: Differences between ranks for environment 1 (single target): pes2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the pessimistic system. pes3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the pessimistic system. opt2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the optimistic system. opt3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the optimistic system. Points in between the two horizontal lines are not significant.

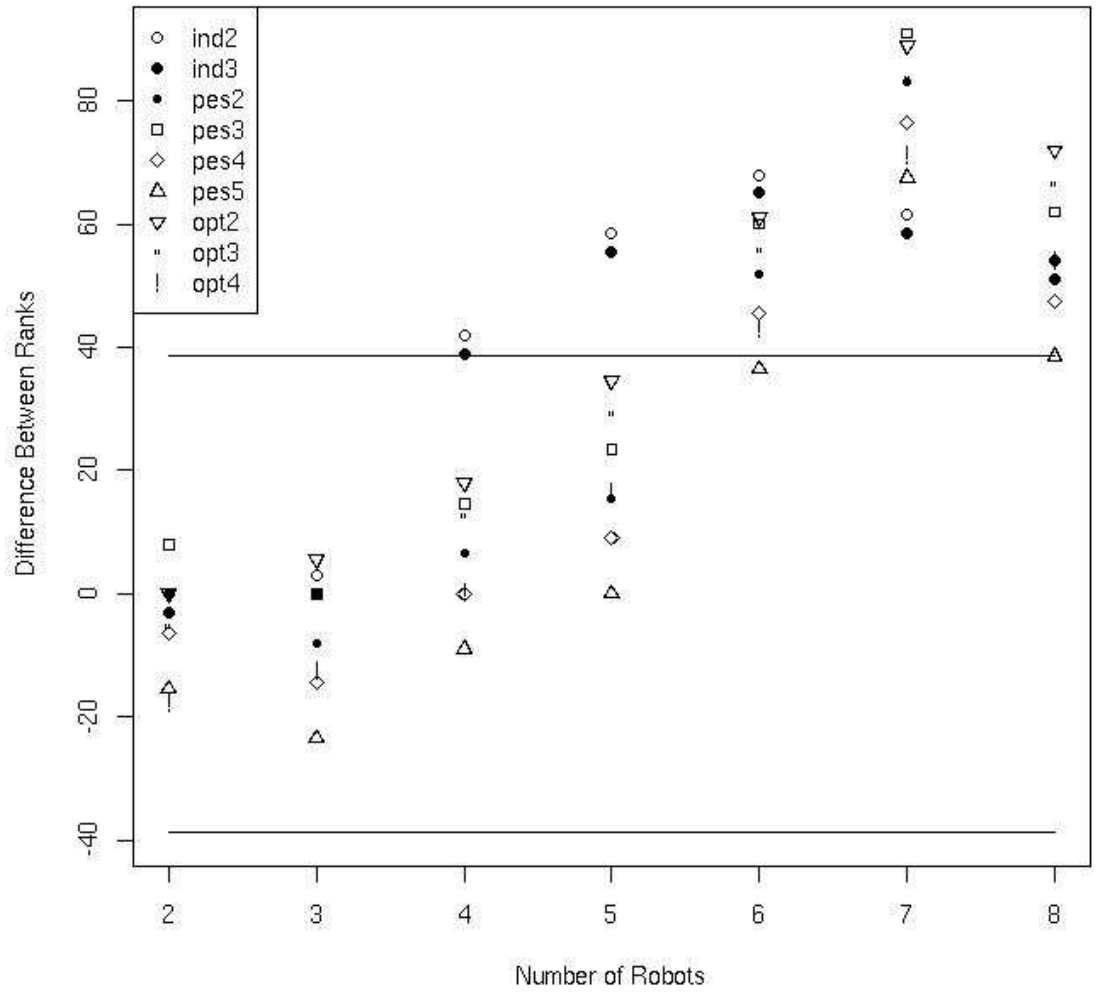


Figure 5.10: Differences between ranks for environment 2 (single target): ind2/ind3 — Differences between the performance of 2/3 robots compared to 2 to 8 robots using the non-sharing system. pes2/pes3/pes4/pes5- Differences between the performance of 2/3/4/5 robots compared to 2 to 8 robots using the pessimistic system. opt2/opt3/opt4- Differences between the performance of 2/3/4 robots compared to 2 to 8 robots using the optimistic system. Points in between the two horizontal lines are not significant.

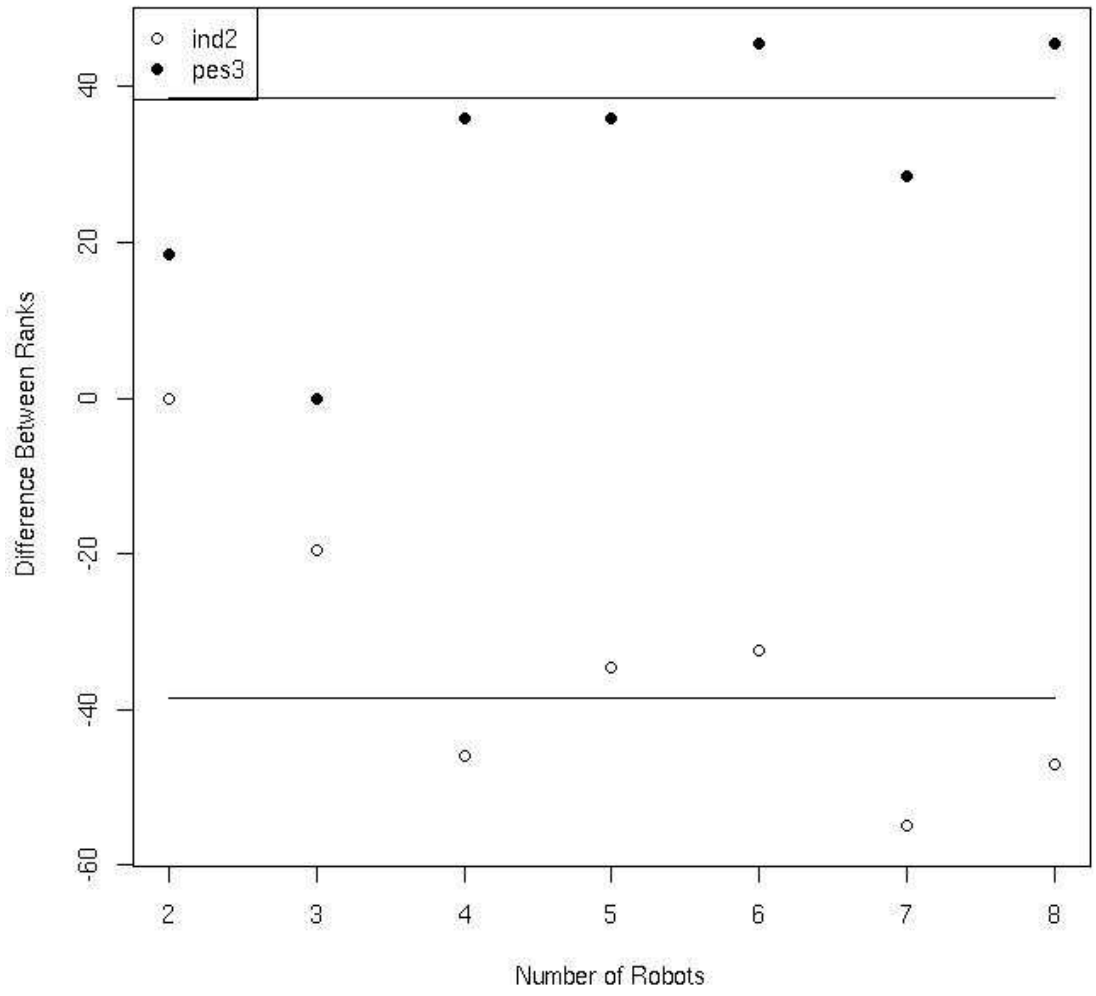


Figure 5.11: Differences between ranks for environment 3 (single target): ind2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the non-sharing system. pes3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the pessimistic system. Points in between the two horizontal lines are not significant.

5.5 Multi-target Search

In the multi-target experiment [13], the robots explored the environments shown in figures 5.1-5.3. As with the single target search, each system (non-sharing, pessimistic and optimistic) completed the task in groups of two to eight robots over three different environments. Each group completed twenty runs. The mean completion times for each system, within each environment, are shown in table 5.6. The fastest time for a category is shown in **bold**. Full results are given in appendix A.2.

Table 5.6: Mean completion (seconds) for each system in each environment for 2-8 robots, to 1 d.p. The standard deviation is given in brackets, to 1 d.p.

	2	3	4	5
Environment 1				
non	300.0 (0.0)	195.0 (113.5)	284.0 (48.3)	241.1 (88.7)
pes	**184.5** (113.1)	**183.5** (120.2)	**104.8** (86.2)	135.1 (103.6)
opt	195.0 (113.5)	188.5 (116.0)	149.5 (105.6)	**117.7** (98.7)
Environment 2				
non	123.3 (105.9)	127.1 (61.0)	85.0 (111.8)	73.9 (83.4)
pes	**70.2** (101.0)	**48.5** (30.8)	53.8 (85.5)	62.3 (65.8)
opt	76.5 (100.7)	68.6 (64.1)	**52.4** (61.9)	**45.7** (64.6)
Environment 3				
non	249.0 (104.7)	173.2 (130.2)	163.3 (115.4)	**104.6** (71.5)
pes	**116.2** (63.6)	**94.5** (16.2)	**135.7** (88.3)	114.5 (83.3)
opt	135.4 (86.0)	110.0 (67.2)	163.6 (104.9)	111.8 (83.4)

	6	7	8
Environment 1			
non	189.2 (98.6)	250.0 (83.5)	163.3 (118.7)
pes	126.3 (101.8)	165.0 (112.6)	**104.0** (85.4)
opt	**85.0** (76.0)	**138.9** (109.5)	139.1 (109.9)
Environment 2			
non	**25.9** (14.2)	37.1 (62.0)	26.7 (17.9)
pes	37.7 (21.0)	**28.0** (17.8)	**25.4** (12.0)
opt	42.2 (62.9)	38.4 (62.8)	40.6 (61.7)
Environment 3			
non	170.3 (102.5)	198.5 (115.3)	103.5 (87.0)
pes	**98.8** (71.3)	103.6 (86.6)	83.3 (59.5)
opt	118.6 (94.6)	**76.3** (40.9)	**75.4** (58.2)

By looking at table 5.6, it can be seen that in environment 1, the pessimistic system performs best with two to four robots, whilst the optimistic system performs better with five to seven robots. In environment 2, the pessimistic system performs best with two to three robots and seven to eight robots, whilst the optimistic system performs better with four to five robots. In environment 3, again the pessimistic system performs best with two to four robots, whilst the optimistic system performs better with seven to eight robots.

5.5.1 Comparison Across Systems

Using the Kruskal-Wallis test described in section 5.3.1, the following results are obtained. Table 5.7 shows that in environment 1 both the pessimistic and the optimistic system perform significantly better than the non-sharing system in all but one case (3 robots). Non-significant differences are shown in *italics*. The results are clearly visible in figure 5.12. In environment 2, the pessimistic system significantly outperforms the non-sharing system in the 2 and 3 robot cases. The optimistic system significantly outperforms the non-sharing system in the 2, 3 and 5 robot cases. The results are clearly visible in figure 5.13. In environment 3, both the sharing systems significantly outperformed the non-sharing system in the 2 and 7 robot cases. This is shown in figure 5.14. As with the single target case, the sharing system's performance did not differ significantly.

Table 5.7: Significant differences between the non-sharing ($R1$), pessimistic ($R2$) and optimistic ($R3$) systems for the multi-target case (to 1 d.p.)

	$R12$	$R13$	$R23$
Two Robots			
env 1	16.5	15.1	*-1.4*
env 2	17.8	15.1	*-2.7*
env 3	17.1	*11.7*	*-5.4*
Three Robots			
env 1	*2.1*	*0.9*	*-1.2*
env 2	24.8	19.7	*-5.1*
env 3	*-0.8*	*-0.7*	*0.1*
Four Robots			
env 1	26.4	17.8	*-8.6*
env 2	*10.2*	*2.8*	*-7.4*
env 3	*-2.1*	*-5.4*	*-3.3*
Five Robots			
env 1	15.7	18.4	*-2.7*
env 2	*2.4*	11.8	*9.4*
env 3	*-1.6*	*0.1*	*1.7*
Six Robots			
env 1	11.8	20.2	*8.4*
env 2	*-5.6*	*3.8*	*9.4*
env 3	*11.0*	*10.4*	*-0.6*
Seven Robots			
env 1	13.0	15.9	*2.9*
env 2	*15.6*	*7.9*	*2.3*
env 3	16.5	19.9	*3.3*
Eight Robots			
env 1	*2.7*	*0.7*	*-2.1*
env 2	*-0.6*	*-6.2*	*-5.6*
env 3	*2.2*	*7.8*	*5.6*

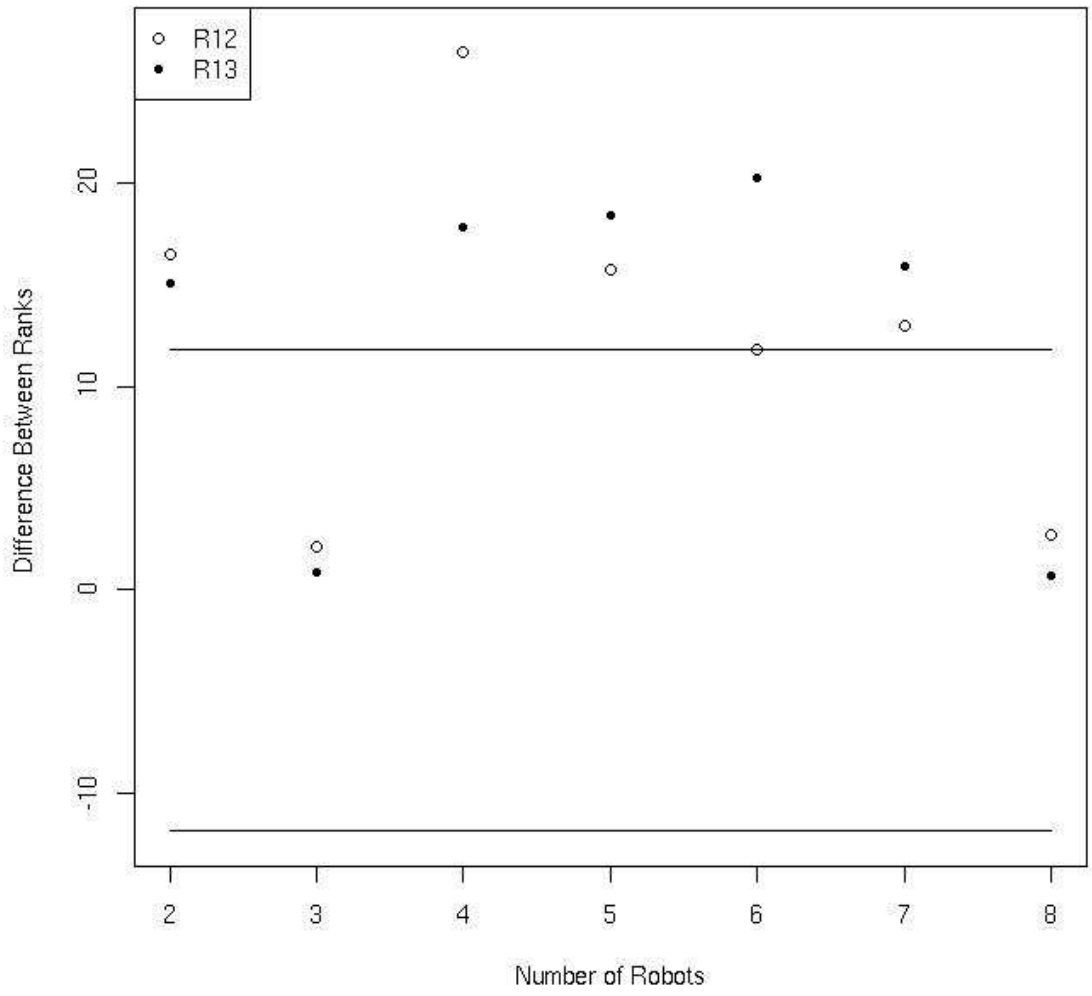


Figure 5.12: Differences between ranks for environment 1 (multiple targets): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

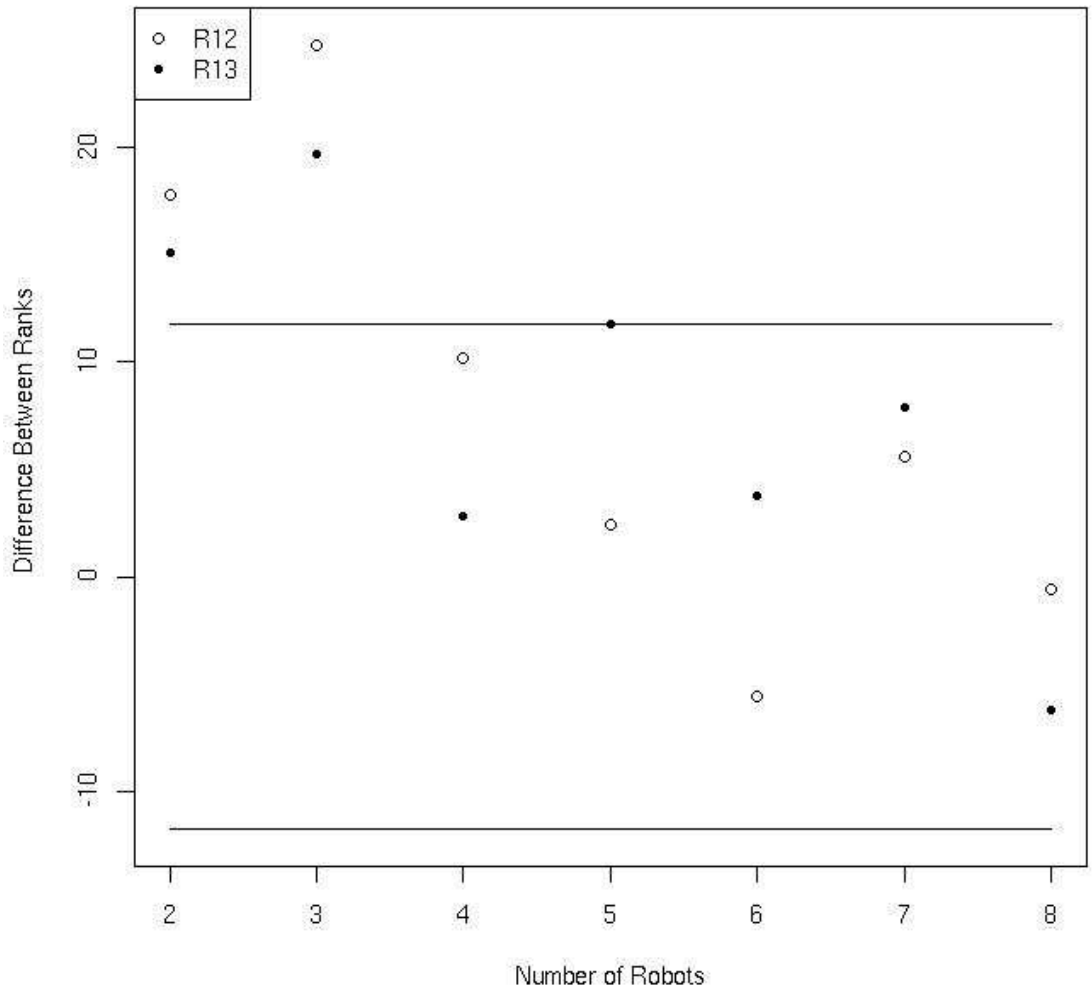


Figure 5.13: Differences between ranks for environment 2 (multiple targets): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

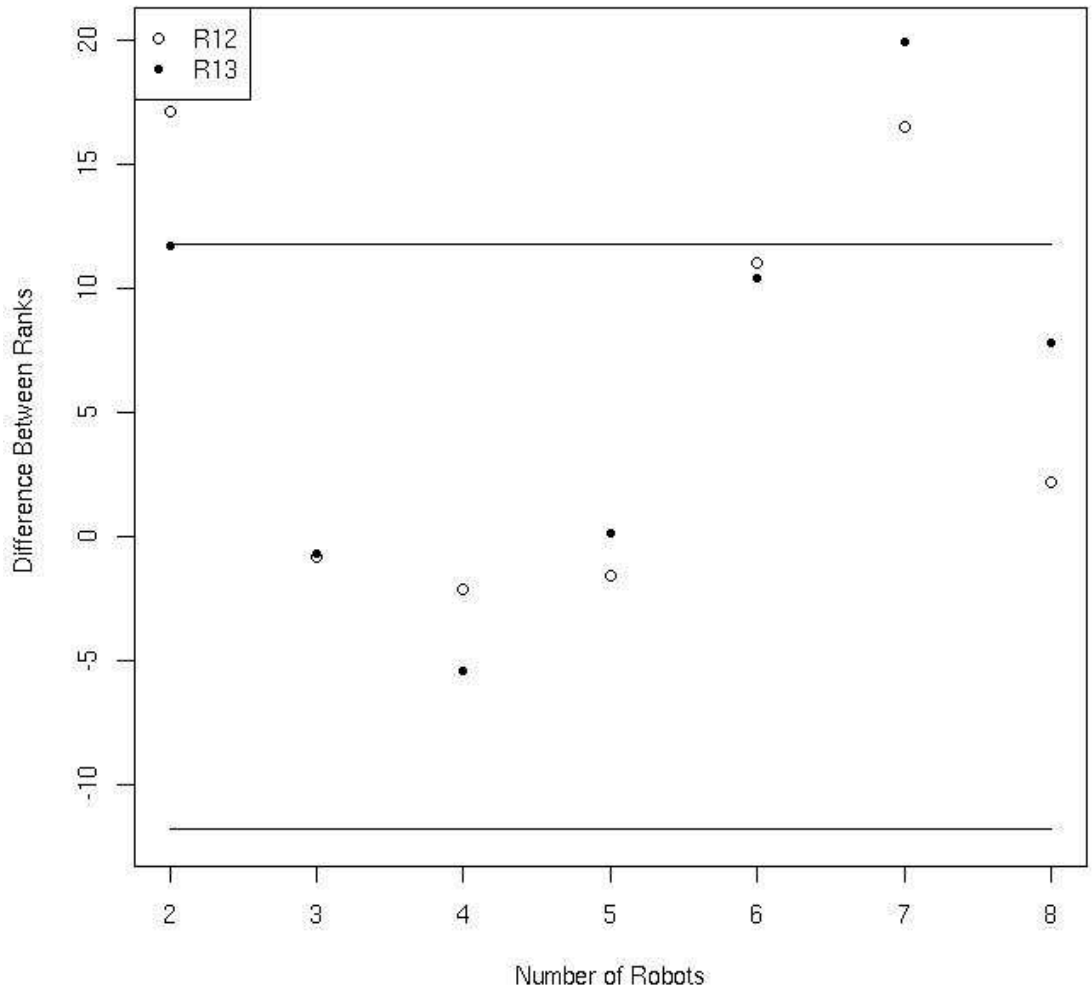


Figure 5.14: Differences between ranks for environment 3 (multiple targets): R12 — differences between the performance of the non-sharing system and the pessimistic system. R13 — differences between the performance of the non-sharing system and the optimistic system. Points in between the two horizontal lines are not significant.

5.5.2 Comparison Across Size

Once again, using the Friedman test described in section 5.3.2, the following results are obtained. Table 5.8 shows that the non-sharing system performed better with 6 or more robots in environments 1 and 2. Significant differences are shown in **bold**. The pessimistic system results suggest no significant advantage can be gained by increasing the number of robots within the system. Table 5.9 shows that the optimistic system performed best with 6 or more robots in environments 1 and 2. Significant differences are shown in **bold**. The results discussed are clearly visible in figures 5.15, 5.16 and 5.17.

Table 5.8: Significant differences between the numbers of robots, for the non-sharing system. For the multi-target case. (to 1 d.p.)

	2	3	4	5	6	7	8
Environment 1							
ind2	NA	36.0	9.0	25.5	**41**	19.5	**47.5**
Environment 2							
ind2	NA	-16.0	33.0	27.5	**59.0**	**50.5**	**49.0**
ind3	16.0	NA	**49.0**	**43.5**	**75.0**	**66.5**	**65.0**
Environment 3							
ind2	NA	28.0	19.5	29.5	12.5	11.5	**39.0**

Table 5.9: Significant differences between the numbers of agents, for the optimistic system. For the multi-target case. (to 1 d.p)

	2	3	4	5	6	7	8
Environment 1							
opt2	NA	-1.5	10,5	27.5	**42.5**	15.5	17.5
opt3	1.5	NA	12.0	29.0	**44.0**	17.0	19.0
Environment 2							
opt3	28.5	NA	20.5	36.5	38.0	**39.5**	26.0
Environment 3							
opt2	NA	9.5	-2.5	18.5	22.5	**48.0**	**54.5**
opt3	-9.5	NA	-13.0	9.0	13.0	38.5	**45.0**
opt4	2.5	12.0	NA	21.0	25.0	**50.5**	**57.0**

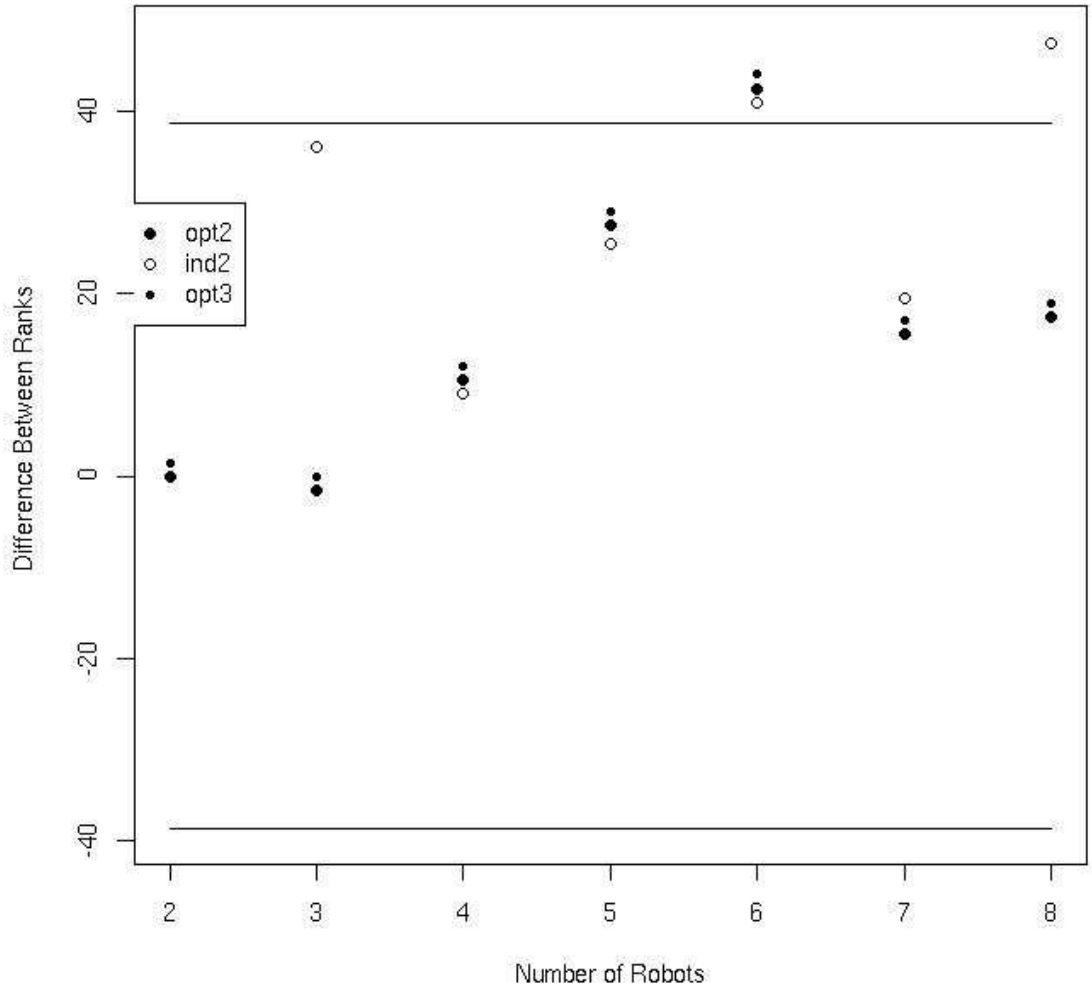


Figure 5.15: Differences between ranks for environment 1 (multiple targets): ind2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the non-sharing system. opt2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the optimistic system. opt3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the optimistic system. Points in between the two horizontal lines are not significant.

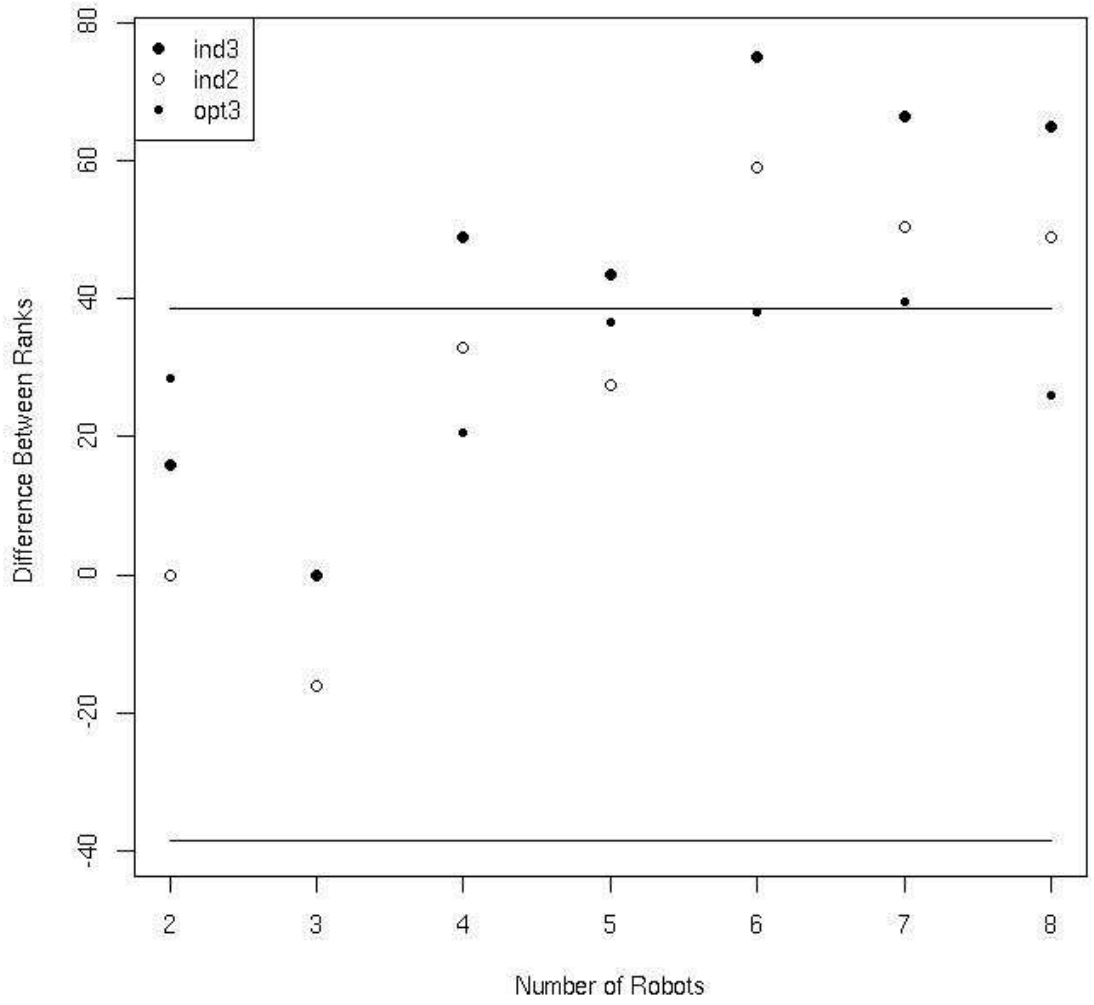


Figure 5.16: Differences between ranks for environment 2 (multiple targets): ind2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the non-sharing system. ind3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the non-sharing system. opt3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the optimistic system. Points in between the two horizontal lines are not significant.

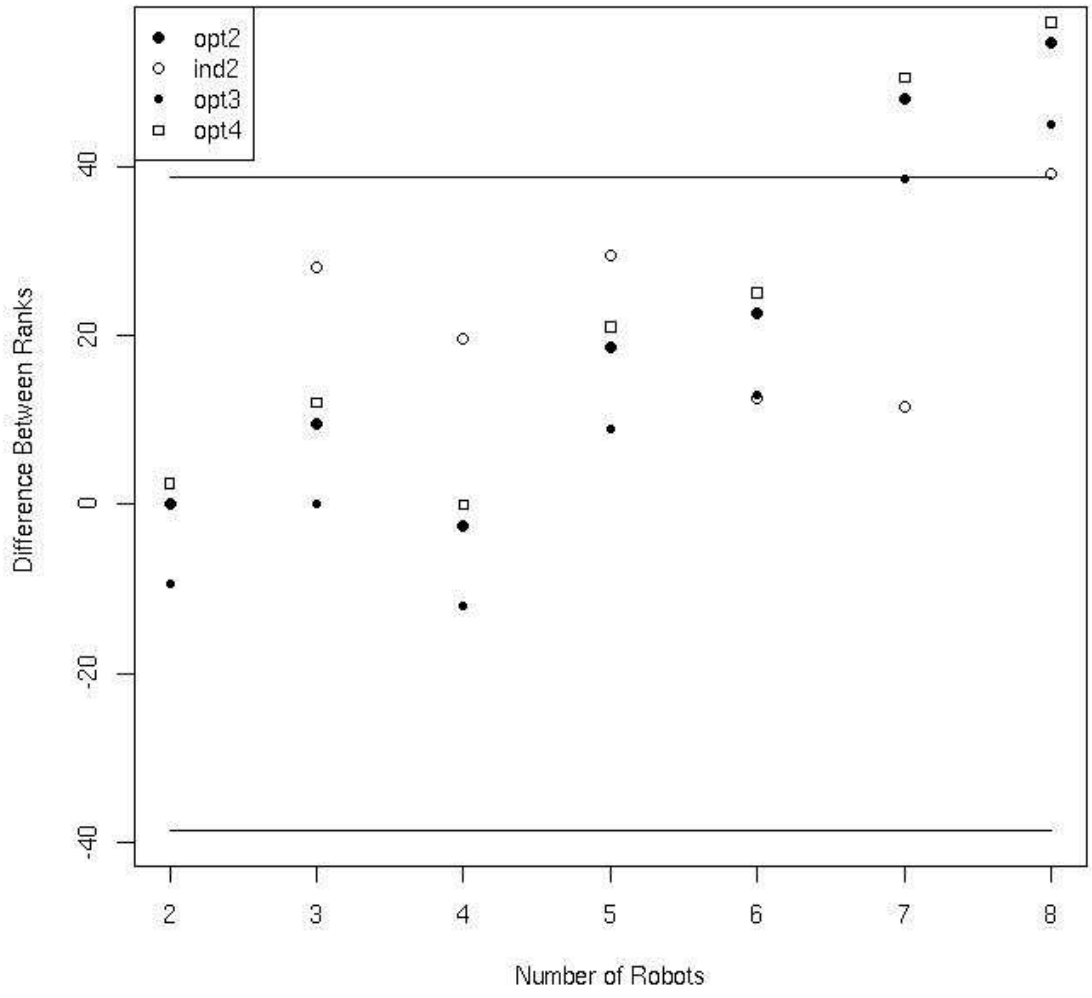


Figure 5.17: Differences between ranks for environment 3 (multiple targets): ind2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the non-sharing system. opt2 — Differences between the performance of 2 robots compared to 2 to 8 robots using the optimistic system. opt3 — Differences between the performance of 3 robots compared to 2 to 8 robots using the optimistic system. opt4 — Differences between the performance of 4 robots compared to 2 to 8 robots using the optimistic system. Points in between the two horizontal lines are not significant.

5.5.3 Discussion of Multi-target Results

The results show that the sharing systems perform significantly better than the non-sharing system. As with the single target case, it was observed that the performance of the pessimistic and optimistic systems did not differ significantly. Again, it is believed that this may be due to the limited amount of noise within the simulation and that the differences in the sharing systems' performance will be more apparent in the real world. The observation that both the non-sharing and the optimistic system perform better with six or more robots makes sense as, in the case of the non-sharing system, more robots in the environment results in a greater area coverage. In the case of the optimistic system, as well as having the same benefits of the non-sharing system, more robots result in more information being shared and so each robot can make better decisions. The observation that the pessimistic system did not benefit from an increased number of robots was not expected and requires further research.

5.6 Single-target Search with Noisy Sensor Readings

In the final set of simulation experiments, the effect of noise within the sensor readings and an increase in group size is investigated [10]. As with the single target search, the two potential field sharing systems (pessimistic and optimistic) had to complete the task of finding an target within the environment. In this experiment only group sizes of eight and sixteen robots were investigated, completing twenty runs each on a single environment (Environment 4 - figure 5.4).

Noise was intorduced into the ultra-sonic readings of each robot. This was achieved by adding guassian noise to the initial reading of each ultra-sonic sensor. Three levels of noise were chosen to be investigate 0.01, 0.05 and 0.1 these values added noise normally distributed within a range of $+/-1cm$, $+/-5cm$ and $+/-10cm$ respectively to each sensor reading.

The mean completion times for each system, are shown in table 5.10. The fastest time for a category is shown in **bold**. Full results are given in appendix A.3.

Table 5.10 clearly shows that apart from the 16 robot pessimistic case the perfor-

Table 5.10: Mean completion (seconds) for each system with varying levels of sensor noise for 8 and 16 robots, to 1 d.p. The standard deviation is given in brackets, to 1 d.p.

	Low	Mid	High
Optimistic			
8	69.1 (48.0)	55.7 (28.3)	**43.0** (5.9)
16	46.1 (18.4)	43.6 (11.9)	**42.6** (5.2)
Pessimistic			
8	76.4 (57.2)	49.0 (13.7)	**44.8** (7.4)
16	**43.9** (14.2)	47.9 (15.9)	46.4 (11.7)

Table 5.11: Kruskal-Wallis rank sum test. Differences between means, pessimistic system, to 1 decimal place.

	low	mid	high
opt (8 - 16)	-6.8	-5.9	-0.7
pes (8 - 16)	**-12.7**	-1.6	-3.4

mance of the system increases with an increase in sensor noise.

5.6.1 Comparison Across Size

The Kruskal-Wallis was used to compare the performace of a system across group size. As in the previous sections, the H statistic was calculated and then the differences in means compared. In this case the following values where substituted into equation 4.2: $k = 2$, $N = 40$ and $z = 1.96$. Therefore, the differences were significant if they were greater than or equal to 7.2 (to 1 decimal place).

It is clear to see from table 5.11 that the effect of group size on the optimistic system's performance was not significant. For the pessimistic system, group size only had a significant effect when the level of sensor noise was low, with 8 robots out-performing 16 robots.

Table 5.12: Kruskal-Wallis rank sum test. Differences between means, to 1 decimal place.

	8	16
Optimistic		
Low - Mid	2.2	-0.3
Low - High	**14.6**	2.7
Mid - High	**12.4**	3.0
Pessimistic		
Low - Mid	**15.6**	-3.0
Low - High	**20.8**	-5.0
Mid - High	5.2	-2.0

5.6.2 Comparison Across Noise

Using the same values in the Kruskal-Wallis test as in section 5.6.1. The effect of noise on the shared potential method was also analysed.

It is shown in table 5.12 that the optimistic system performed the best when high levels of sensor noise were introduced. It also shows that the pessimistic systems performance was worst when there was only low levels of sensor noise.

5.6.3 Discussion of Single-target with Noise Results

The results show that the pessimistic system performed better with 8 robots when a low level of sensor noise was introduced. It is believed that an increase in the number of robots within the pessimistic system causes the group of robots to become more cautious within the environment, and thus take longer to complete a given task. Higher levels of noise counteract this affect. It is also shown that both the sharing systems performed better with the higher levels of sensor noise. It is beleived this is due to addition of sensor noise causing the systems to avoid cases of local minima, and so enabled them to navigate the environment faster. For example, not getting "stuck" in 'U' shaped parts of the environment.

5.7 Summary

In this chapter, the simulated experimental setup has been discussed. The potential field sharing system was compared against a non-sharing system over three different environments with varying group sizes. In order to aid fast prototyping no noise was added to the simulation. The effect of noise on the systems was left for the investigation on real robots.

In both sets of experiments, single target and multi-target, the behaviour of the systems was similar. This is due to the fact that the addition of targets merely increases the number of obstacles in the environment and, therefore, only affects performance (in terms of task completion time) rather than behaviour. It was shown that the pessimistic system significantly outperformed the non-sharing system with group sizes of five or more robots and that the optimistic system outperformed the non-sharing system with six or more robots.

Larger group sizes increased the performance of all the systems; the non-sharing system's performance increased with 4 or more robots when compared to the performance of just 2 or 3 robots. The pessimistic system only showed an increase in performance in the one target case, where groups of 6 or more robots outperformed groups of 2 to 4 robots. The optimistic system's performance increased when in groups of 6 or more robots when compared against groups of just 2 to 4 robots.

When noise was introduced to the simulation, both of the sharing systems performed better with the higher levels of noise. It is beleived that the increase in sensor noise allows the system to avoid areas of local minima within the potential feild. This enables the sytems to navigate the environment faster.

The experiments, presented in this chapter, have helped this project meet one of its goals given in chapter 1.

- **To design and implement a multi-robot system that is not reliant upon explicit information gathered from other robots.**

  The potential field sharing method did not at anytime, during the simulation experiments, explicitly co-ordinate the team members. Co-ordination was an

emergent property of combined potential fields.

The findings from the experiments discussed in this chapter were published in the following conference proceedings and book chapter.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "The Effect of Potential Field Sharing in Multi-Agent Systems", *In the proceedings of 3rd International Conference on Autonomous Robots and Agents (ICARA 2006)*, Palmerston North, New Zealand, 12th-14th December, pp 33-38, 2006.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "Multi-Robot Search and Rescue: A Potential Field Based Approach", in *Autonomous Robots and Agents*, series: Studies in Computational Intelligence book series, Vol. 76, Mukhopadhyay, Subhas; Sen Gupta, Gourab (Eds.), Springer-Verlag, pp 9-16, 2007.

The simulation investigation into the affect of sensor noise on the systems is presented in a journal paper under review.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi, S. Groenemeyer & M. Norman, "Multi-robot Co-ordination Using Shared Potential Fields", **submitted to** *IEEE Transactions on Robotics*, 2009.

Chapter 6

Laboratory Search Problems

6.1 Introduction

After the completion of the simulated experiments, the next phase was to run a number of experiments on real robots in a laboratory environment. In this chapter, the Miabot (Mobile Intelligent Autonomous Robot) and the laboratory setup used in the experiments in this chapter and chapter 7 are described. In the experiments in this chapter, the robots had the same task to complete as in the simulated single target search problem (see chapter 5). However, the size of the environments was considerably smaller due to restrictions in the overhead camera system (discussed in further detail in section 6.3). Results for these experiments are shown and analysed, and the chapter ends with a summary.

6.2 Robot Specification

The type of robot used in the experimentation is a Merlin Miabot Pro (more information is available in section 3.2), see figure 6.1. Each robot was approximately $18cm \times 8cm \times 8cm$ and was equipped with the following sensors/actuators:

- *Position2d*: a differential steering drive, with an optical encoder resolution of $0.04mm$ and a maximum reported speed of $3.5m/s$. As in the simulation, the Miabot is non-holonomic.

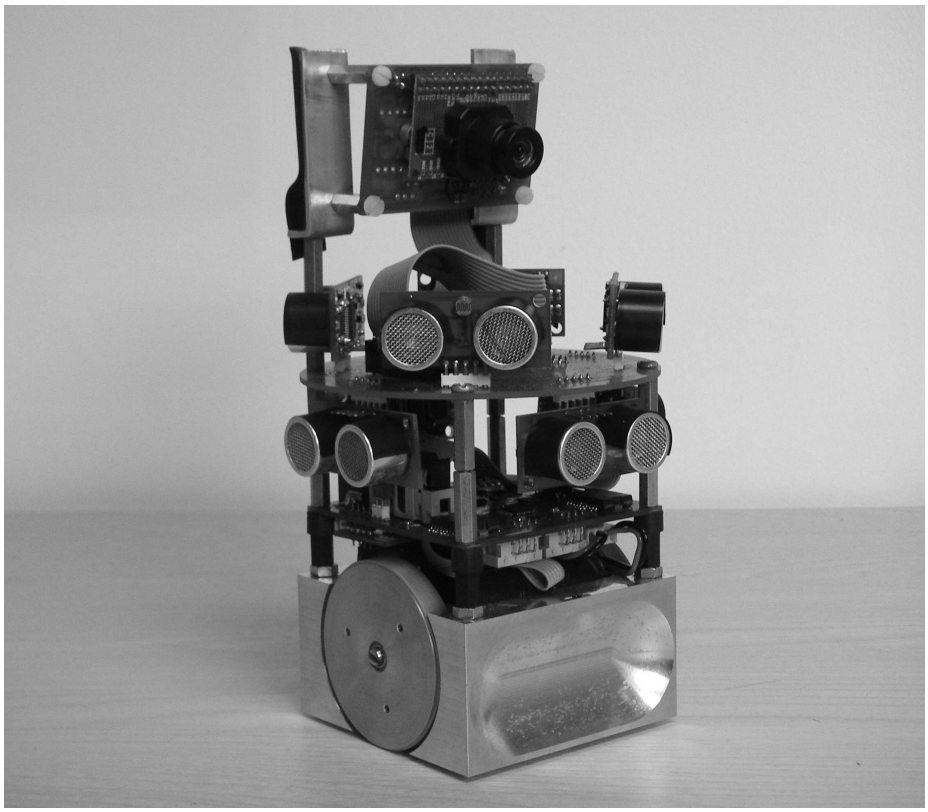


Figure 6.1: Merlin Miabot Pro with ultra-sonic sensor array and camera modules.

- *Sonar*: an ultra-sonic sensor array with a range of approximately $3cm - 2m$ and a field-of-view of approximately 360^o. As with the simulated Miabot, small gaps of approximately 15^o existed between each individual sensor.

- *Blob-finder*: a camera with a fixed forward orientation and 30^o field-of-view. An on-board blob-finder algorithm tracked the environment for a single blob defined by an RGB value sampled from a single frame grabbed from a camera approximately $30cm$ away from the target. Unlike the simulated blob-finder, the real blob-finder has no physical range limit as such. However, the further an object is from the camera the more out of focus it becomes, making it harder to differentiate specific blobs from the background.

In order to control the Miabots using the Player architecture, a Player plug-in driver was developed for the Miabot (see section 3.3.4 for more details).

In order to position the Miabots accurately within the environment needed by the sharing systems and the hybrid system (discussed in detail in chapter 7), an overhead camera system was used to track the Miabots (more information is provided in section 3.2.4). Again, a Player plug-in driver was developed to add the tracking system to the Player environment. See section 3.3.5 for more information.

6.3 Environment Specification

As our overhead tracking system could not cover the same area as was used in the simulated experiments, the experimental environments had to be redesigned. It was decided to test the systems on three different environments differing in the number of obstacles present, from a sparse environment (20% obstacle coverage) to a cluttered environment (40% obstacle coverage); see figures 6.2, 6.3 and 6.4. The positions of obstacles, the target, and the Miabot's starting location in the cluttered environment were generated randomly. To create the sparse environments four obstacles were removed randomly (one from each image quarter) to create environment 2, and another four to make environment 3. Each environment was approximately $1.7m \times 2.5m$, as this was the maximum area that the overhead camera system could cover, and was enclosed within $15cm$ high walls. The reason why the environment sizes in the simulation experiments in sections 5.4 and 5.5 do not match the sizes used in these laboratory experiments, is that when the simulation experiments were conducted it was beleived that it would be possible run the laboratory experiments under the same conditions, unfortunately this did not turn out to be the case. The target was a pink can approximately $12cm \times 7cm \times 7cm$. Obstacles were rectangular boxes approximately $16cm \times 12cm \times 8cm$.

6.3.1 Environment Noise

Unlike in simulation, in which the Miabots were given perfect (noiseless) information, the environment in the lab had numerous sources of noise. A brief description of the major contributors were:

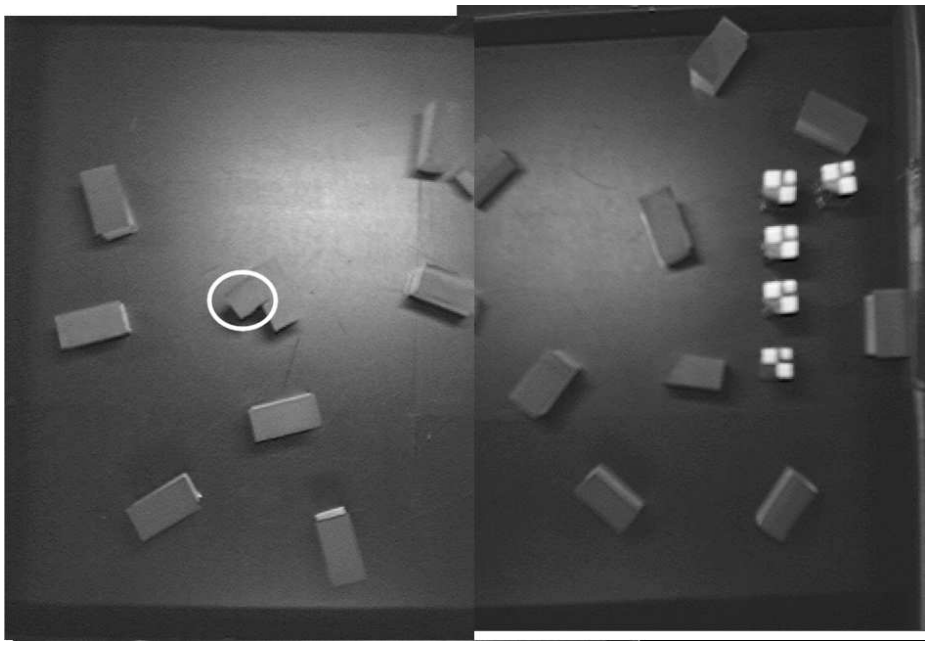


Figure 6.2: Overhead view of the arena (two camera frames): Environment 1 — cluttered (with 5 Miabots at their initial starting locations). The target is circled.

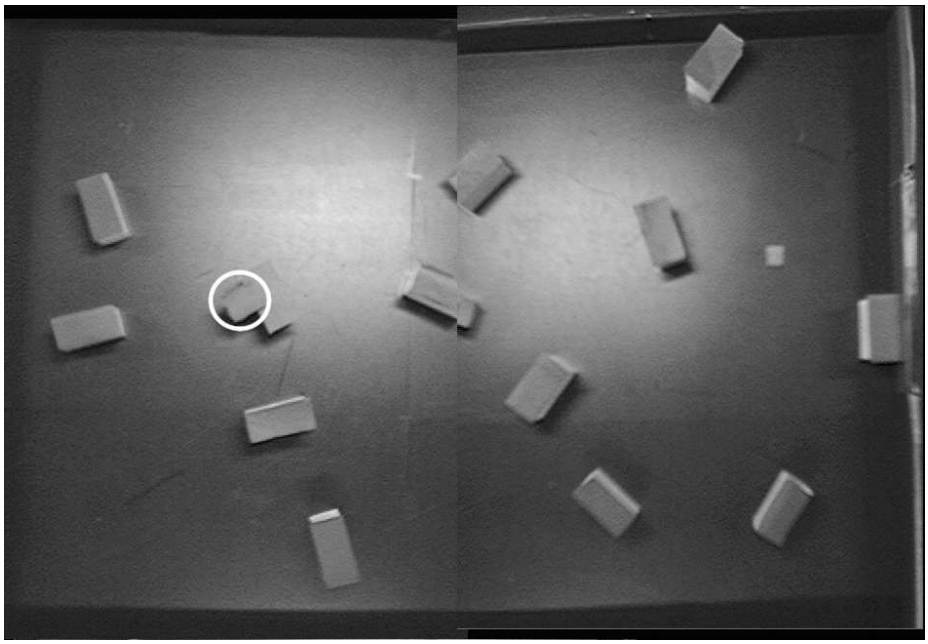


Figure 6.3: Overhead view of arena (two camera frames): Environment 2 — normal. The target is circled.

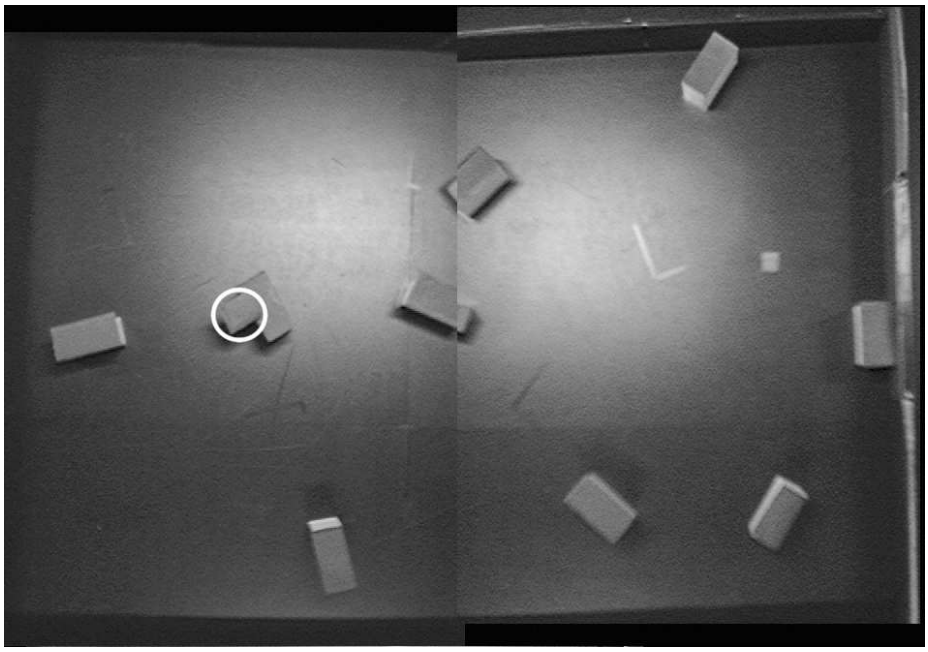


Figure 6.4: Overhead view of arena (two camera frames): Environment 3 — sparse. The target is circled.

- *Ultra-sonic sensors*: Each Miabot used in the experimentation was equipped with an ultra-sonic sensor array as described earlier. The most fundamental thing to remember when using ultra-sonic sensors is that the retrieval of echoes is not guaranteed due to numerous environmental factors that could lead to the source single being absorbed, or the echo dissipating before reaching the transceiver on the ultra-sonic sensor.

  One of the drawbacks of manufacturing ultra-sonic sensors to the required size to be mounted on a Miabot is that the frequencies used by the sensors are set by the manufacturer and cannot be altered. As all of the ultra-sonic sensors are set to the same frequency, if two ultra-sonic sound waves from different Miabots collide then the result is either false object detection at the point of collision, or no echo is received by the Miabots resulting in no object detection. Both of these situations can be dangerous as either the Miabots avoid objects that are not there, or do not avoid objects that are there. An example is shown in figure 6.5.

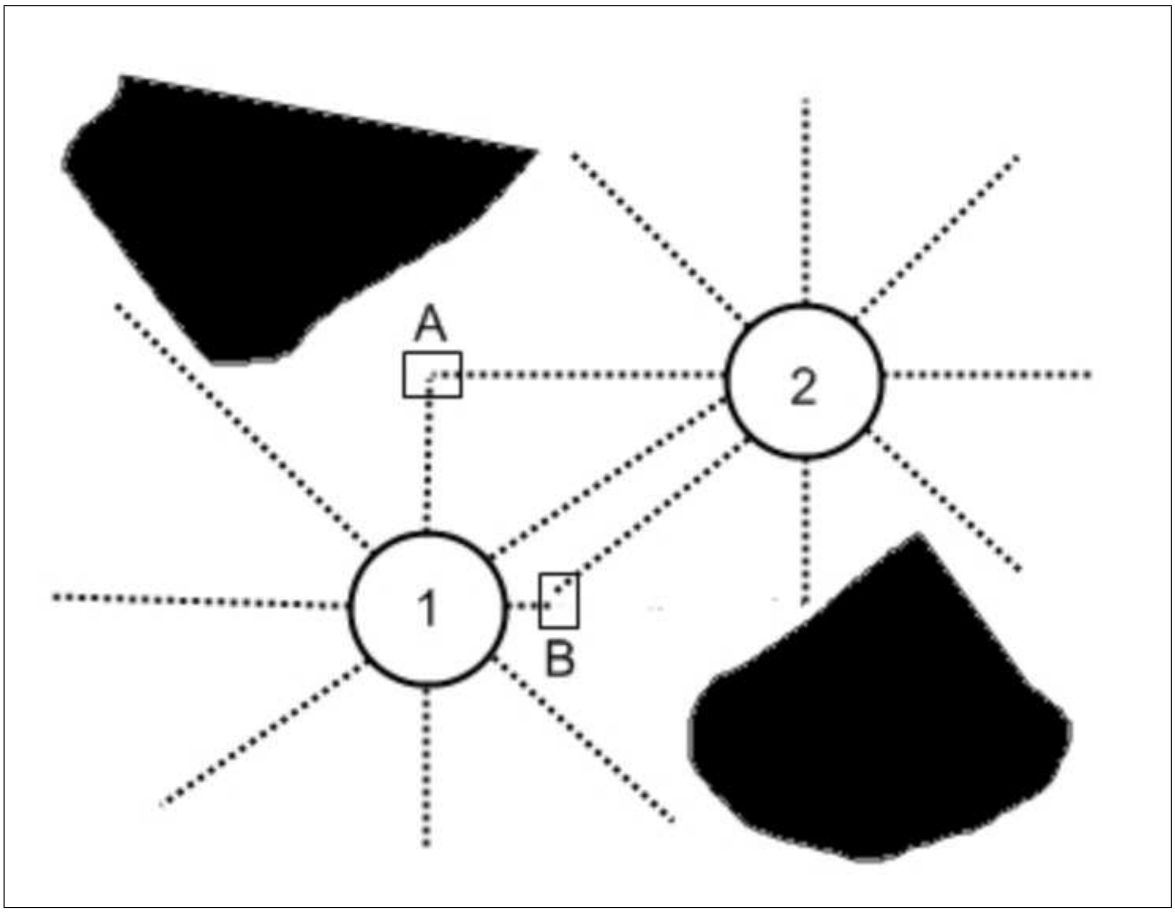


Figure 6.5: Example of ultra-sonic collisions. In case A, the collision stops both robots from detecting the object in the top left corner. In case B, the collision stops robot 1 from detecting the object in the bottom right corner.

As the minimum range of the ultra-sonic sensors is $3cm$, any object that gets closer to a Miabot than that may cause a collision. Miabots can collide with three different types of objects within the environment; obstacles, the target or other Miabots. If a Miabot collides with an obstacle, the Miabot can move the obstacle out of the way, and resume the search after getting clear of the obstacle (assuming the Miabot has sufficient momentum to move the obstacle). If this happens during an experimental run using the wavefront propagation path planner, the result could be a less efficient path being plotted to a goal, due to planning paths through unmapped obstacles (more information on the

wavefront algorithm is given in section 7.2.1). Otherwise, the Miabot may get stuck against the obstacle (in the case of the potential field systems, as they do not implement a crash recovery mechanism) or they will take an unknown amount of time to recover from the crash.

If a Miabot collides with the target, it will push the target until it becomes unattached (as the target's weight is much lower than the Miabot's). This could either be helpful, by moving the target into a more accessible area of the arena, or un-helpful, by moving the target to a less accessible part of the arena. If a Miabot collides with another Miabot, then either it will be a glancing blow and the Miabots will carry on as normal or, as in the obstacle case, the Miabot will be stuck for an unknown amount of time, perhaps indefinitely.

- *Blob-finder*: Unlike in simulation, in which all objects within the environment had a single uniform colour, in the real world objects are made up of many colours. Even objects that are one colour to the naked eye are in fact many variations of the colour at the blob-finder level. As such, the blob-finder is susceptible to both false positives and false negatives. An example of the type of blob data received is given in figure 6.6, where multiple blobs are tracked from the same target (Labelled A-E in the figure).

  False positives occur when the blob-finder detects the correct RGB value for that target, but in fact is not "looking" at the target. To overcome this a minimum area is defined, all blobs that have an area smaller than this are disqualified as possible targets (in the example only the large blob would be classed as a target). However, when a Miabot does have a false positive, the experiment must be restarted, as the reliability of the blob-finder is not being tested.

  False negatives occur when the blob-finder fails to detect the target even though it is "looking" at it. This happens if the Miabot gets too close to the target and so the target area is too small to be recognised, or due to lighting conditions such as shadows cast by other Miabots

- *Tracking system*: The tracking system used to track the Miabots across two

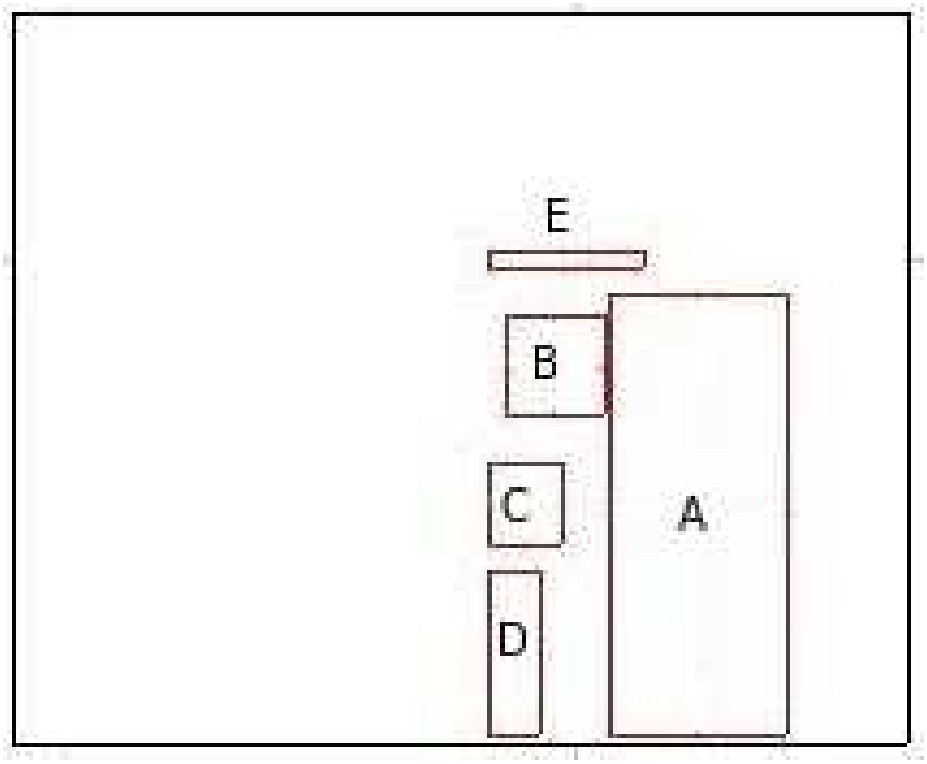


Figure 6.6: Example of blob data: The Miabot is approximately $15cm$ from the target. Multiple blobs are found (Labelled A-E). The rectangles represent areas of the image with the same RGB value.

separate camera frames is heavily affected by lighting conditions. To help create stable conditions, the windows to the laboratory were blacked out in order to stop natural light entering. However, due to the laboratory configuration, the arena in which the Miabots perform experiments is directly below a light fitting. This can be see in figures 6.2, 6.3 and 6.4 with the bright patch of light. Positioning the Miabots whilst they are in this area is less accurate than when the Miabots are in other parts of the arena. The inaccuracy is only really apparent when the Miabots are moving at a velocity under $10cm/s$, and it usually results in an incorrect orientation calculation.

The area of the arena covered by both of the camera frames is also an area where the calculated orientation can be inaccurate. The reason for this is that when a Miabot moves in between the two frames, the smallest tracking blob

can be lost, leading to a false orientation being reported.

The tracking system works by finding three blobs (two large, one small) that are within close proximity to each other. However, if two Miabots collide with each other, the blobs from two different Miabots can be used to calculate the position of one Miabot, hence creating a false position. Collisions can also lead to Miabots swapping positioning IDs, if the tracking system mistakes one Miabot for another. This will result in the Miabots carrying out commands which have no relation to their current position. More can be read about the tracking system in section 3.2.4.

- *Communications Lag*: The Miabots communicate to the Player server through a virtual serial communications link, via Bluetooth. The time between a Miabot completing an action and sending information back to the Player server can be up to $40ms$, this is due to numerous verification processes Bluetooth conducts to provide reliable communications. Unlike traditional radio communications.

  The Miabots single communications bus leads to the fact that the more devices connected to the Miabot, the higher the lag between send and receive commands. In experimentation, only three modules are used: the motor drive; the ultra-sonic sensor array; and the blob-finder. As the motor drive only receives commands, its impact on communications lag is insignificant. However, the ultra-sonic sensor array and the blob-finder poll the bus continuously throughout the experiment. As the blob-finder is the master of the bus, this leads to the ultra-sonic data not being transmitted to the Player server until the blob-finder is finished with the bus. This can obviously lead to the Miabot not only processing commands slowly but also to putting it at risk of colliding with obstacles in areas with a high blob count.

- *Slippage*: Unlike in the simulation experiments in which input and output of the motors are related by a linear function, on the real Miabots a number of factors cause unexpected motor behaviour. These include wheel slippage; weight distribution; efficiency of motors; voltage level; and dominance of individual

motors. However, the weight distribution and voltage levels of individuals can be controlled within acceptable limits during experimentation. The efficiency and dominance of individual motors and the effect of wheel slippage are factors that can only be tackled by providing each Miabot with its own individual low level motor controller, which is a research area not covered by this thesis.

6.4 Single Target Search

During this experiment [10], the motion of the Miabots was limited to either move forwards or rotate. They moved forward at approximately $0.1m/s$, rotated at approximately $0.87rad/s$ (with a forward motion of approximately $0.005m/s$, to help avoid the oscillation problems discussed in section 2.6.1). The on-board blob-finder algorithm in the camera was used to detect the colour of the target. Once the target was verified by the blob-finder, the Miabot(s) came to a permanent halt.

Table 6.1 shows the average time taken, in seconds, for each system to complete the search task with a given group size. The best result for each group size is shown in **bold**. Full results are given in appendix A.4.

It can be seen that, in the cluttered environment (environment 1), the pessimistic system performed best in group sizes of 4 or more. In environment 2, the pessimistic system performed best in all group sizes in all but two cases (4 and 8 Miabots). In the sparse environment (environment 3), the optimistic system performed best in groups of 4 to 6 Miabots.

6.4.1 Comparison Across Systems

The Kruskal-Wallis test, previously described in section (5.3.1), was used to compare the performance of each system against the other systems over the different environments. Table 6.2 shows the differences between the means of ranks for environments 1 and 2. Significant differences are in **bold**. It can be seen from table 6.1, despite some clear trends, that in environment 1, only two significant cases occur. The pessimistic system performed better than the non-sharing system with five Miabots and the pessimistic system performed better than the optimistic system with four Miabots. This

Table 6.1: Mean completion (seconds) for each system in each environment for 2-8 Miabots, to 1 d.p. The standard deviation is given in brackets, to 1 d.p.

	2	3	4	5
Environment 1				
non	**266.0** (61.3)	**220.6** (84.6)	216.3 (68.9)	232.8 (82.2)
pes	273.7 (57.5)	228.9 (85.3)	**177.9** (61.2)	**174.6** (73.5))
opt	291.0 (26.1)	243.2 (73.0)	221.8 (68.5)	197.1 (69.9)
Environment 2				
non	219.4 (105.6)	184.4 (102.7)	160.1 (100.7)	219.4 (73.2)
pes	**182.5** (100.1)	**161.8** (82.9)	140.2 (74.1)	**115.4** (60.3)
opt	216.2 (112.5)	212.1 (87.3)	**134.5** (103.9)	183.9 (96.8))
Environment 3				
non	**189.4** (109.4)	**161.6** (117.2)	143.9 (106.1)	154.9 (100.0)
pes	216.8 (113.3)	177.4 (105.2)	171.7 (104.3)	143.9 (105.2)
opt	207.0 (106.1)	163.1 (113.5)	**117.0** (90.0)	**97.3** (72.7)

	6	7	8
Environment 1			
non	219.3 (85.5)	236.1 (74.4)	172.4 (69.0)
pes	**187.3** (76.4)	**212.9** (75.9)	**167.2** (81.0)
opt	194.8 (64.7)	225.3 (79.0)	169.2 (77.7)
Environment 2			
non	180.9 (90.7)	**184.5** (97.0)	167.4 (90.6)
pes	**164.4** (83.5)	189.0 (84.5)	**153.7** (88.9)
opt	167.4 (93.4)	187.8 (85.9)	178.2 (85.5)
Environment 3			
non	111.8 (88.9)	**84.9** (66.1)	102.1 (72.9)
pes	107.1 (87.9)	102.2 (68.6)	**100.1** (82.0)
opt	**65.0** (39.9)	119.7 (84.0)	101.2 (90.0)

can be seen clearly in figure 6.7, in which, points between the horizontal lines relate to there being no significant difference between the performances of all the systems. Again, in environment 2, only two significant cases occur, with the pessimistic system outperforming the non-sharing and optimistic systems with five Miabots. This can be seen clearly in figure 6.8. Note that no significant cases occurred in environment 3.

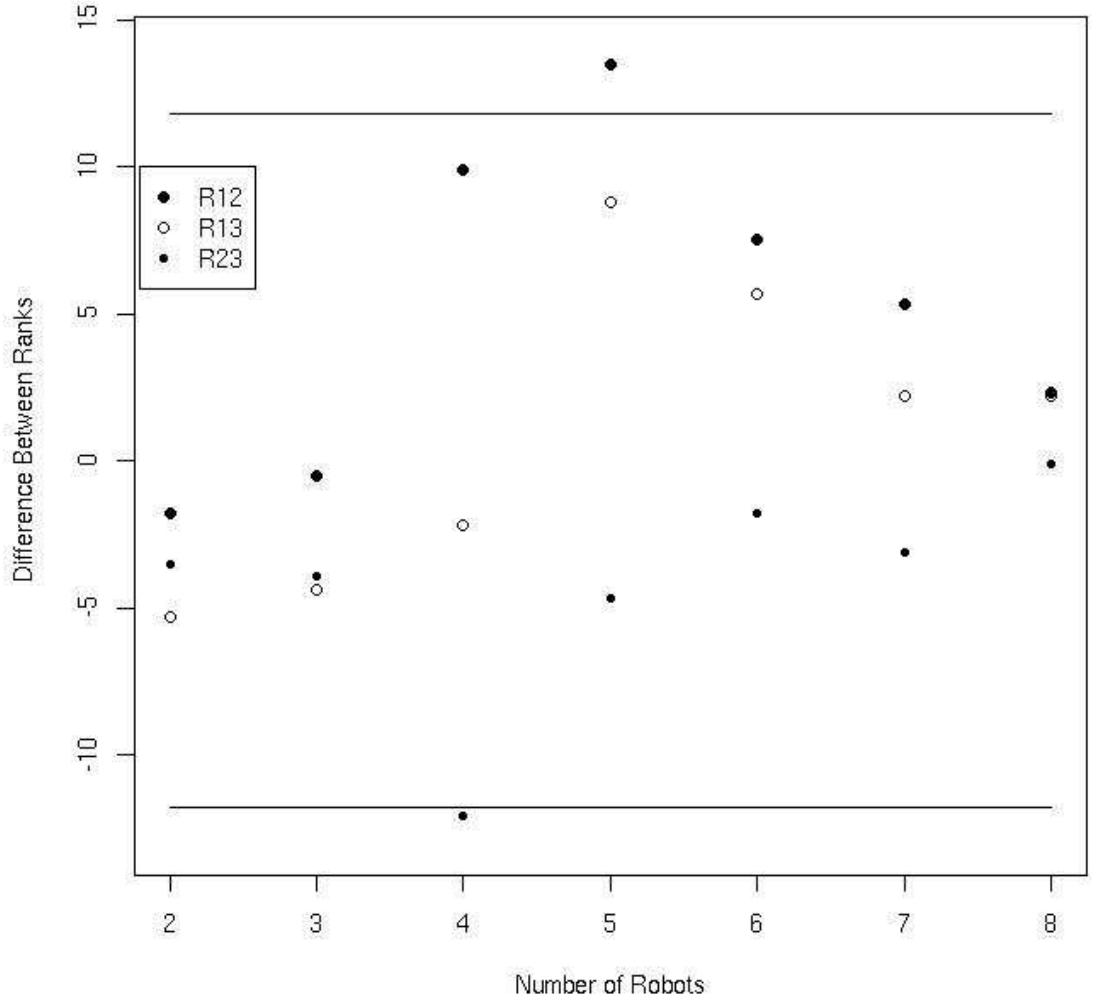


Figure 6.7: Differences between ranks for environment 1: R12 — differences between non-sharing and pessimistic systems. R13 — differences between non-sharing and optimistic systems. R23 — differences between pessimistic and optimistic systems. Points in between the two horizontal lines are not significant. Points in between the two horizontal lines are not significant.

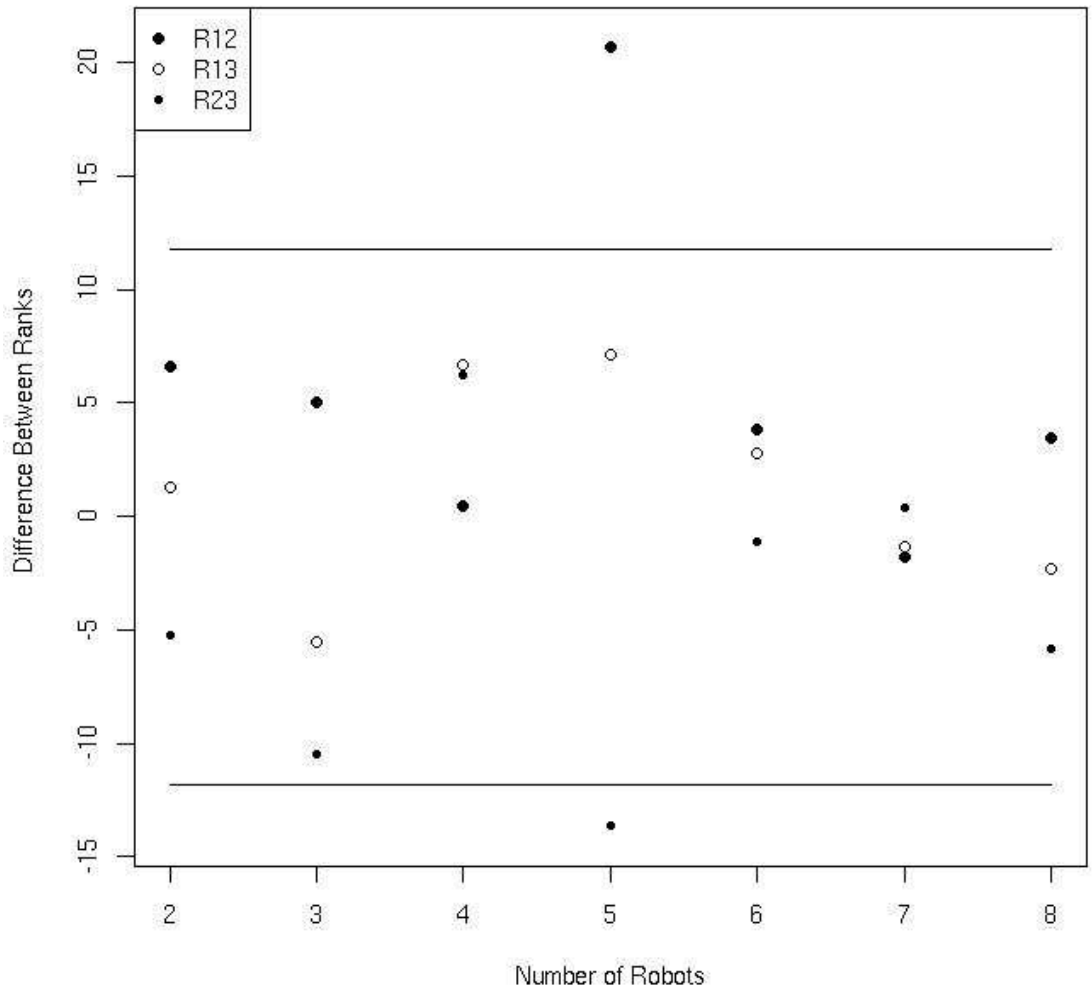


Figure 6.8: Differences between ranks for environment 2: R12 — differences between non-sharing and pessimistic systems. R13 — differences between non-sharing and optimistic systems. R23 — differences between pessimistic and optimistic systems. Points in between the two horizontal lines are not significant. Points in between the two horizontal lines are not significant.

Table 6.2: Significant differences between the non-sharing ($R1$), pessimistic ($R2$) and optimistic ($R3$), systems (to 1 d.p.)

	2	3	4	5	6	7	8
Environment 1							
$R12$	-1.8	-0.5	9.9	**13.5**	7.5	5.3	2.3
$R13$	-5.3	-4.4	-2.2	8.8	5.7	2.2	2.2
$R23$	-3.5	-3.9	**-12.1**	-4.7	-1.8	-3.1	-0.1
Environment 2							
$R12$	6.55	5.0	0.5	**20.7**	3.8	-1.8	3.5
$R13$	1.3	-5.5	6.7	7.1	2.8	-1.3	-2.3
$R23$	-5.2	-10.5	6.2	**-13.6**	-1.1	0.4	-5.8
Environment 3							
$R12$	-3.9	-4.0	-4.6	4.0	-0.3	-5.3	1.6
$R13$	-2.3	-0.6	4.3	10.7	7.7	-7.9	2.0
$R23$	1.6	3.4	8.8	6.7	7.9	-2.7	0.4

6.4.2 Comparison Across Size

The Friedman test was used to compare the performance of each system with differing group sizes. See section 5.3.2 for more information on the Friedman test.

Tables 6.3, 6.4 and 6.5 show the differences between the column sums for different group sizes for each system. Significant differences are in **bold**. It can be seen again, that despite clear trends from table 6.1, only one case occurs for the non-sharing system of any statistical significance. This is where groups of 8 Miabots out perform groups of two Miabots. This can be seen in figure 6.9. For the pessimistic system, again the number of significant cases is low with groups of four, five, six and eight Miabots out performing groups of two Miabots. This can be seen in figure 6.10. The optimistic system has similar results with groups of five, six and eight Miabots out performing groups of two Miabots. This can be seen in figure 6.11. Note that only in environment 1 did the size of groups have any effect on a systems performance.

Table 6.3: Significant differences between the number of Miabots, for the non-sharing system in environment 1, to 1 d.p.

	2	3	4	5	6	7	8
2 Robots	NA	25.5	27.0	22.0	24.0	16.0	**43.0**
3 Robots	-25.5	NA	1.5	-3.5	-1.5	-9.5	17.5
4 Robots	-27.0	-1.5	NA	-5.0	-3.0	-11.0	16.0
5 Robots	-22.0	3.5	5.0	NA	2.0	-6.0	21.0
6 Robots	-24.0	1.5	3.0	-2.0	NA	-8.0	19.0
7 Robots	-16.0	9.5	11.0	6.0	8.0	NA	27.0
8 Robots	**-43.0**	-17.5	-16.0	-21.0	-19.0	-27.0	NA

Table 6.4: Significant differences between the number of Miabots, for the pessimistic system in environment 1, to 1 d.p

	2	3	4	5	6	7	8
2 Robots	NA	25.0	**51.0**	**58.5**	**43.5**	35.0	**56.5**
3 Robots	-25.0	NA	26.0	33.5	18.5	10.0	31.5
4 Robots	**-51.0**	-26.0	NA	7.5	-7.5	-16.0	5.5
5 Robots	**-58.5**	-33.5	-7.5	NA	-15.0	-23.5	-2.0
6 Robots	**-43.5**	-18.5	7.5	15.0	NA	-8.5	13.0
7 Robots	-35.0	-10.0	16.0	23.5	8.5	NA	21.5
8 Robots	**-56.5**	-31.5	-5.5	2.0	-13.0	-21.5	NA

Table 6.5: Significant differences between the number of Miabots, for the optimistic system in environment 1, to 1 d.p.

	2	3	4	5	6	7	8
2 Robots	NA	26.5	34.0	**58.0**	**55.5**	35.5	**63.5**
3 Robots	-26.5	NA	7.5	31.5	29.0	9.0	37.0
4 Robots	-34.0	-7.5	NA	24.0	21.5	1.5	29.5
5 Robots	**-58.0**	-31.5	-24.0	NA	-2.5	-22.5	5.5
6 Robots	**-55.5**	-29.0	-21.5	2.5	NA	-20.0	8.0
7 Robots	-35.5	-9.0	-1.5	22.5	20.0	NA	28.0
8 Robots	**-63.5**	-37.0	-29.5	-5.5	-8.0	-28.0	NA

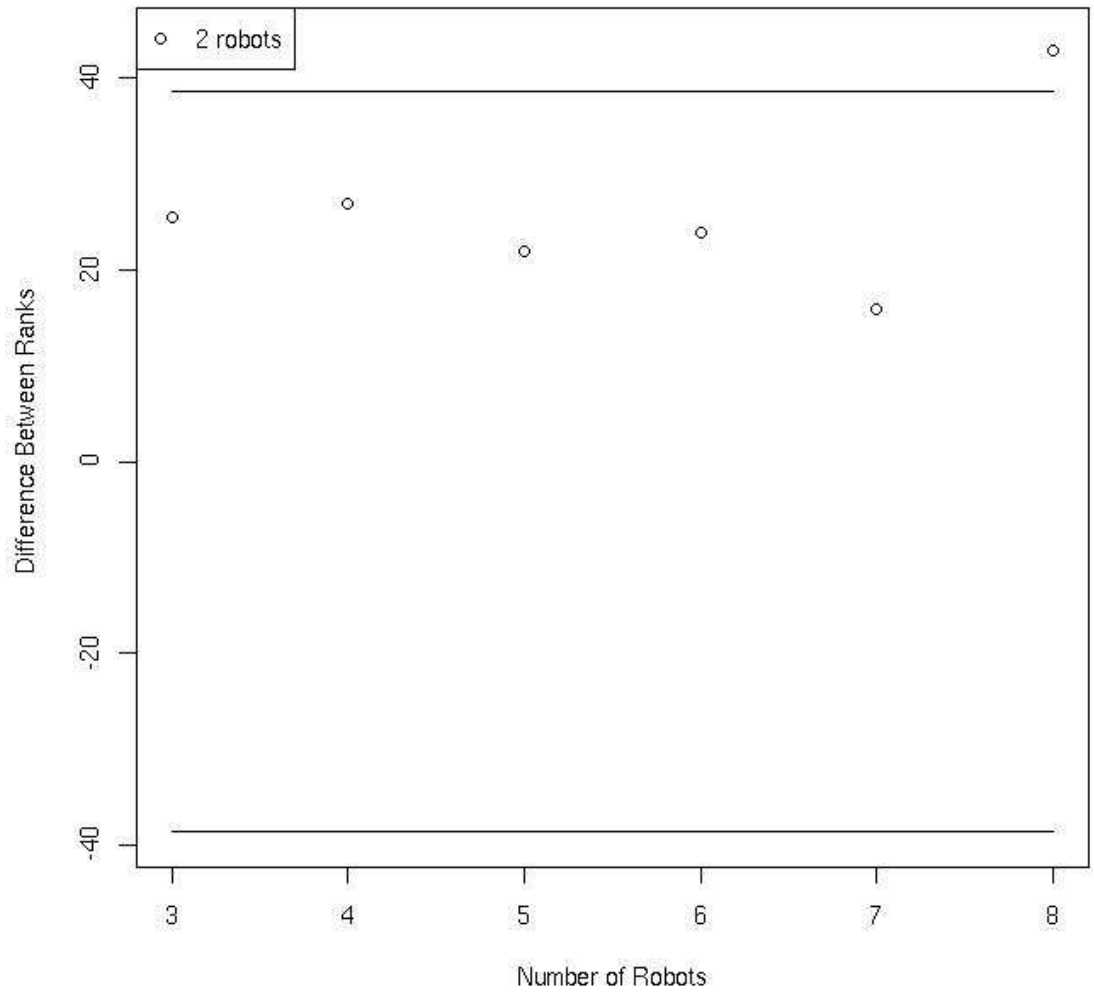


Figure 6.9: Differences between ranks for the non-sharing system: Differences between performance of 2 robots compared to 3 to 8 robots. Points in between the two horizontal lines are not significant.

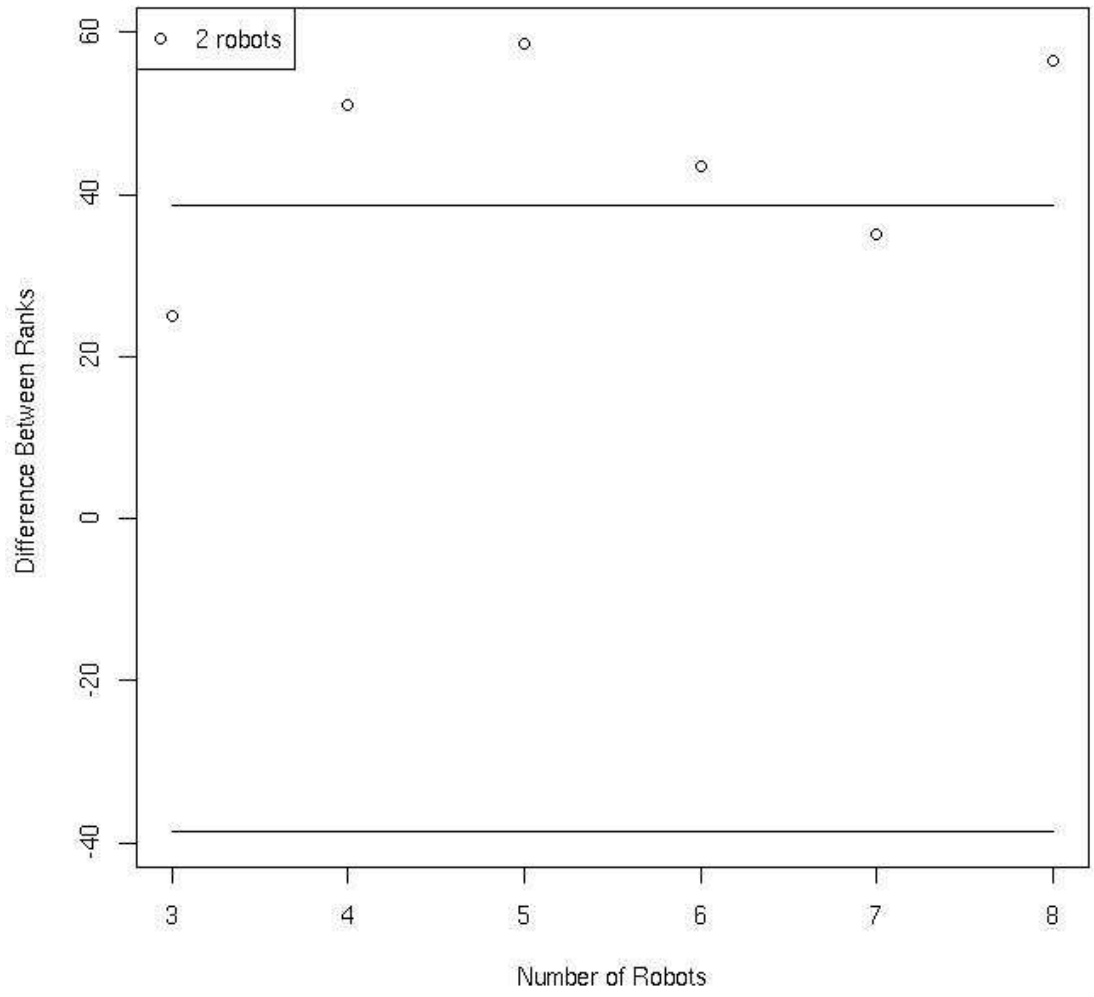


Figure 6.10: Differences between ranks for the pessimistic system: Differences between performance of 2 robots compared to 3 to 8 robots. Points in between the two horizontal lines are not significant.

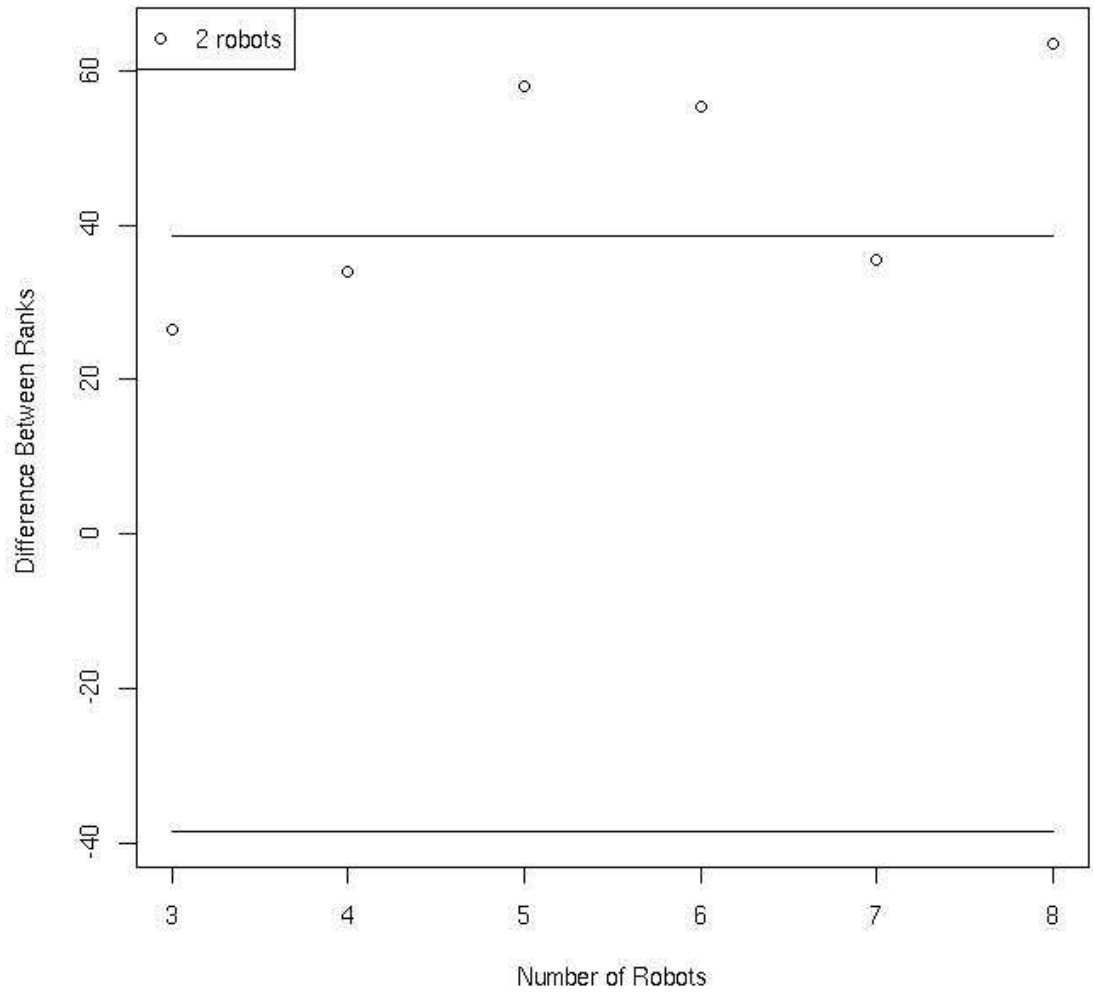


Figure 6.11: Differences between ranks for the optimistic system: Differences between performance of 2 robots compared to 3 to 8 robots. Points in between the two horizontal lines are not significant.

6.4.3 Comparison Across Environments

The Kruskal-Wallis test was employed to compare system performance across the three different environments (see section 5.3.1 for more information on the Kruskal-Wallis test).

Tables 6.6, 6.7 and 6.8 show the differences between the means of ranks for each system across the three different environments. Significant differences are in **bold**. It can be seen that the non-sharing systems performance increased as the environment became more sparse. The non-sharing system performing better in environment 3 than in environment 1 in every case but one (3 Miabots), and better in environment 3 than in environment 2 with six or more Miabots. This can be seen in figure 6.12. The pessimistic system performance worsened the more sparse the environment became (when a small number of Miabots were deployed). The pessimistic system performed better in environment 2 than in environment 1 with up to five Miabots, and better in environment 3 than in both environment 1 and 2 in cases with six or more Miabots. This can be seen in figure 6.13. The optimistic system performance increased, the more sparse the environment became. The optimistic system performed better in environment 3 than in environment 1 in all cases and it performed better in environment 3 than in environment 2 with five or more Miabots. This can be seen in figure 6.14.

Table 6.6: Significant differences between means, non-sharing system, to 1 d.p.

	2	3	4	5	6	7	8
$R12$	6.18	5.4	11.0	3.1	6.9	8.2	2.2
$R13$	**11.9**	10.7	**13.6**	**13.7**	**19.7**	**27.4**	**17.1**
$R23$	5.7	5.4	2.7	10.6	**12.8**	**19.2**	**14.9**

Table 6.7: Significant differences between means, pessimistic system, to 1 d.p.

	2	3	4	5	6	7	8
$R12$	**15.1**	**12.0**	9.6	**12.0**	5.6	4.9	3.2
$R13$	8.8	8.5	2.3	9.2	**18.8**	**22.2**	**16.0**
$R23$	-6.3	-3.5	-7.3	-2.8	**13.2**	**17.2**	**12.8**

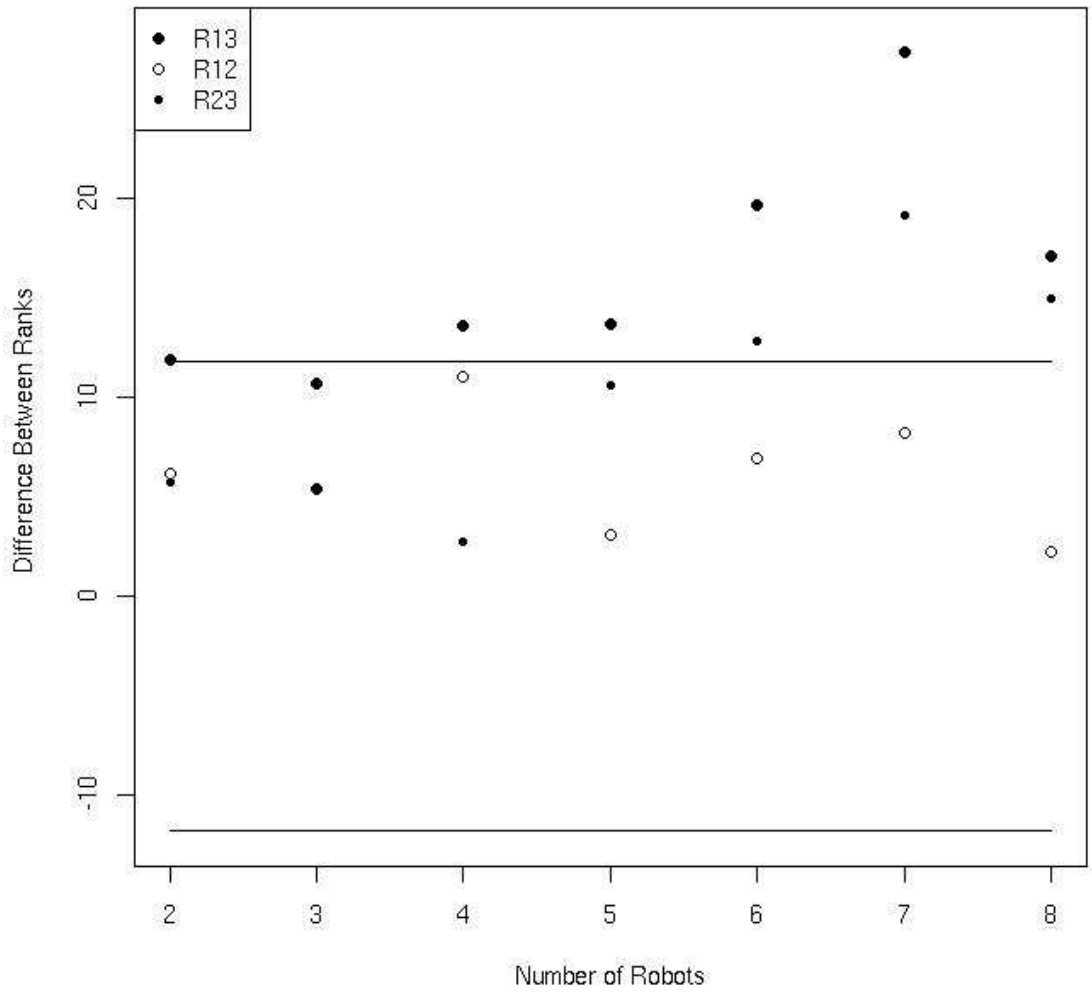


Figure 6.12: Differences between ranks for the non-sharing system: R12 — differences between Environment 1 and Environment 2. R13 — differences between Environment 1 and Environment 3. R23 — differences between Environment 2 and Environment 3. Points in between the two horizontal lines are not significant.

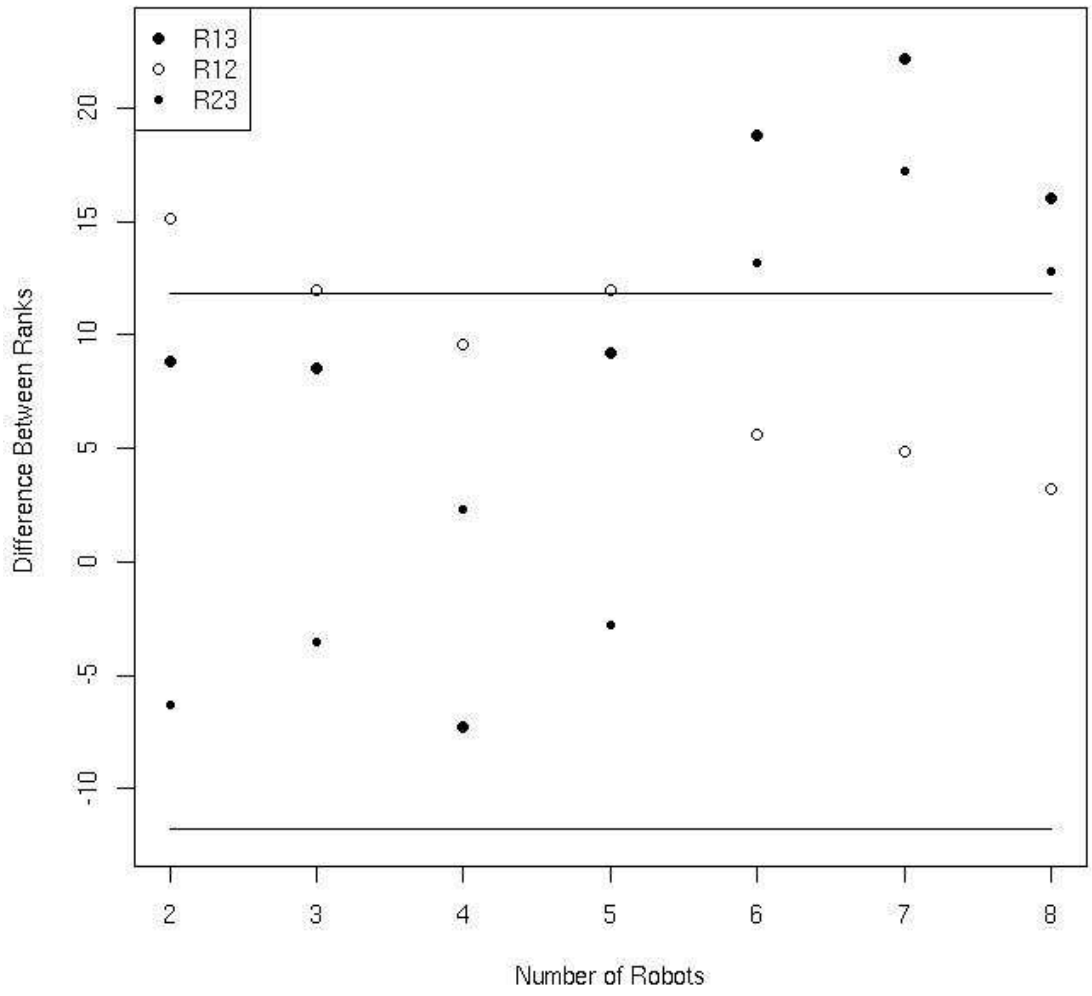


Figure 6.13: Differences between ranks for the pessimistic system: R12 — differences between Environment 1 and Environment 2. R13 — differences between Environment 1 and Environment 3. R23 — differences between Environment 2 and Environment 3. Points in between the two horizontal lines are not significant.

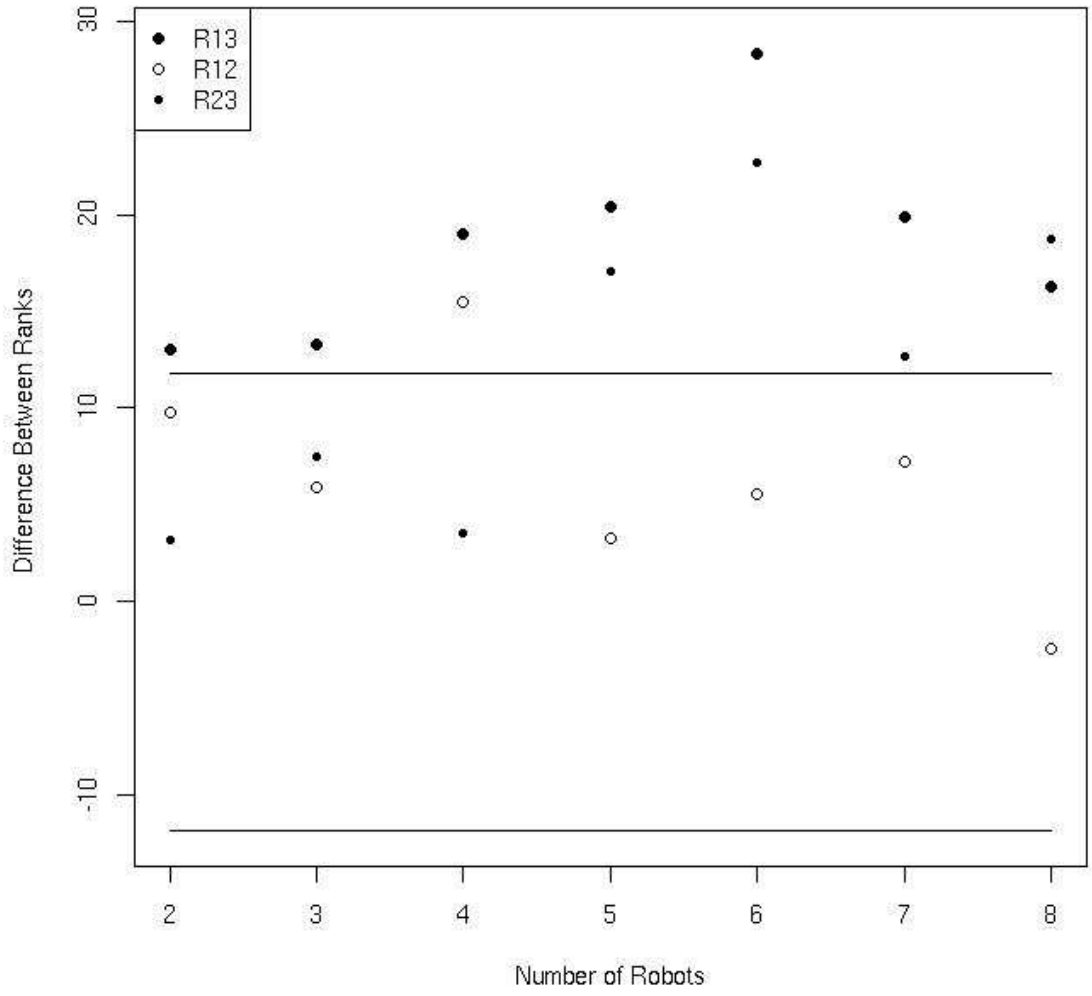


Figure 6.14: Differences between ranks for the optimistic system: R12 — differences between Environment 1 and Environment 2. R13 — differences between Environment 1 and Environment 3. R23 — differences between Environment 2 and Environment 3. Points in between the two horizontal lines are not significant.

Table 6.8: Significant differences between means, optimistic system, to 1 d.p.

	2	3	4	5	6	7	8
$R12$	9.8	5.9	**15.5**	3.3	5.6	7.2	-2.4
$R13$	**13.0**	**13.4**	**19.0**	**20.4**	**28.3**	**19.9**	**16.3**
$R23$	3.2	7.5	3.5	**17.1**	**22.7**	**12.7**	**18.7**

6.4.4 Comparison to Simulation Results

Only the pessimistic system with five Miabots significantly outperformed the non-sharing system in the cluttered environments (environments 1 and 2). This is in contrast to our simulation results, shown in chapter 5, in which both the pessimistic and the optimistic system outperformed the non-sharing system with six or more Miabots. This is unlike in the simulation experiments where the results show that groups of 6 or more Miabots outperformed groups of 5 or less. In the laboratory experiments, only groups of 2 Miabots performed significantly worse than other group sizes. The differences in results were due to the inherent noise within the real Miabot sensors which was not taken into consideration in the simulations.

6.4.5 Discussion of Single-target Results

The results show that the pessimistic system performed better in the more sparse environment (environment 3) than in the cluttered environments (environments 1 and 2) with six or more Miabots. It is believed that this is due to the emergent behaviour of the pessimistic system to be more cautious and hence able to navigate through cluttered environments better than the less cautious optimistic system. However, the optimistic system performed better in the more sparse environment (environment 3) with 2 or more Miabots than in the cluttered environment (environment 1), and with five or more Miabots than in environment 2. It is believed that this is because the emergent behaviour of the system is less cautious and, hence, its navigation through sparse environments is more effective. Perhaps surprisingly, the size of the Miabot groups only had an effect in the cluttered environment (environment 1). This is probably due to the fact that a higher number of Miabots allowed the group to

saturate the environment. The limited size of the environment was probably an important factor. In a larger environment, it is hypothesised that the number of Miabots in the group would have a more significant impact on a system's performance. The differences between the simulation and laboratory results were due to the inherent noise within the real Miabot sensors and the laboratory environment.

6.5 Summary

In this chapter, the laboratory experimental setup has been described. The potential field sharing system was compared against a non-sharing control in three different environments (differentiated by object density), with group sizes of 2 to 8 Miabots.

It has been shown that in groups of five or more Miabots, the pessimistic variant of our potential field sharing system outperforms the non-sharing control. This is in contrast to our simulation findings where both variants outperformed the control.

The chapter also demonstrated the relationship between the density of objects within the environment and system performance. The pessimistic variant performed better in the cluttered environment than the optimistic variant, and the optimistic variant performed better in the sparse environment than the pessimistic variant. This is due to the pessimistic system being more cautious (in terms of belief in sensor data) than the optimistic system, which is of benefit in cluttered environments. The optimistic system is less cautious. This is a benefit in sparse environments.

Interestingly, group size had little effect on the performance of any of the systems. The non-sharing system performed better with 8 Miabots than with 2, the pessimistic system performed better with 4 or more Miabots than with 2, and the optimistic system performed better with 5 or more Miabots than with 2. The reason for the improvement in performance in the cluttered environment was due to the increased number of robots allowing the systems to increase area coverage. The sharing systems have a lower threshold for this improvement due to the benefits of sharing information.

The experiments, presented in this chapter, have helped this project meet two of its goals given in chapter 1.

- **To design and implement a multi-robot system that is not reliant**

upon explicit information gathered from other robots.

As with the simulation experiments, the potential field sharing method did not at anytime explicitly co-ordinate the team members, in the set of experiments discussed in this chapter. Co-ordination was an emergent property of combined potential fields.

The findings from the experiments described in this chapter has been submitted to the following journal.

- J.L. Baxter, E.K. Burke, J.M. Garibaldi, S. Groenemeyer & M. Norman, "Multi-robot Co-ordination Using Shared Potential Fields", **submitted to** *IEEE Transactions on Robotics*, 2009.

Chapter 7

Comparison Against a Hybrid System

7.1 Introduction

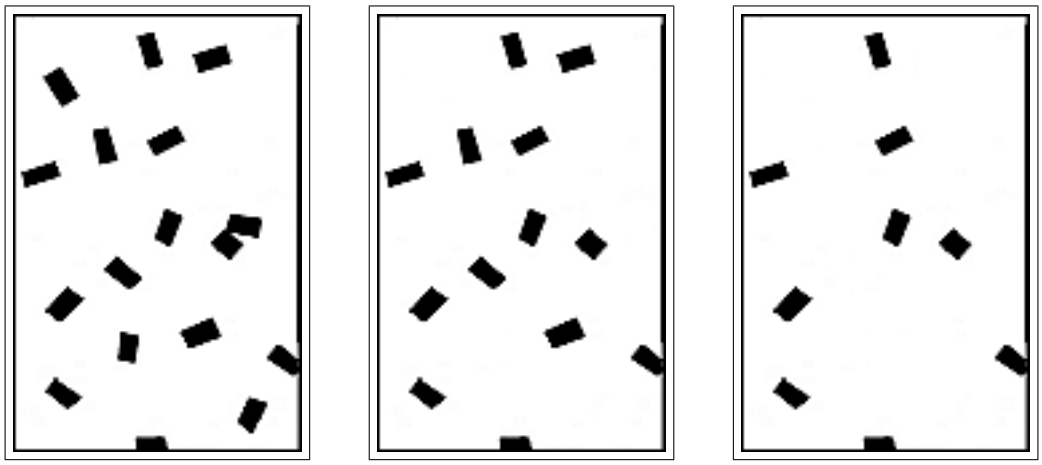
The next logical set of experiments was to compare a "known" robotic architecture against the potential field sharing method presented in this thesis.

The potential field sharing system gathers no information about the environment *a priori* and sensor inputs have a direct relationship to motor actions. As such, the system can be classified as a reactive system, as discussed in section 2.3.1. It was decided that it would be interesting to compare this reactive system against a non-reactive system, which left the choice of either a deliberative system or a hybrid system, both of which are discussed in detail in section 2.3. Given the highly dynamic nature of the environment within the experiments (multiple moving robots), it was decided that a purely deliberative system could not hope to compete with a reactive system in this scenario. Therefore, it was decided that it would be compared against a hybrid system, as knowledge gathered *a priori* could be used by a deliberative planner to create feasible solutions, whilst a reactive controller enables the system to adjust any plans due to unforeseen circumstances.

In this chapter, therefore, a hybrid robotic system attempts the single target search problem defined in chapter 6. Results are shown and compared against the results obtained by the potential field sharing system. The chapter ends with a summary.

7.2 The Hybrid System

The hybrid system was comprised of two modules. The deliberative module was the Wavefront propagation path planner, which when given a map (see figure 7.1) of the experimental environment calculated the shortest path to randomly generated targets. The reactive module, was the ND algorithm developed by Minguez *et al.*, as described in section 2.3.1, which enables the Miabot to avoid non-mapped obstacles. It is clearly visible by comparing the environments in figures 6.2-6.4 and the maps, that the target is not shown in the maps. This is to avoid the Wavefront algorithm from punishing the Miabot from being too close to the target (via the configuration space described in section 7.2.1).



(a) (b) (c)

Figure 7.1: Map files of the experimental environments provided to the Wavefront propagation path planner:(a) cluttered, (b) normal, (c) sparse. Note: Target not drawn on maps to avoid the Wavefront algorithm punishing the Miabot for going near the target.

Random targets were generated using a lagged Fibonacci (where $j = 273$ and $k = 607$) pseudo-random number generator over a uniform distribution in order to provide each robot with the "classic" random walk behaviour. The targets x and y positions and θ orientation were all generated separately, using the current system time as a seed for the pseudo-random number generator. This provided a spread of targets across the environments. An example distribution is shown in figure 7.2 which shows

that, after two hundred target generations, a high percentage of the environment has been covered. It should be noted that the target generation did not take into account obstacles within the environment. Once, given a map of the environment, targets generated within obstacles were ignored by the wavefront algorithm (as described in section 7.2.1).

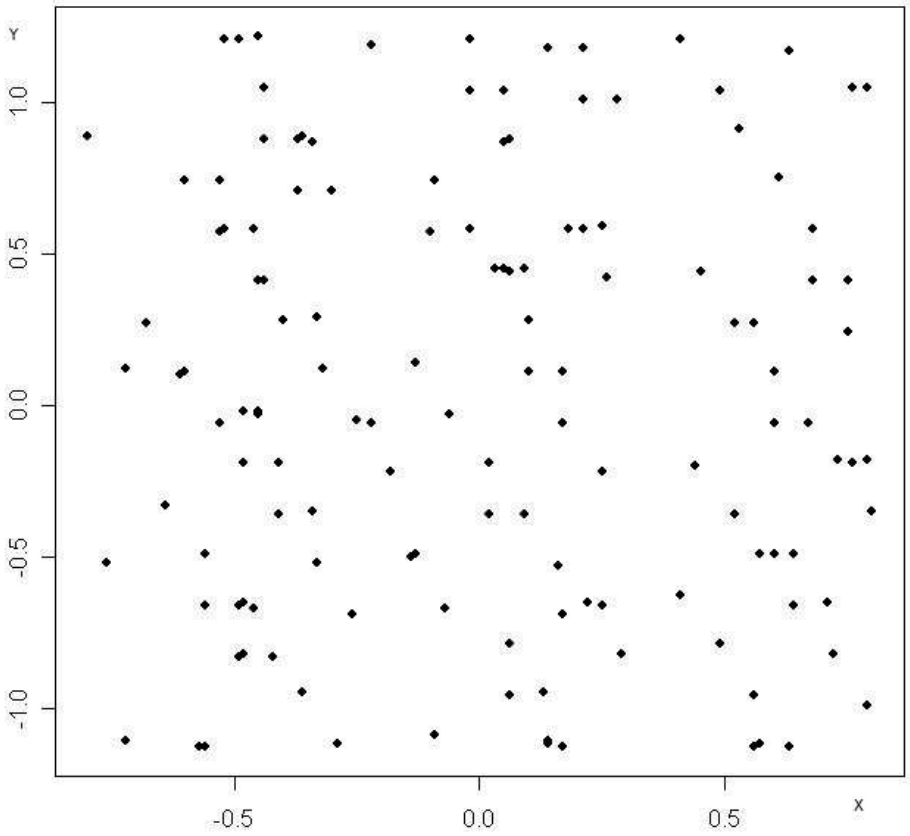


Figure 7.2: Example distribution of 200 points using the lagged Fibonacci pseudo-random number generator: The X and the Y axis represent the arena total area (measured in meters). In this case, points were plotted in pairs not closer than $0.2m$ to each other.

A well as differing in terms of being a hybrid system rather than reactive. The local level controller of the hybrid system in classed as unaware in Farinelli's mutli-robot taxonomy. Whereas, the system presented in chapter 4 has a local level controller which is classified as weakly co-ordinated. The global level controllers of the systems also differ, with, the hybrid system having a stongly co-ordinated, stongly centralised controller, and the system presented in this thesis having a global contorller which is

defined as unaware.

The reactive level of the hybrid system is an unaware multi-robot system. However, the deliberative level can be defined as a strongly co-ordinated, strongly centralised multi-robot system. As the current goals of all Miabots taking part in the experiment were globally available, and so the current goals of other Miabots were taken into account when generating new goals for Miabots (new goals were forbidden to be generated within a $20cm$ radius of current goals). This is shown in figure 7.3 with a five robot example. The circles are the Miabot's current locations, the large triangles are the Miabot's target locations, the small triangles are waypoints and, finally, the straight lines are the suggested paths (suggested by the wavefront algorithm). Note that one of the current locations is not joined up to a target location. This is because the Miabot has wandered too near an obstacle (represented by black rectangles) and so is in the process of re-planning.

One major advantage of using the Wavefront and ND algorithms was that drivers already existed for them within the Player architecture. Please see section 3.3.6 for more information on the Player implementations.

7.2.1 The Wavefront Propagation Algorithm

The algorithm works as follows:

1. Initially the algorithm checks whether or not the goal provided is valid, i.e. not within an obstacle. If the goal is invalid, a new goal is requested.

2. Given a map of the environment, a configuration space of grid cells is calculated (the dashed line in figure 7.4). Each cell is given a cost based upon its distance from any obstacles. To save computational time, the radius of the configuration around the Miabot is limited to $20cm$. All cells outside of the radius are given a cost value of zero. In the example, the obstacle cost is shown on the right hand side of the cells. The limit is represented by a dashed line.

3. Next, the wavefront is calculated. Each cell is given a value based upon its distance from the goal. If the cell has an obstacle value, this value is added

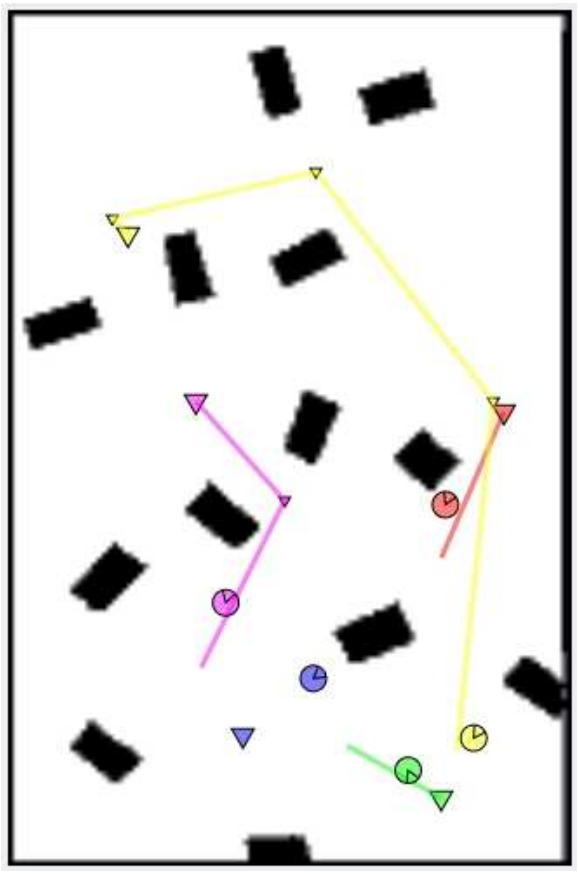


Figure 7.3: Example of a group of 5 Miabots planning paths within environment 2: Note that the target is not shown on the map.

to the value calculated by the wavefront. In the example, cells outside the configuration space only have one value. Cells inside have the plan cost on the left hand side and the total cost in the middle of the cell.

4. The algorithm then calculates a path by joining the current location cell and the goal cell via the lowest available adjacent cells. These cells are highlighted in the example.

5. If the Miabot gets stuck, a new path is calculated. Various properties can be set to help detect problems i.e. a distance threshold can be set to force the algorithm to re-plan if the Miabot gets too close to an obstacle. See section 3.3.6 for detailed information on the Wavefront configuration.

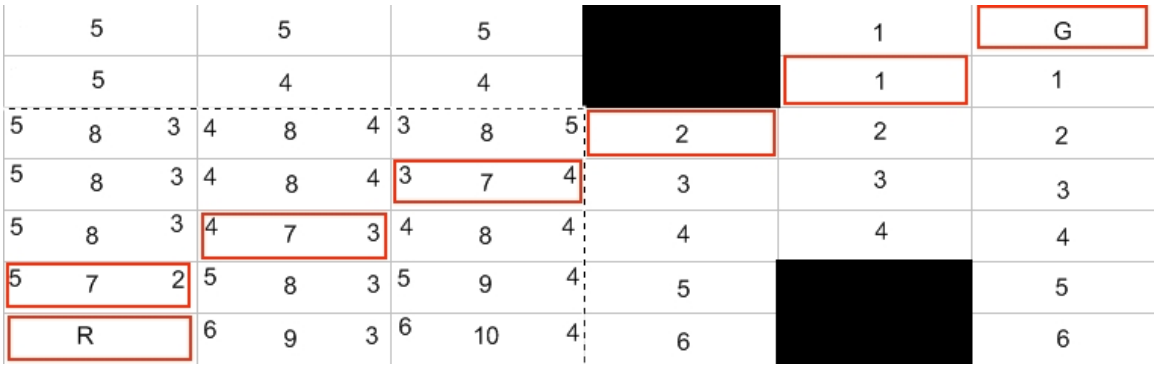


Figure 7.4: Example of the wavefront propagation algorithm: The two black rectangles are known obstacles. R is the starting location of the robot. G is the goal. The cells highlighted in red represent the path calculated by the algorithm.

7.3 Single Target Search

During the hybrid experiments [14, 15], the Miabots had a maximum velocity of approximately $0.1m/s$ or $10deg/s$, and a minimum velocity of $0.02m/s$ or $5deg/s$. Again, the on-board blob-finder algorithm in the camera was used to detect the colour of the target. The Miabot(s) came to a permanent halt once the target was found.

Table 7.1 shows the average time taken, in seconds, for the hybrid system to complete the search task with a given number of Miabots. It is clear that the best results for a given number of Miabots are all in the sparse environment (environment 3). Values shown in **bold** are the best results achieved compared to the three other systems (See table 6.1). Full results are given in appendix A.5.

7.3.1 Comparison Across Systems

As with the previous laboratory experiments, the Kruskal-Wallis test (see section 5.3.1) is used to compare the performance of the hybrid system against the three systems previously defined, over the different environments. However, as the number of samples has increased, the parameters used in the test need to be altered. Therefore, $k = 4$, $N = 80$ and $z = 2.4$ (to 1 d.p.). Table 7.2 shows the differences between the

Table 7.1: Mean completion (seconds) for the hybrid system in each environment for 2-8 Miabots, to 1 d.p. The standard deviation is given in brackets, to 1 d.p.

	2	3	4	5
Environment 1	**264.6** (67.1)	267.9 (60.0)	248.4 (73.6)	254.0 (70.6)
Environment 2	210.8 (83.0)	230.4 (95.9)	168.9 (72.5)	200.6 (78.8)
Environment 3	**156.7** (106.9)	**142.5** (84.4)	128.6 (70.2)	176.8 (98.1)
	6	7	8	
Environment 1	209.7 (73.6)	246.6 (60.0)	232.3 (72.8)	
Environment 2	219.5 (76.1)	**155.0** (83.8)	161.4 (81.0)	
Environment 3	126.5 (75.0)	115.7 (57.4)	117.9 (50.7)	

means of ranks for the three potential field systems and the hybrid system. Significant differences are in **bold**.

Table 7.2: Significant differences between the potential field systems (non-sharing (R_1), pessimistic (R_2) and optimistic (R_3)) and the hybrid (R_4) system (to 1 d.p.)

	2	3	4	5	6	7	8
Environment 1							
$R14$	0.2	-12.5	-10.9	-6.1	3.1	-2.9	-17.5
$R24$	2.5	-11.7	**-23.2**	**-23.5**	-6.8	-9.9	**-19.9**
$R34$	7.1	-6.8	-8.3	**-18.2**	-4.3	-5.7	**-19.5**
Environment 2							
$R14$	2.5	-9.7	-7.8	4.5	-10.8	6.0	0.6
$R24$	-6.7	-16.8	-9.6	**-24.3**	-15.8	9.1	-4.2
$R34$	1.1	-3.7	-16.8	-5.4	-14.2	8.4	3.9
Environment 3							
$R14$	5.2	-0.4	-0.5	-5.9	-10.6	-17.1	-11.8
$R24$	9.8	5.6	6.9	-11.0	-11.2	-8.5	-13.1
$R34$	8.5	1.2	-6.3	**-20.4**	**-22.1**	-4.4	-14.5

As only values of 17.6 (to 1 d.p.) or higher are significant, it can be seen that, in environment 1, the hybrid system performs as well as the non-sharing system, but worse than the pessimistic system with 5 or less and 8 Miabots. The optimistic system also performs better with 5 and 8 Miabots. This can be seen in figure 7.5 and table 7.2. In environment 2, the hybrid system performs as well as all of the potential field systems, in all but one case. This can be seen in figure 7.6 and table 7.2. In

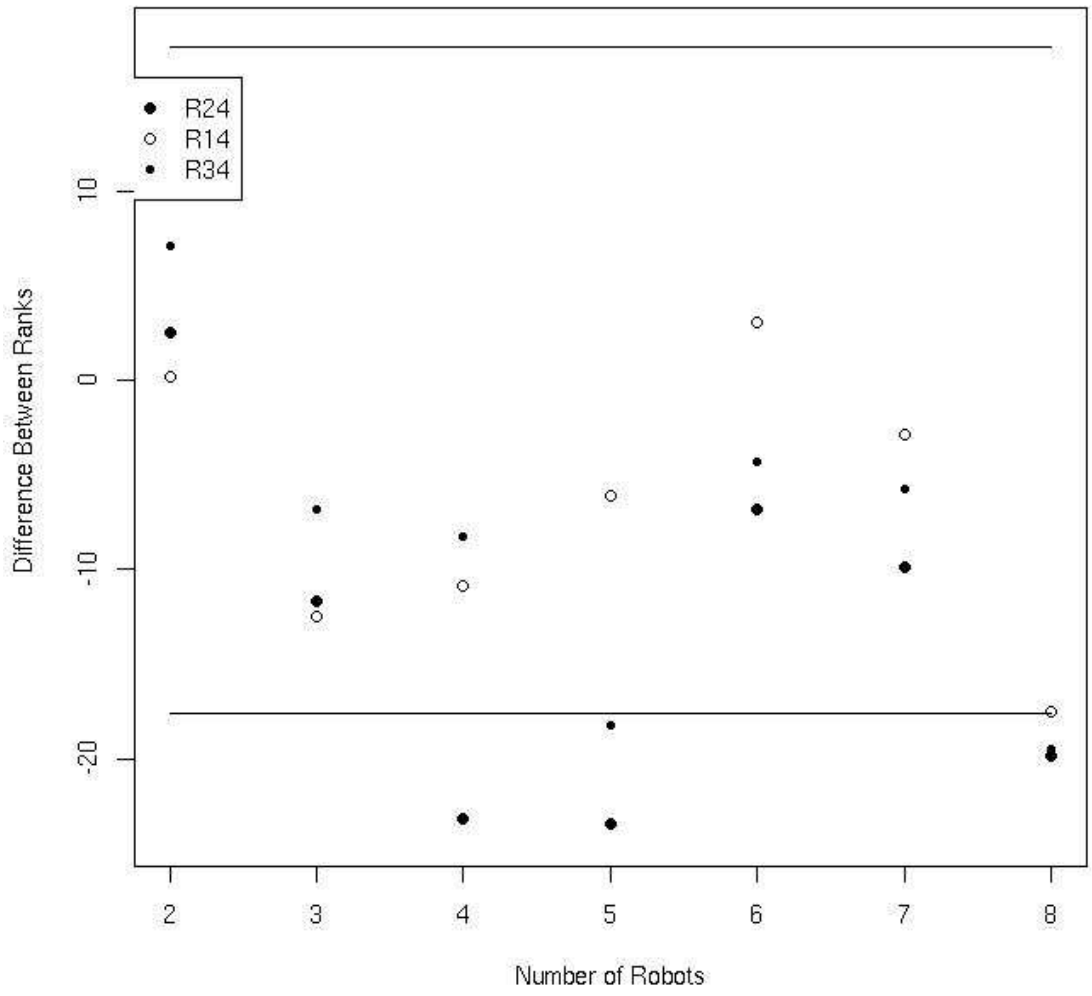


Figure 7.5: Differences between ranks for environment 1: R14 — differences between the non-sharing system and the hybrid system. R24 — differences between the pessimistic system and the hybrid system. R34 — differences between the optimistic system and the hybrid system. Points in between the two horizontal lines are not significant.

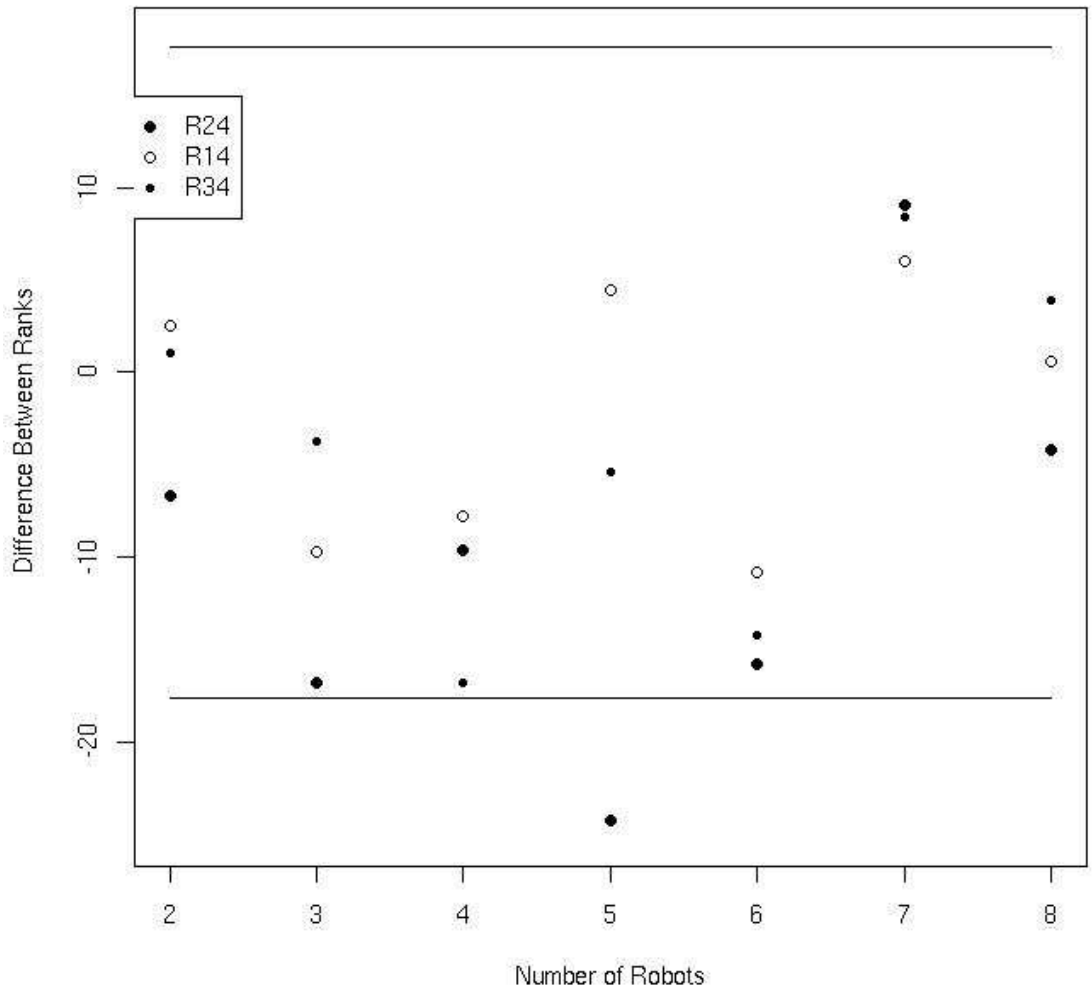


Figure 7.6: Differences between ranks for environment 2: R14 — differences between the non-sharing system and the hybrid system. R24 — differences between the pessimistic system and the hybrid system. R34 — differences between the optimistic system and the hybrid system. Points in between the two horizontal lines are not significant.

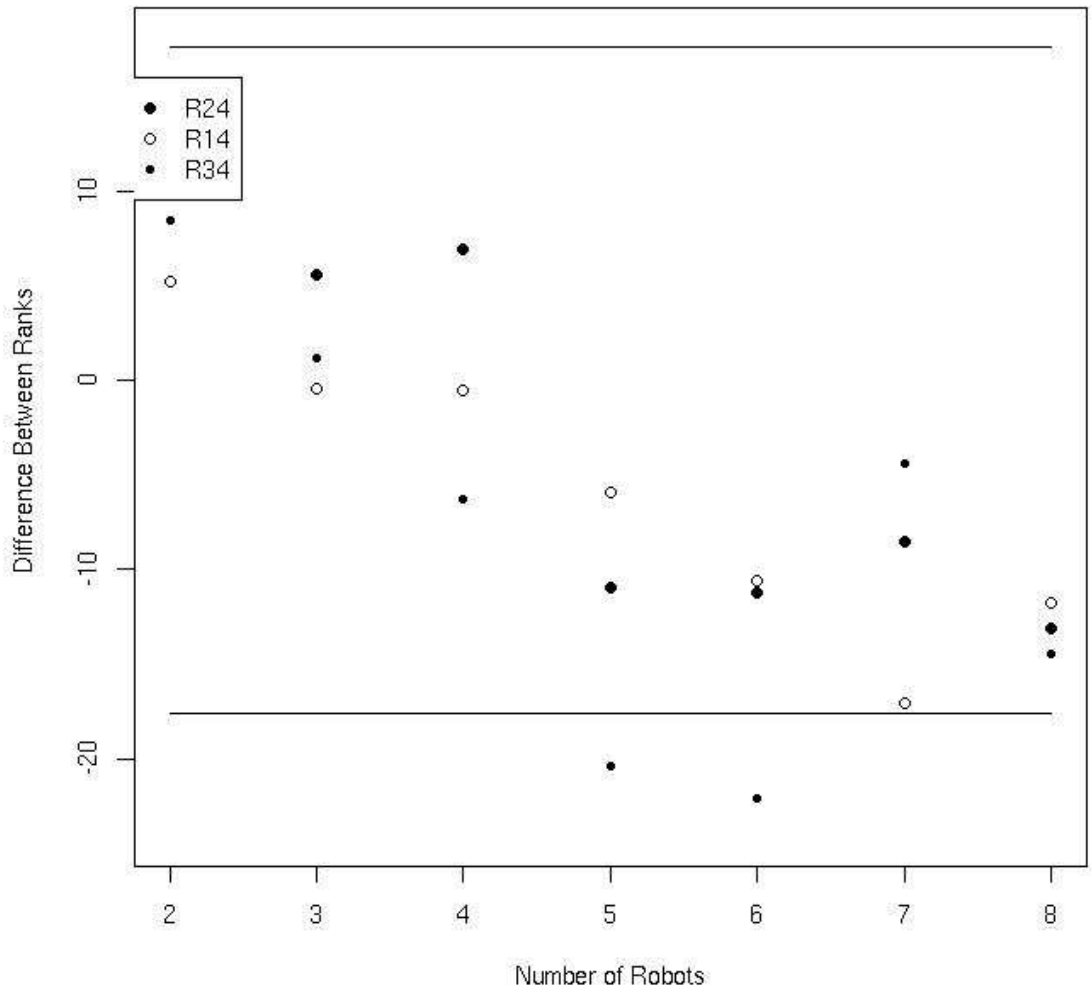


Figure 7.7: Differences between ranks for environment 3: R14 — differences between the non-sharing system and the hybrid system. R24 — differences between the pessimistic system and the hybrid system. R34 — differences between the optimistic system and the hybrid system. Points in between the two horizontal lines are not significant.

environment 3, the hybrid system has the same performance as the non-sharing and pessimistic systems. The optimistic system performs better with 5 to 6 Miabots. This can be seen in figure 7.7 and table 7.2.

7.3.2 Comparison Across Size

Again, the Friedman test (see section 5.3.2) was used to compare the performance of each system with differing group sizes. However, the number of Miabots in the experiment had no significant effect on the performance of the hybrid system in any of the environments.

7.3.3 Comparison Across Environments

Again, the Kruskal-Wallis test was employed to compare system performance across the three different environments. This time, however, the test used the parameters from section 5.3.1. As only values above 11.8 (to 1 d.p.) are significant, it can be seen that the hybrid system clearly performs better in environment 3 than in environment 1. This is shown in figure 7.8 and table 7.3. The performance across the other environments are mixed, with the hybrid system performing better in environment 2 than in environment 1 with 4 and 7 or more Miabots. In contrast, the hybrid system performs better in environment 3 than in environment 2 in just two cases (3 and 6 Miabots).

Table 7.3: Significant differences between means, hybrid system, to 1 d.p.

	2	3	4	5	6	7	8
$R12$	9.5	6.3	**14.3**	9.2	-1.2	**18.0**	**15.3**
$R13$	**18.6**	**21.8**	**23.9**	**14.0**	**18.3**	**26.3**	**24.5**
$R23$	9.2	**15.5**	9.5	4.8	**19.4**	8.3	9.2

7.3.4 Discussion of Single Target Results

The results in this section clearly show that the hybrid system performs best in the sparse environment (environment 3). This is probably due to two main reasons.

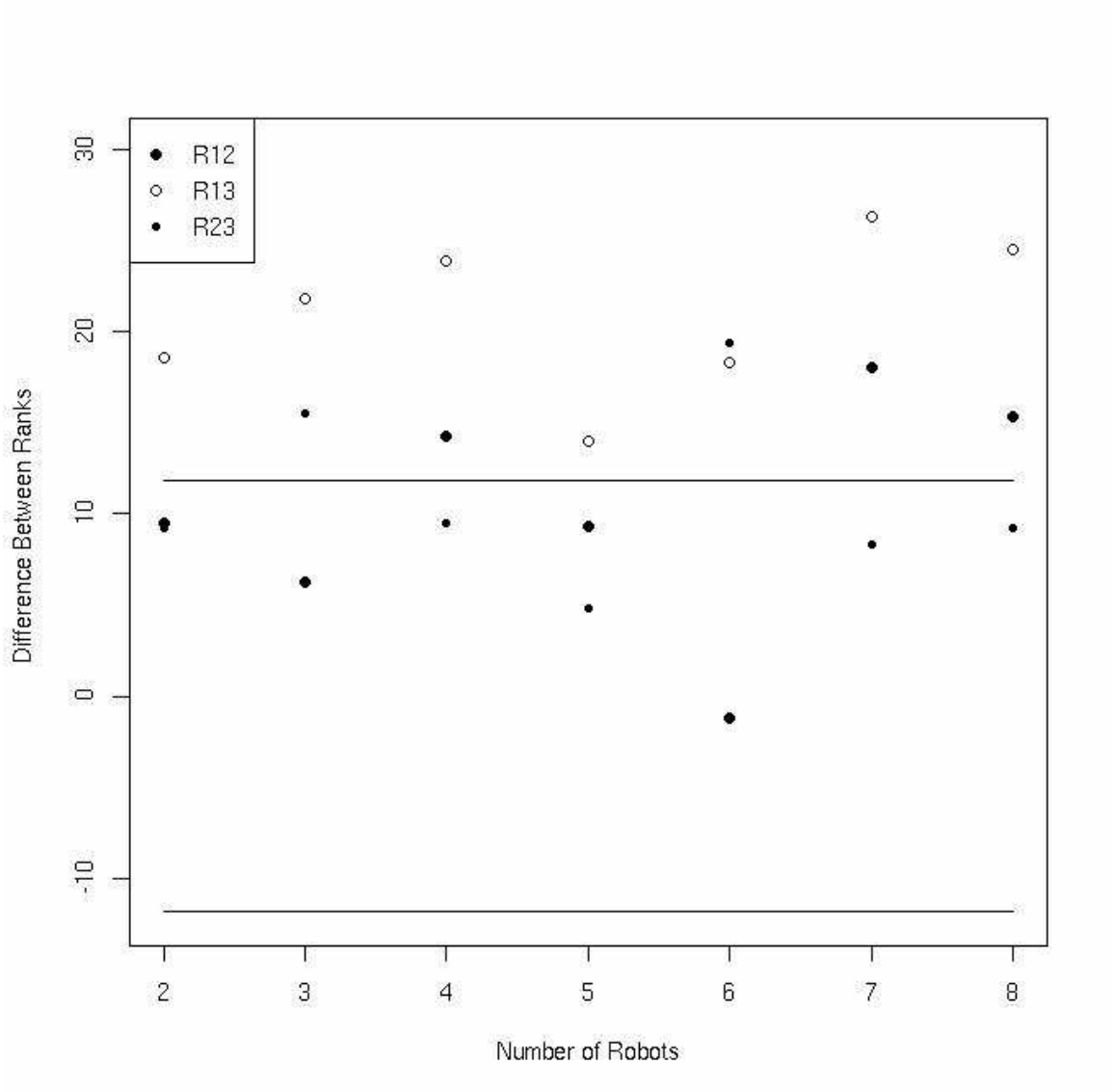


Figure 7.8: Differences between ranks for the hybrid system: R12 — differences between Environment 1 and Environment 2. R13 — differences between Environment 1 and Environment 3. R23 — differences between Environment 2 and Environment 3. Points in between the two horizontal lines are not significant.

Firstly, the more sparse the environment, the more valid paths/goals the wavefront algorithm can plan, which leads to a greater area of the environment being explored. Secondly, as the environment was less cluttered, the ND algorithm could move the Miabot at its maximum velocity more frequently. The number of Miabots within the hybrid system had no bearing on its performance. This is due to the relatively small increase in group size, which did not lead to any substantial increase in probability that the global planner would select target locations near the actual position of the target within the environment. The hybrid system performs as well as the non-sharing system, but worse than the sharing systems. The optimistic system performs better with 5 to 6 Miabots in the sparse environment (environment 3), the pessimistic system performs better in the cluttered environment (environment 1) with 5 or less and 8 Miabots, whilst the optimistic system performs better with 5 or 8 Miabots. This is because the sharing systems react to the least resistance in the shared potential field, whereas the hybrid system plots a path of least resistance to a goal. Hence, the shared potential field is more adaptable to environmental changes than the wavefront algorithm i.e. other Miabots moving in the environment.

7.4 Summary

The potential field sharing system (both pessimistic and optimistic) was categorised as a reactive system. As such it was compared against a non-reactive system. The hybrid class of system was chosen over a deliberative system due to its inherent weaknesses in highly dynamic environments.

It has been shown that the hybrid system has a similar performance to the non-sharing system. It has also been shown that the size of the multi-robot system has no bearing on the hybrid system's performance. This is thought to be due to the relatively small increase in group size, which did not lead to an increased probability that the global planner would select target locations near the actual position of the target within the environment. The hybrid system performs best in the sparse environment. This is probably due to the wavefront algorithm being able to plot more goals within the environment, increasing the chances of a goal near the target. The

ND algorithm can also move the Miabot at its maximum speed whilst in obstacle free zones.

The pessimistic system performs better than the hybrid system in the cluttered environment with 5 or less Miabots. The optimistic system performs better than the hybrid system with 5 or 6 Miabots in the sparse environment. The sharing systems perform better than the hybrid system due to the sharing systems reacting to areas of least resistance, whereas the hybrid system plots a path of least resistance. In the highly dynamic environment used in the experimentation this reactive design was more adaptable.

The experiments, presented in this chapter, have helped this project meet one of its goals given in chapter 1.

- **To design and implement a multi-robot system that is not reliant upon information gathered *a priori*.**

  In the experiments, discussed in this chapter, the potential field sharing method system is never given any information concerning the environment *a priori*.

The findings from the experiments discussed in this chapter have been published in the following conference proceedings:

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman, "Real-world Evaluation of a Novel Potential field sharing method", *In the 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008)*, Linz, Austria, 19th-21st June, pp 73-78, 2008.

An extended version of the paper is to be published.in the following journal:

- J.L. Baxter, E.K. Burke, J.M. Garibaldi & M. Norman,"Shared Potential Fields and their Place in a Multi-Robot Coordination Taxonomy", *Robotics and Autonomous System*, **to be published**, 2009.

Chapter 8

Conclusions

In this chapter, the contribution made by this research to the field of multi-robot systems will be discussed. The known limitations of the multi-robot system proposed will be explored, and directions of future research will be suggested.

8.1 Thesis Overiew

In this thesis a new multi-robot system was introduced. The system is made up of two levels. The local level is defined as a group of robots within an arbitary distance from one another. These robots are able to communicate and co-ordinate with one another. The global level is defined as the interaction (or lack of) between robots outside of these local groups. These robots do not communicate or co-ordinate with one another. The aim of the system is to reduce the amount of communications load, which is a source of failure within multi-robot systems. As communication only occurs within the local levels the communications load is reduced. The systems communication and co-ordination strategy was based upon the concept of sharing potential field information within dynamic local groups. Each member of the multi-robot system creates their own potential field based upon individual sensor readings. Team members that are assigned to local groups share their individual potential fields in order to create a combined potential field which reduces the effect of sensor noise. This, therefore, allows the team members to make better decisions. Two variants of the system were proposed termed “pessimistic” and “optimistic” in terms of their belief in sensor

data. It has been shown that through real robot experiments that when performing a search type task in a cluttered environment it is advantageous to be "pessimistic". Whilst, in a sparse environment it is advantageous to be "optimistic". A number of experiments, both in simulation and in laboratory environments, compared the performance of the system against a non-sharing control and a hybrid system. It was shown that the proposed system significantly outperformed the other methods in the search type problem.

8.2 Major Contributions

As stated in chapter 1, the following contributions to research have been made:

1. A new type of multi-robot system, which performs no co-ordination or communication at the global level, but is weakly co-ordinated at the local level, has been introduced in chapter 4. This contribution was also peer reviewed in the following publications:[12, 13, 14, 15, 10].

2. A method of sharing information through fusing sensory information into a potential field has been shown to be a more effective method of communication and co-ordination in a search type task than a non-sharing control system (see chapters 5 and 6) and a hybrid system (see chapter 7). This contribution was also peer reviewed in the following publications: comparison against a non sharing control [12, 13], comparison against a hybrid system [14, 15]. In order to investigate the effectiveness of the system on a different task, it would be necessary to fuse other types of sensor data into the potential field. These sensors would be task dependent e.g. the blob-finder data would be needed when conducting a surveillance type task, to encourage the robot to investigate unexpected phenomenon.

3. It has been shown that taking a more pessimistic view in terms of sensor belief is advantageous in cluttered environments, whilst performing a search type task. It has also been shown that taking a more optimistic view is advantageous in

sparse environments, whilst performing a search type task on real robots (see chapter 6). This contribution was also peer reviewed in the following publication: [10]. The type of sensor data fused into the potential field would also have an effect on the performance of the different variants of the system in different scenarios. Currently only ultra-sonic data is used and so the increase in performance is due to the number of reduced collisions or trap situations during the search task. Although of benefit in other tasks as well, there may be other factors that effect the performance of the system in other tasks. For example, in the hunting type of task the speed of the of the "prey" would be important.

8.3 Discussion of Goals Achieved

In this section the goals set out in chapter 1 and achieved in this thesis will be discussed in detail.

8.3.1 A New Multi-robot System

The multi-robot system described in chapter 4, is not constrained to one category of Farinelli *et al.*'s taxonomy (described in detail in section 2.4.2). The system is *unaware* at the global level, yet *weakly-co-ordinated* at the local level.

The global level (elements of the system outside of an individual robot's local group radius) is defined as *unaware* because at this level robots are not assigned to local groups and, therefore, have no concept of team members. The local level (elements of the system within an individual robot's local group radius) is defined as *weakly co-ordinated* as, at this level, robots are assigned to local groups and so co-ordinate implicitly through the use of shared potential fields. However, no explicit co-ordination occurs as robots do not have the ability to distinguish team members.

This two level architecture allows the system to benefit from team co-ordination, but not be dependant upon it to make decisions. As such, the system can be said to be fault tolerant with respect to the loss of robots or communications failure. If there was a communications failure within the system, the robots would simply revert to their *unaware* state of operation - making individual decisions based upon individual

sensor readings. If a robot within a local group has a motor failure, it will simply become a stationary sensor device which becomes part of a local group if and when other mobile robots mover near it. Should a robot sensors malfunction, this has little effect on the actions of other robots within its local group, as all robots are treated a obstacles within the system (due to the robots inability to distinguish team members).

8.3.2 Shared Potential Fields

The potential field sharing method outlined in chapter 4 has been shown, through experimentation, to be effective.

The two systems that implemented potential field sharing as a communication and co-ordination method significantly outperformed a non-co-ordinated system (see chapter 5 and 6). This showed that there was an advantage to be had by co-ordinating a group of robots whilst attempting the task set during the experimentation. More importantly the potential field sharing method also outperformed a system which implemented co-ordination by broadcasting plan information (see chapter 7). This showed that there was no advantage to giving information to the systems *a priori*.

This abstraction of sensory information into a potential field allows the design and implementation of a relatively simple action selection method. This is general enough to be deployed on a wide range of robots and over a wide range of tasks. As is discussed in section 8.5, it is proposed that the system could be adapted to numerous multi-robot problem domains. Another benefit of the system design is low expense in terms of both computation and communication. This allows the system to be implemented on low cost hardware, making the event of a robot failure an indifferent one, as the robot can be replaced with relatively little expense.

8.3.3 Reliance upon *a priori* Information

It has been shown that *a priori* information has little impact on system performance during the task carried out in experimentation. The potential field sharing method significantly outperformed a system which was given a map of the environment *a priori*, and co-ordinated its efforts to solve the problem based upon this map (see

chapter 7).

In many tasks (especially those related to the search task tackled in this thesis i.e. Urban Search and Rescue (USAR)), complete *a priori* information about the environment is likely to be infeasible. As such, designing a system that is not dependent on such information is critical to performance. This is not to say that a system cannot benefit from partial information of the environment, but such information should be treated as an uncertain additional source to any system. For example, a blueprint to a building could have its uses during a search and rescue operation — which areas of the building are more likely to have people in. However, the accuracy of the blueprint should not be relied upon and should be constantly measured against sensory input.

8.3.4 Implicit Communication

The potential field sharing method, introduced in chapter 4, did not at anytime explicitly co-ordinate team members i.e. team members did not transmit information to other (specific) robots in order to co-ordinate actions to perform a given task. Instead, co-ordination was an emergent property of combined potential fields. The robots within the system broadcast their individual potential fields within a local radius, any other robot within that local radius used that potential field in conjunction with its own to make a decision. There was no concept of "team" at either end of this process. The transmitting robot did not know which (if any) robot would receive the information. The receiver of the message did not know where the message came from.

This implicit method of co-ordination negates the need for more complex explicit co-ordination strategies, which inevitably result in the need for more complex and therefore expensive robotic hardware. Implicit co-ordination also results in less frequent communications (no need for negotiations) and hence limits a substantial source of failure (communications loss). If there is communications loss the effectiveness of the system is reduced, but it is still able to complete the task (as discussed in chapter 4). The resultant relatively simple communications are also not a burden on the communications bandwidth.

8.4 Limitations of the Proposed System

The major limitation of the sharing systems is their reliance upon accurate positional readings (provided by an overhead tracking system in the experiments presented in this thesis). In the real world, the robots within the system would (most probably) have to rely largely on their own internal readings of position (e.g. odometry). Due to limitations with the Miabots, it was not possible to experiment using only the Miabots' odometry readings to supply position. However, if the reliability of the odometric readings was improved and techniques such as particle filtering were incorporated into the system, this reliance upon the tracking system could be removed.

The centralised nature of the system is also a weakness because if the global tracking system fails, then the whole system is reverted to the non-sharing multi-robot system and, as has been shown in this thesis, the performance of the system would therefore be heavily affected. If the computer facilitating the co-ordination was to fail, this would be a catastrophic situation, as the whole system would fail due to none of the robots being controllable.

As shown in section 4.2.4, the behaviour of the system is influenced by the sequence in which the robots compare forces, leading to either what is termed normal behaviour or to more extreme behaviour. Currently, the sequence depends upon the ID assigned to each robot before the experiment begins. Hence, the behaviour of a robot at any given time cannot be determined *a priori*. In an industrial application this would need to be addressed, most probably by introducing a check into the system to remove extreme behaviour.

Currently the system makes no attempt to filter out noisy input. Improving the system to handle this sensor noise could improve the system's reliability and performance. A number of methods exist to reduce the effect of noise. These include the use of probabilistic algorithms [89] and sensor fusion [68]. It is believed that once sensor noise is reduced, the performance of the system in the laboratory will be more akin to the performance of the system under simulation.

8.5 Directions of Future Research

There a number of possible future research directions:

1. Increase scale of experiments
2. Move to a distributed architecture
3. Further system investigation
4. New multi-robot tasks

These directions are discussed in detail in the following sub-sections.

8.5.1 Increase Scale of Experiments

As mentioned in chapter 6, the size of the environment used during the experimentation was limited due to the number of overhead cameras available. The size of the environment can only be increased if the number of cameras in the laboratory was to be increased, hence increasing the total area of coverage. Alternatively, the size of the environment could be improved by removing the system's reliance on the global tracking system and by improving the reliability of the Miabots' odometric readings, and so removing the artificial restriction on size. Having a larger environment would also allow the system to be extended to a very large scale robotic (VLSR) system (hundreds of robots) such as the one discussed in section 2.5.5. The results from chapter 6 show that eight robots was the realistic limit for the current environment size.

8.5.2 Move to a Distributed Architecture

It would be interesting to move the co-ordination servers/clients from the remote computer (centralised system) onto the individual Miabots (distributed system). This would eliminate the potential single point of failure (the remote computer). In a truely distributed system, if any of the Miabots failed, the system would simply continue. In fact, the system would not even take note, having no concept of other robots

or team mates. The benefits of the Miabots tracking their own position within the environment and broadcasting this information to other Miabots within a local group are also similar in nature. If a Miabot encounters a fault and stops transmitting its position, it would be removed from any local group and so would be treated as an obstacle by the other Miabots within the environment. The faulty Miabot itself would revert to the non-sharing behaviour.

As discussed in chapter 3, moving to a distributed system could feasibly reduce the communication efficiency of the system, due to the serial nature of Bluetooth. However, if the hardware/communication protocol was changed to one that was not serial in nature, e.g. RF broadcasts, this issue could be solved. The distributed system would bring other issues to bear, which are not encountered in the centralised system. As the Miabots would be tracking their own position within the environment, any errors that occurred would be accumulative and poor system performance could ensue. Communication would also not be guaranteed as in the centralised system (guaranteed as long as the central computer does not fail). This would not stop the system from working as discussed previously, but it would impact on system performance.

8.5.3 Further System Investigation

As mentioned in chapter 4, in simulation a local group radius of $2m$ was used and in the laboratory a radius of $75cm$ was set. These distances were chosen arbitrarily. It would be of interest to see what, if any, impact the local group radius would have on system performance. It is hypothesised that over a certain range, the effect on performance would be detrimental. As the further away group members are from one another, the less relevant their sensor information becomes to one another. An example is shown in figure 8.1, in which robots A and C are in different parts of the environment, which differ significantly in terms of object density (robot A's environment is cluttered, whereas robot C's environment is sparse). If they were to share information, they would have a distorted view of the environment and, hence, could make poor decisions. Conversely, the smaller the local group radius, the more likely

it is that individual robots will revert to the non-sharing mode of behaviour.

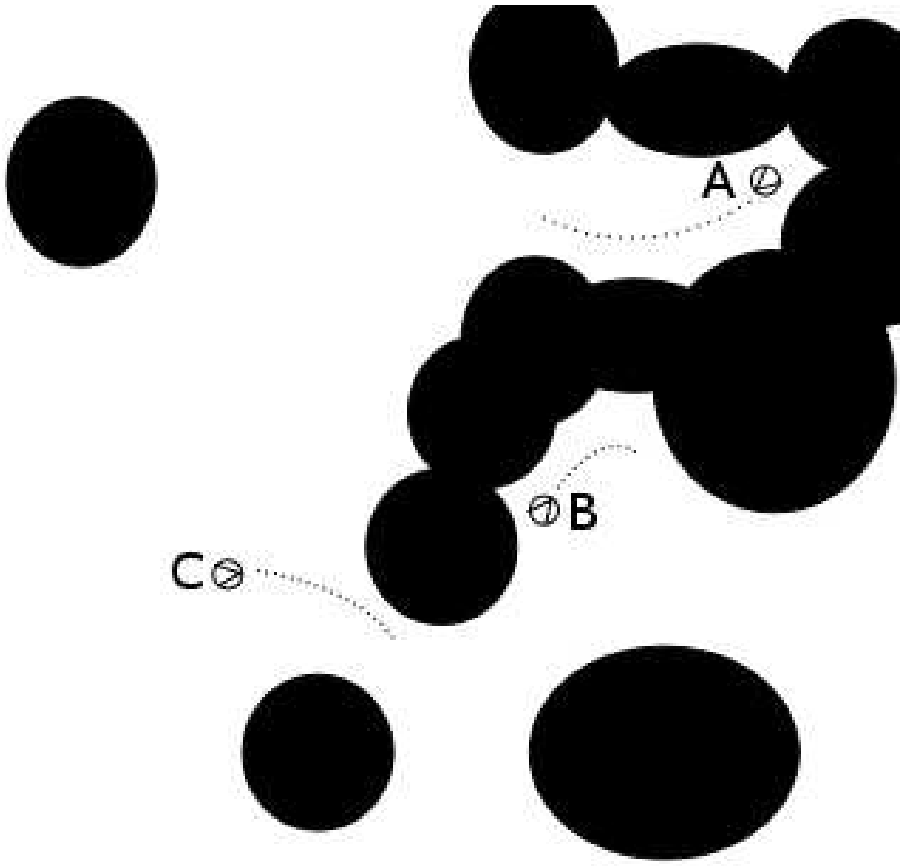


Figure 8.1: Poor local group radius choice: The range chosen is too large, resulting in robots having a distorted view of the environment.

In the current system, only ultra-sonic data is shared through the potential field. The architecture could be extended to allow the input of multiple sensors. In particular, blob-finder data could be fused within the potential field in order to encourage robots to move towards areas that possibly include targets (blob data above a valid threshold). The blob-finder could also be used to aid robot dispersal, as currently other robots are treated as obstacles, rather than the special case obstacle which they are.

As described in chapter 4, the calculations used in the sharing systems is just a choice of the maximum force or the minimum force. It may be of interest to see the effect that different methods of combining information could have on the system

performance. For example, using the mean of the values or assigning a belief value to each robot (belief in sensor readings accuracy) and performing a comparison on that value rather than that of the force value.

In chapter 4, Coulomb's law is simplified to an inverse square law. It would be interesting to see what effect implementing Coulomb's law fully would have on system performance. It would enable the assignment of different types of obstacles to different unit values. For example, if other robots had a higher unit charge, the repulsion rate would increase, helping the system disperse more evenly throughout the environment.

8.5.4 New Multi-Robot Tasks

Finally, it would be of interest to apply the proposed system to other common multi-robot system problems such as serach and rescue, formation control, the coverage problem, hunting and robot football.

In the current system, only a search problem is attempted. It would be interesting to extend the system to implement a search and rescue type problem; once the target has been found it needs to be rescued (taken to a designated position within the environment). This rescue operation may require multiple robots to complete it. That is, if the target's weight is greater than a single robots servo limitations, multiple robots will be needed to "rescue" it.

In order to attempt the formation control problem as defined in chapter 2, the system would have to be adjusted so that robots that are a part of a group are attracted/repelled to/from, each other to allow them to form formations, as in the work presented in section 2.5.2.

The coverage problem (as defined in chapter 2) could be attempted with the current system, with the target detection removed. Group member recognition would have to be improved, as detailed above, in order to enable the robots to spread out from one another to improve coverage. A method of halting the robots when they are at an optimum position within the environment would also be needed.

Robot football has become a popular test bed for multi-robot research over the

last decade. Indeed, the University of Nottingham sent a team of Miabots to the FIRA 2005 world championship [11]. In order to apply the system discussed in this thesis to the problem of robot football, a number of alterations and improvements would need to take place. In the MiroSot league (in which the University took part in 2005), the environment consists of a non-static target (the ball), moving obstacles (opposition members and team mates) and static obstacles (pitch enclosing walls). If the ball is made the main attractor within the environment and the opposition and team mates made to be repellers, team mates could also be attractors to help maintain formations, as per the formation control problem. As a global view of the environment is available throughout a match, the definition of a local group could be altered. Defenders could be assigned to one group, forwards to another and the goal keeper to its own group. This would allow team members outside of the local groups to be treated differently to those inside (as obstacles in fact). See figure 8.2 for an example of local groups within robot football. If team members wished to change roles whilst in a match, this would simply involve leaving one local group and joining another. Unlike in the search problem when this is based upon distance from other robots, the switch could be based upon a "coach" agent's decision (based upon the robot's current location in relation to the ball, team mates and opposition players).

The hunting problem (as defined in chapter 2) could readily be attempted with the current system, as the problem is just an extension of the search problem (a moving target). Again, this is reliant upon the improvement of object recognition.

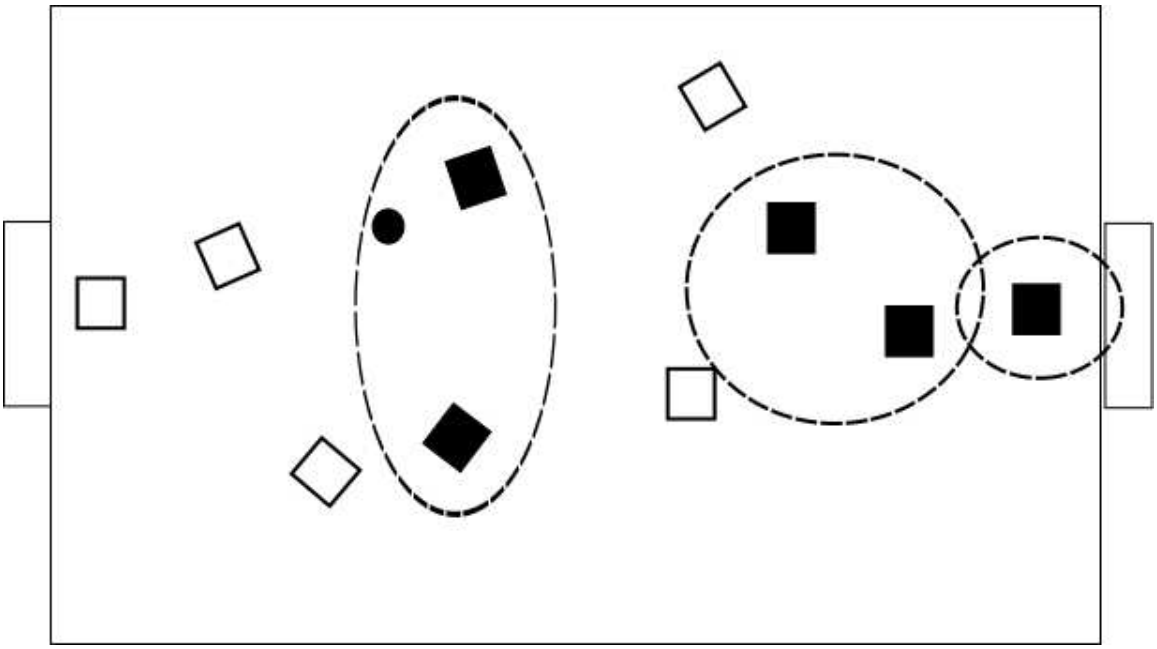


Figure 8.2: Local groups within robot football: Local groups are shown in dashed ellipses. Our team are the black squares, the opposition team are white squares. The ball is the black circle.

References

[1] R. C. Arkin. Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[2] R. C. Arkin. Reactive Robotic Systems. In M. A. Arbib, editor, *The Handbook of brain theory and neural networks*, pages 793–796. MIT Press, Cambridge, MA, 1995.

[3] R. C. Arkin and J. Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *The Seventh International Workshop on Advanced Motion Control (AMC'02)*, 2002.

[4] R. C. Arkin and D. C. Mackenzie. Planning to Behave: A Hybrid Deliberative/Reactive Robot Control Architecture for Mobile Manipulation. In *International Symposium on Robotics and Manufacturing*, pages 5–12, 1994.

[5] T. Balch. Taxonomies of Multirobot Task and Reward. Technical report robotics institute, CMU, 1998.

[6] T. Balch and R. C. Arkin. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robotics*, 1(1):1–25, 1994.

[7] T. Balch and R. C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[8] T. Balch and M. Hybinette. Social Potentials for Scalable Multirobot Formations. In *IEEE International Conference on Robotics and Automation*, pages 73–80, 2000.

[9] M. A. Batalin and G .S. Sukhatme. Spreading Out: A Local Approach to Multi-Robot Coverage. In *The 6th International Symposium on Distributed Autonomous Robotics Systems*, pages 373–382, Fukuoka, Japan, June 2002.

[10] J. L. Baxter, E. K. Burke, J. M. Garibaldi, S. Groenemeyer, and M. Norman. Multi-robot Co-ordination Using Shared Potential Fields. *IEEE Transactions on Robotics*, 2009. Submitted for Review.

[11] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. Statistical Analysis in MiroSot. In *RoboWorld FIRA World Congress 2005*, Singapore, December 2005. CD-ROM ONLY.

[12] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. The Effect of Potential Field Sharing in Multi-Agent Systems. In *The 3rd International Conference on Autonomous Robots and Agents (ICARA 2006)*, pages 33–38, Palmerston North, New Zealand, December 2006.

[13] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. Multi-Robot Search and Rescue: A Potential Field Based Approach. In S. Mukhopadhyay and G. S. Gupta, editors, *Autonomous Robots and Agents*, volume 76 of *Studies in Computational Intelligence*, pages 9–16. Springer-Verlag, 2007.

[14] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. Performance Comparison of the Potential Field Sharing Method Against a Hybrid System. In *The 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008)*, pages 73–78, Linz, Austria, April 2008.

[15] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. Shared Potential Fields and their Place in a Multi-Robot Coordination Taxonomy. *Robotics and Autonomous System*, 2009. To be Published.

[16] E. Bicho and S. Monterio. Formation control for multiple mobile robots: a non-linear attractor dynamics approach. In *International Conference on Intelligent Robots and Systems(IROS 2003)*, pages 2016– 2022, 2003.

[17] A. L. Blum and M. L. Frust. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[18] B. Bonet and H. Geffner G. Loerincs. A Robust and Fast Action Selection Mechanism for Planning. In *14th National Conference of the American Association for Artificial Intelligence*, pages 714–719, 1997.

[19] B. Bonet and H. Geffner. High-level Planning and Control with Incomplete Information Using POMDP's. In *American Association for Artificial Intelligence Fall Symposium on Cognitive Robotics*, 1998.

[20] B. Bonet and H. Geffner. Planning with Incomplete Information as Heuristic Search in Belief Space. In *5th International Conference on Artificial Intelligence Planning and Scheduling*, pages 52–61, 2000.

[21] V. Braitenburg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, 1984.

[22] T. V. Brook. Drones' supply short of demand. USA Today, March 2007.

[23] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. A.I. Lab Memo 864, MIT, September 1985.

[24] R. A. Brooks. Achieving Artificial Intelligence Through Building Robots. A.I. Lab Memo 899, MIT, May 1986.

[25] R. A. Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[26] R. A. Brooks. The Behaviour Language; User's Guide. A.I. Lab Memo 1227, MIT, April 1990.

[27] R. A. Brooks. Intelligence Without Reason. A.I. Lab Memo 1293, MIT, April 1991.

[28] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[29] H. D. Burkhard, M. Hannebauer, J. Wendler, H. Myritz, G. Sander, and T. Meinert. BDI Design Principles and Cooperative Implementation in Robocup. In *RoboCup-99: Robot Soccer Worldcup III*, number 1856 in Lecture Notes in Artificial Intelligence, pages 351–541. Springer-Verlag, 2000.

[30] Z. Cao, M. Tan, L. Li, N. Gu, and S. Wang. Cooperative Hunting by Distributed Mobile Robots Based on Local Interaction. *IEEE Transactions on Robotics*, 22(2):403–407, April 2006.

[31] L. Chaimowicz, M. F. M. Campos, and V. Kumar. Dynamic Role Assignment for Cooperative Robots. In *IEEE International Conference on Robotics and Automation*, pages 293–298, Washington DC, 2002.

[32] A. Cimatti and M. Roveri. Conformant Planning via Model Checking. In *5th European Conference on Planning: Recent Advances in AI Planning*, pages 21–34. Springer-Verlag, 1999.

[33] Staff Sgt. D. Clare. California Air National Guard embraces new mission. http://www.af.mil, August 2006.

[34] C. M. Clark, S. M. Rock, and J.-C. Latombe. Dynamic Networks for Motion Planning in Multi-Robot Space Systems. In *7th International Symposium on Artificial Intelligence*, Nara, Japan, 2003.

[35] C. M. Clark, S. M. Rock, and J.-C. Latombe. Motion planning for multiple mobile robot systems using dynamic networks. In *IEEE International Conference on Robotics and Automation*, pages 4222–4227, 2003.

[36] B. Damas, P. Lima, and L. Custódio. A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer. In G. A. Kaminka, P. U. Lima, and R. Rojas, editors, *RobotCup 2002*, number 2752 in Lecture Notes on Artificial Intelligence, pages 65–77. Springer-Verlag, 2003.

[37] M. B. Dias and A. Stentz. Opportunistic Optimization for Marker-Based Multirobot Control. In *International Conference on Intelligent Robots and Systems*, pages 2714–2720, October 2002.

[38] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A Taxonomy for Swarm Robotics. In *International Conference on Intelligent Robots and Systems*, pages 26–30, Yokohama, Japan, July 1993.

[39] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A Taxonomy for Multi-Agent Robotics. *Autonomous Robots*, 3(4):375–397, December 1996.

[40] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An Approach to Planning With Incomplete Information. In *The Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 115–125. Morgan Kaufmann, 1992.

[41] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot Systems: A Classification Focused on Coordination. *IEEE Transactions on Systems, Man and Cybernetics B*, 34(5):2015–2028, October 2004.

[42] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

[43] C-H. Fua and S. S. Ge. COBOS: Cooperative Backoff Adaptive Scheme for Multirobot Task Allocation. *IEEE Transactions on Robotics*, 21(6):1168–1178, December 2005.

[44] V. Gazi. Swarm aggregations using artificial potentials and sliding-mode control. *IEEE Transactions on Robotics*, 21(6):1208–1214, December 2005.

[45] B. P. Gerky and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. Technical Report CRES-03-013, USC, Center for Robotics and Embedded Systems, July 2003.

[46] B. P. Gerky, R. T. Vaughan, and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *International Conference*

on Advanced Robotics (ICAR 2003), pages 317–323, Coimbra, Portugal, June 2003.

[47] J. D. Gibbons. *Nonparametric Methods for Quantitative Analysis*. American Sciences Press, Columbus, Ohio, third edition, 1997.

[48] J. Greensmith, U. Aickelin, and S. Cayzer. Introducing dendritic cells as a novel immune inspired algorithm for anomaly detection. In *The 4th International Conference on Artificial Immune Systems (ICARIS 2005)*, number 3627 in Lecture Notes in Computer Science, pages 153–167, Banff, Canada, 2005. Springer-Verlag.

[49] A. Howard, M. J. Matarić, and G. S. Sukhatme. An Incremental Deployment Algorithm for Mobile Robot Teams. In *International Conference on Intelligent Robots and Systems*, pages 2849–2854, October 2002.

[50] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *The 6th International Symposium on Distributed Autonomous Robotics Systems*, volume 22, pages 299–308, June 2002.

[51] J. S. Jennings and C. Kirkwoord-Watts. Distributed Mobile Robotics by the Method of Dynamic Teams. In T. Luth, P. Dario, and H. Worn, editors, *Distributed Autonomous Robotic Systems*, volume 3, pages 47–56. Springer-Verlag, New York, 1998.

[52] J. S. Jennings, G. Whelan, and W. F. Evans. Cooperative search and rescue with a team of mobile robots. In *IEEE International Conference on Advanced Robotics*, pages 193–200, July 1997.

[53] H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Tenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, Portland, Oregon, 1996.

[54] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.

[55] Y. Koren and J. Borenstein. Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1398–1404, Sacramento, CA, 1991.

[56] S. Leroy, J. P. Laumond, and T. Siméon. Multiple path coordination for mobile robots: A geometric algorithm. In *16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1118–1123, Stockholm, Sweden, 1999.

[57] X.-W. T. Li and J. Baltes. An Intuitive and Flexible Architecture for Intelligent Mobile Robots. In *The 2nd International Conference on Autonomous Robots and Agents*, pages 52–57, 2004.

[58] V. Lumelsky and K. R. Harinarayan. Decentralized Motion Planning for Multiple Robots: The Cocktail Party Model. *Autonomous Robots*, 4:121–135, 1997.

[59] L. S. Martins-Filho, E. E. N. Macau, and R. Rocha. Planning of Unpredictable Trajectories for Surveillance Mobile Robots. In *3rd Internation Conference on Autonomous Robots and Agents*, pages 153–158, Palmerston North, New Zealand, December 2006.

[60] M. J. Matarić. Learning to Behave Socially. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pages 453–462, 1994.

[61] M. J. Matarić. Designing and Understanding Adaptive Group Behavior. *Adaptive Behavior*, 4(1):50–81, December 1995.

[62] M. J. Matarić. Behaviour Based Control: Examples from Navigation, Learning and Group Behaviour. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):323–336, 1997.

[63] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *The Ninth National Conference on Artificial Intelligence*, pages 634–639, July 1991.

[64] D. McFarland. *Animal Behaviour*. Longman Scientific and Technical Harlow, 1985.

[65] J. Minguez and L. Montano. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. *IEEE Transactions on Robotics and Automation*.

[66] J. Minguez, J. Osuna, and L. Montano. A Divide and Conquer Strategy based on Situations to Achieve Reactive Collision Avoidance in Troublesome Scenarios. In *IEEE International Conference on Robotics and Automation*, pages 3855–3862, New Orleans, 2004.

[67] S. Monterio and E. Bicho. A dynamical systems approach to behavior-based formation control. In *IEEE International Conference on Robotics and Automation*, pages 2606–2611, Washington, 2002.

[68] R. R. Murphy. Dempster-Shafer Theory for Sensor Fusion in Autonomous Mobile Robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206, April 1998.

[69] R. Oates, J. Greensmith, U. Aickelin, J. M. Garibaldi, and G. Kendall. The Application of a Dendritic Cell Algorithm to a Robotic Classifier. In L. N. de Castro, F. J. V Zuben, and H. Knidel, editors, *The 6th International Conference on Artificial Immune Systems (ICARIS 2007)*, number 4628 in Lecture Notes in Computer Science, pages 204–115, Santos, Brazil, August 2007. Springer-Verlag.

[70] P. Ögren, E. Fiorelli, and N. Leonard. Formations with a mission: Stable coordination of vehicle group maneuvers. In *The 15th International Symposium on Mathematical Theory of Network Systems*, Notre Dame, IN, August 2002.

[71] A. N. Chand G. C. Onwubolu. A Mobile Robot for Autonomous Book Retrieval. In S. Mukhopadhyay and G. S. Gupta, editors, *Autonomous Robots and Agents*,

volume 76 of *Studies in Computational Intelligence*, pages 101–107. Springer-Verlag, 2007.

[72] L. E. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, Dept. Electr. Eng. Comput. Sci., Mass. Inst. of Technol. Cambridge, MA, 1994.

[73] L. E. Parker. L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behaviour-Based Systems. *Advanced Robotics*, 11(4):305–322, 1997.

[74] L. E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.

[75] D. Parsons and J. Canny. A Motion Planner for Multiple Mobile Robots. In *IEEE International Conference on Robotics and Automation*, pages 8–13, Cincinnati, OH, USA, May 1990.

[76] K. Pathak and S. K. Agrawal. An Integrated Path-Planning and Control Approach for Nonholonomic Unicycles Using Switch Local Potentials. *IEEE Transactions on Robotics*, 21(6):1201–1208, December 2005.

[77] R. Petrick and F. Bacchus. A knowledge-Based Approach to Planning with Incomplete Information and Sensing. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS02)*, pages 37–46, 2002.

[78] R. A. Jarvis and M. S. Marzouqi. Efficient Robotic Search Strategies for Finding Disaster Victims. In *3rd International Conference on Autonomous Robots and Agents*, pages 515–520, Palmerston North, New Zealand, December 2006.

[79] J. H. Reif and H. Wang. Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots. *Robotics and Autonomous Systems*, 27:171–194, 1999.

[80] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. In *Annals of Mathematics and Artificial Intelligence*, volume 31, pages 7–40. Kluwer Academic, 2001.

[81] J. Ren, K. A. McIsaac, and R. V. Patel. Modified Newton's Method to Potential Field Based Navigation for Mobile Robots. *IEEE Transactions on Robotics*, 22(2):384–391, April 2006.

[82] M. Roth, D. Vail, and M. Veloso. A World Model for Multi-Robot Teams with Communication. In *IROS 2003*. under submission.

[83] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination of Multi-Robot Exploration and Mapping. In *The 17th National Conference on Artificial Intelligence*, pages 852–858, 2000.

[84] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly. In *International Symposium on Experimental Robotics*, December 2000.

[85] D. E. Smith and D. S. Weld. Conformant Graphplan. In *The Fifteenth National Conference on Artificial Intelligence*, pages 889–896. AAAI Press, 1998.

[86] A. Stentz and M. B. Dias. A Free Market Architecture for Coordinating Multiple Robots. Technical Report CMU-R1-TR-99-42, Camegie Mellon University, December 1999.

[87] K. Sugawara and M. Sano. Cooperative acceleration of task performance: Foraging behavior of interacting multi-robot system. *Physica D*, 100:343–354, 1997.

[88] A. Tews and G. F. Wyeth. Using Centralised Control and Potential Fields for Multi-robot Cooperation in Robotic Soccer. In T. Ishida, editor, *Pacific Rim International Workshop on Multi-Agents (PRIMA)*, pages 176–190, Singapore, November 1998.

[89] S. Thrun. Probabilistic Algorithms in Robotics. Technical Report CMU-CS-00-126, Carnegie Mellon Univ, Pittsburgh, PA, April 2000.

[90] V. I. Utkin, S. V. Drakunov, H. Hashimoto, and F. F. Harashima. Robot Path Obstacle Avoidance Control Via Sliding Mode Approach. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1287–1290, Osaka, Japan, November 1991.

[91] D. Vail and M. Veloso. Multi-Robot Dynamic Role Assignment and Coordination Through Shared Potential Fields. In A. Schultz, L. Parker, and F Schneider, editors, *Multi-Robot Systems*, pages 87–98. Kluwer: Hingman, MA, 2003.

[92] M. Veloso and P. Stone. Individual and Collaborative Behaviors in a Team of Homogeneous Robotic Soccer Agents. In *The 3rd International Conference on Multi-Agent Systems*, pages 309–316, 1998.

[93] B. B. Werger. Ayllu: Distributed Port-Arbitrated Behaviour-Based Control. Technical Report 99-01, Ullanta Performance Robotics, 1999.

[94] B. B. Werger. Cooperation without deliberation: A Minimal Behavior-Based Approach to Multi-Robot Teams. *Artificial Intelligence*, 110:293–320, 1999.

[95] B. B. Werger and M. J. Matarič. Broadcast of Local Eligibility for Multi-target Observation. In *Distributed Autonomous Robotic Systems*, pages 347–256, 2000.

[96] M. M. Zavlanos and G. J. Pappas. Potential Fields for Maintaining Connectivity of Mobile Networks. *IEEE Transactions on Robotics*, 23(4):812–816, August 2007.

[97] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-Robot Exploration Controlled by a Market Economy. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 3016–3023, May 2002.

Appendix A

Detailed Results

A.1 Simulation 1 Target

Table A.1: Non-sharing system results for environment 1, with 1 target.

	2	3	4	5	6	7	8
1	300	300	300	300	300	300	300
2	300	300	300	300	300	295	300
3	300	300	126	300	300	300	259
4	300	300	146	245	198	300	104
5	300	73	300	300	68	300	300
6	300	300	300	300	120	300	300
7	300	300	300	300	300	300	300
8	300	300	300	233	300	300	300
9	300	146	300	48	300	180	300
10	300	300	131	158	233	202	300
11	300	85	300	54	70	300	300
12	300	300	300	43	300	300	300
13	300	300	300	300	300	156	300
14	300	133	165	300	300	300	300
15	300	300	300	300	300	300	300
16	300	300	300	300	145	300	300
17	300	300	300	300	71	127	300
18	300	76	49	300	93	300	300
19	300	300	300	64	49	300	300
20	300	300	300	56	57	300	300

Table A.2: Non-sharing system results for environment 2, with 1 target.

	2	3	4	5	6	7	8
1	300	63	20	78	18	7	24
2	46	300	21	17	21	21	82
3	300	300	300	23	24	31	13
4	300	300	72	4	32	66	51
5	48	44	18	18	10	83	32
6	37	74	71	20	19	19	22
7	54	52	22	27	12	9	26
8	97	75	300	21	63	18	33
9	73	70	71	24	20	10	19
10	300	300	73	26	81	36	8
11	51	160	24	94	33	62	8
12	300	64	25	71	18	62	25
13	54	65	52	22	23	88	23
14	53	50	68	21	18	10	31
15	300	87	137	24	27	20	35
16	56	55	5	20	156	19	21
17	56	119	67	63	24	19	43
18	300	61	21	22	37	25	42
19	300	52	17	20	9	37	36
20	51	300	18	19	3	22	23

Table A.3: Non-sharing system results for environment 3, with 1 target.

	2	3	4	5	6	7	8
1	76	56	57	65	63	66	300
2	53	53	58	86	99	58	300
3	49	54	58	57	62	57	68
4	57	49	57	64	56	47	49
5	49	56	67	62	47	66	58
6	47	68	59	55	66	116	49
7	52	48	57	86	66	66	91
8	53	49	59	64	47	57	45
9	49	54	64	300	71	300	300
10	53	50	64	49	59	67	104
11	49	300	124	61	54	53	300
12	53	49	61	53	67	300	57
13	49	56	59	68	96	66	48
14	48	55	58	53	61	83	91
15	48	49	53	53	65	59	82
16	50	54	64	44	53	300	300
17	53	67	66	49	49	57	45
18	53	300	124	86	68	300	49
19	49	56	124	61	59	69	300
20	53	300	59	55	46	67	52

Table A.4: Pessimistic system results for environment 1, with 1 target.

	2	3	4	5	6	7	8
1	155	65	141	46	101	75	100
2	300	300	300	300	73	132	107
3	86	275	243	72	48	67	50
4	300	300	70	74	48	79	85
5	300	87	90	108	44	55	300
6	300	300	237	79	45	47	57
7	59	74	61	234	112	92	50
8	300	300	137	300	300	300	50
9	56	109	186	63	300	300	50
10	300	300	65	80	52	57	45
11	159	300	300	44	51	51	98
12	152	300	300	69	300	57	119
13	300	84	90	97	60	300	300
14	58	300	300	300	89	300	112
15	157	300	300	300	47	57	88
16	300	83	68	60	67	300	102
17	90	300	70	52	300	58	49
18	300	70	197	56	51	41	50
19	300	300	215	44	300	69	300
20	95	75	81	300	44	48	300

Table A.5: Pessimistic system results for environment 2, with 1 target.

	2	3	4	5	6	7	8
1	22	24	15	4	21	3	3
2	300	73	33	5	6	4	5
3	24	23	15	73	7	5	5
4	23	66	32	21	21	3	5
5	23	66	30	60	6	3	7
6	25	26	66	4	7	4	13
7	59	24	62	64	7	6	20
8	22	84	19	5	6	4	50
9	93	28	30	5	6	5	5
10	24	55	32	300	7	4	17
11	23	31	28	21	8	7	3
12	300	25	32	21	6	4	4
13	23	76	17	128	4	4	17
14	300	28	34	32	7	5	3
15	172	83	26	19	6	4	7
16	19	76	34	60	7	5	5
17	24	23	17	27	7	4	20
18	19	23	300	66	6	4	21
19	20	300	25	5	7	4	6
20	29	24	16	78	7	4	21

Table A.6: Pessimistic system results for environment 3, with 1 target.

	2	3	4	5	6	7	8
1	70	77	78	58	83	58	300
2	70	80	79	74	57	60	73
3	62	65	58	48	64	300	48
4	70	300	56	66	71	300	61
5	67	300	56	71	86	72	39
6	60	77	73	55	58	55	46
7	73	76	63	86	56	300	300
8	70	79	58	57	61	61	61
9	62	300	54	74	66	59	72
10	70	76	77	66	56	57	39
11	82	75	56	300	58	73	73
12	300	77	74	81	69	50	58
13	67	102	81	67	55	300	57
14	70	53	54	67	72	42	300
15	300	300	87	74	61	72	66
16	68	46	59	73	72	72	56
17	300	69	58	53	48	89	72
18	74	300	79	63	60	300	68
19	74	83	76	66	74	300	74
20	300	300	82	77	66	49	54

Table A.7: Optimistic system results for environment 1, with 1 target.

	2	3	4	5	6	7	8
1	300	300	68	104	45	300	300
2	300	66	300	62	51	300	45
3	300	55	265	47	59	81	242
4	300	300	154	79	51	70	47
5	300	300	75	77	79	47	242
6	300	300	62	62	106	50	65
7	300	300	70	160	86	53	47
8	300	78	67	106	300	59	300
9	300	300	62	76	43	61	52
10	86	300	63	300	46	108	75
11	300	66	66	300	42	73	50
12	252	76	290	84	60	65	68
13	101	300	72	50	46	57	300
14	197	300	67	47	65	300	50
15	300	300	66	72	300	49	119
16	300	129	56	66	79	300	119
17	114	70	300	49	300	47	50
18	300	66	300	258	89	55	49
19	300	70	300	105	42	79	50
20	300	300	300	83	71	50	50

Table A.8: Optimistic system results for environment 2, with 1 target.

	2	3	4	5	6	7	8
1	18	28	64	31	7	4	3
2	197	18	21	18	7	4	21
3	98	24	31	8	62	4	18
4	300	18	31	66	15	4	3
5	300	18	24	6	8	4	5
6	287	59	25	56	9	5	20
7	24	26	17	22	6	4	5
8	272	24	67	47	5	4	13
9	16	35	69	62	5	4	5
10	300	29	17	22	8	4	3
11	29	77	17	31	6	4	3
12	19	46	17	17	32	4	4
13	47	69	30	22	7	4	17
14	67	70	34	4	6	4	21
15	300	78	24	5	4	4	3
16	59	21	16	3	4	4	5
17	17	83	23	20	19	5	3
18	18	24	16	50	3	4	13
19	45	68	33	16	8	5	5
20	228	37	58	19	6	6	20

Table A.9: Optimistic system results for environment 3, with 1 target.

	2	3	4	5	6	7	8
1	70	83	59	72	73	114	72
2	70	118	57	73	60	96	300
3	67	66	78	98	82	74	300
4	74	68	59	69	56	114	77
5	70	300	57	126	59	300	48
6	300	300	68	75	63	300	57
7	300	44	61	300	103	72	51
8	63	67	113	66	60	53	56
9	69	300	78	68	78	45	300
10	68	67	54	52	72	74	72
11	62	300	58	57	83	73	66
12	71	74	78	84	61	64	300
13	81	80	79	300	53	74	68
14	300	300	72	51	60	300	59
15	70	83	87	43	58	69	39
16	300	68	65	64	81	55	77
17	63	52	70	72	69	51	55
18	76	52	59	75	55	58	61
19	67	67	81	300	50	72	68
20	68	81	76	68	57	74	73

A.2 Simulation 2 Targets

Table A.10: Non-sharing system results for environment 1, with 2 targets.

	2	3	4	5	6	7	8
1	300	300	300	47	63	300	52
2	300	300	137	124	63	169	300
3	300	96	300	300	300	300	63
4	300	300	300	300	199	300	63
5	300	96	300	300	60	300	300
6	300	57	300	300	213	300	62
7	300	300	300	119	300	73	197
8	300	57	300	163	300	300	300
9	300	61	300	300	120	50	300
10	300	95	300	300	194	300	300
11	300	300	300	300	60	300	60
12	300	300	300	300	300	169	52
13	300	96	300	239	57	300	51
14	300	300	149	110	192	169	62
15	300	300	300	119	300	169	300
16	300	300	300	300	223	300	300
17	300	57	300	300	300	300	57
18	300	300	300	300	120	300	95
19	300	57	300	300	120	300	300
20	300	228	294	300	300	300	52

Table A.11: Non-sharing system results for environment 2, with 2 targets.

	2	3	4	5	6	7	8
1	300	79	22	18	23	300	23
2	62	161	21	73	23	19	19
3	61	161	22	300	23	22	25
4	300	161	300	38	70	22	23
5	72	161	67	19	23	20	21
6	62	161	300	64	21	21	102
7	300	68	300	300	23	23	23
8	62	161	21	21	19	23	19
9	61	161	28	24	23	33	21
10	20	68	19	73	64	22	26
11	61	161	24	73	23	20	25
12	62	68	18	25	19	20	22
13	69	76	300	19	22	32	26
14	118	161	80	73	19	32	19
15	67	161	20	73	19	22	24
16	67	68	22	112	19	23	25
17	61	68	21	112	19	21	25
18	300	68	22	20	22	22	21
19	300	68	67	20	24	22	21
20	61	300	26	20	19	22	23

Table A.12: Non-sharing system results for environment 3, with 2 targets.

	2	3	4	5	6	7	8
1	300	47	60	61	300	300	300
2	300	300	72	82	300	72	59
3	300	45	60	86	300	79	99
4	300	46	60	66	66	300	300
5	300	300	300	82	66	300	300
6	300	300	300	73	140	72	65
7	300	300	60	66	140	66	99
8	300	300	300	82	140	72	57
9	45	300	109	73	59	300	57
10	45	47	86	300	59	79	99
11	45	45	300	70	140	300	65
12	300	300	67	86	300	86	99
13	300	300	65	64	300	300	36
14	300	300	59	158	67	64	65
15	45	300	300	59	140	300	68
16	300	48	300	81	300	79	56
17	300	48	109	75	96	300	51
18	300	45	59	300	53	300	45
19	300	47	300	147	140	300	51
20	300	45	300	81	300	300	99

Table A.13: Pessimistic system results for environment 1, with 2 targets.

	2	3	4	5	6	7	8
1	52	300	109	60	96	300	83
2	228	300	300	48	84	300	73
3	62	71	56	300	72	71	83
4	300	300	55	300	46	300	70
5	300	71	94	300	70	300	66
6	96	65	56	72	147	48	61
7	300	300	56	156	127	221	65
8	62	52	56	115	300	300	109
9	300	52	73	58	46	111	62
10	300	300	67	48	54	48	54
11	300	300	300	156	300	61	300
12	300	300	300	300	300	221	63
13	300	300	56	300	57	48	80
14	57	71	49	115	57	300	300
15	96	109	70	115	257	46	51
16	57	55	57	48	300	65	68
17	90	300	72	55	46	79	300
18	96	300	111	55	48	111	70
19	93	52	100	45	72	70	69
20	300	71	58	55	46	300	53

Table A.14: Pessimistic system results for environment 2, with 2 targets.

	2	3	4	5	6	7	8
1	15	35	16	33	20	17	24
2	14	59	24	19	19	23	30
3	300	53	16	129	20	19	17
4	79	108	16	29	30	16	29
5	27	68	26	60	69	78	22
6	23	108	16	53	69	19	17
7	15	100	18	24	24	22	17
8	69	54	59	61	17	20	28
9	15	17	27	21	18	19	72
10	300	62	21	22	66	32	19
11	58	67	14	129	19	19	27
12	15	19	300	17	66	17	31
13	19	19	49	300	28	24	23
14	15	19	300	65	20	23	15
15	20	29	16	65	52	21	25
16	47	28	16	61	60	66	19
17	43	28	47	17	23	21	25
18	15	62	59	16	19	60	25
19	300	19	20	32	59	24	17
20	15	15	16	92	55	20	25

Table A.15: Pessimistic system results for environment 3, with 2 targets.

	2	3	4	5	6	7	8
1	300	121	101	48	300	62	300
2	87	74	81	109	66	300	75
3	96	101	73	94	300	51	106
4	106	91	115	300	119	41	48
5	109	84	144	50	82	51	71
6	98	93	300	102	82	69	54
7	98	95	87	73	82	68	70
8	71	84	300	50	52	51	106
9	98	94	73	50	101	54	37
10	104	104	69	119	96	300	60
11	84	60	300	300	82	73	107
12	106	118	67	64	62	73	52
13	99	94	144	50	73	72	157
14	105	108	300	73	46	300	54
15	95	64	60	98	102	63	47
16	103	101	144	94	76	128	107
17	94	92	73	106	60	87	41
18	85	114	106	103	46	70	66
19	85	89	68	106	72	78	38
20	300	108	108	300	76	80	70

Table A.16: Optimistic system results for environment 1, with 2 targets.

	2	3	4	5	6	7	8
1	300	122	72	79	84	85	128
2	300	51	300	39	96	300	54
3	96	63	107	132	46	45	109
4	300	300	60	39	41	90	71
5	96	300	300	54	46	300	300
6	57	71	300	58	46	87	300
7	300	109	96	72	46	90	73
8	57	58	300	48	300	300	71
9	61	300	57	48	55	90	300
10	95	300	300	60	300	45	71
11	300	82	74	67	60	92	60
12	300	52	56	54	72	59	51
13	96	300	189	300	112	82	300
14	300	109	300	48	76	300	300
15	300	300	56	72	57	49	46
16	300	52	101	300	70	300	49
17	57	300	56	156	44	300	66
18	300	300	72	128	54	70	49
19	57	300	118	300	48	44	83
20	228	300	75	300	46	49	300

Table A.17: Optimistic system results for environment 2, with 2 targets.

	2	3	4	5	6	7	8
1	23	49	47	26	18	68	41
2	51	53	16	42	300	19	27
3	20	45	79	70	21	18	26
4	27	45	52	22	15	18	300
5	16	53	52	24	62	300	30
6	108	300	27	15	19	28	45
7	49	17	59	64	21	36	23
8	116	20	16	20	18	19	36
9	15	16	49	20	16	20	19
10	300	75	49	101	22	17	45
11	16	75	58	20	42	20	29
12	14	19	16	69	65	15	19
13	300	68	16	18	62	43	22
14	44	43	67	15	18	21	19
15	20	100	16	16	18	18	23
16	14	100	64	16	18	25	18
17	15	140	24	22	42	23	18
18	19	19	16	300	18	18	23
19	300	35	24	18	22	18	23
20	63	100	300	15	27	23	25

Table A.18: Optimistic system results for environment 3, with 2 targets.

	2	3	4	5	6	7	8
1	96	113	300	300	98	107	52
2	104	300	104	95	57	80	70
3	300	63	60	66	61	54	54
4	68	105	122	60	300	72	45
5	109	101	116	300	300	74	92
6	300	300	89	300	60	66	112
7	106	101	144	103	300	44	45
8	300	58	300	56	70	52	50
9	104	101	300	66	60	102	40
10	110	69	300	74	52	60	52
11	106	63	73	97	300	75	40
12	106	108	300	67	77	52	40
13	108	102	300	105	69	74	92
14	106	86	72	60	105	55	70
15	64	85	111	121	119	59	52
16	62	94	300	73	78	236	300
17	68	85	69	68	76	60	107
18	85	112	69	66	70	55	50
19	106	69	73	104	60	72	104
20	300	85	69	54	60	76	41

A.3 Simulation 1 Target with Noise

Table A.19: Sharing Potential Field Systems, group size 8. Varying levels of noise.

	Pessimistic			Optimistic		
	low	mid	high	low	mid	high
1	300	91	45	146	83	45
2	57	45	35	45	74	35
3	47	63	44	55	46	44
4	77	45	41	129	50	41
5	56	52	41	222	55	41
6	69	45	45	31	62	45
7	38	34	36	40	32	36
8	78	37	55	40	34	55
9	83	60	55	53	41	55
10	78	66	55	57	163	55
11	43	49	40	60	46	40
12	56	44	35	42	45	35
13	60	47	55	48	47	55
14	70	47	32	50	54	32
15	59	54	45	40	32	45
16	51	37	41	47	45	41
17	78	47	49	41	61	49
18	48	52	55	60	46	55
19	139	30	47	119	47	47
20	40	35	44	57	50	44

Table A.20: Sharing Potential Field Systems, group size 16. Varying levels of noise.

	Pessimistic			Optimistic		
	low	mid	high	low	mid	high
1	31	39	47	72	53	35
2	29	39	46	46	36	38
3	63	49	53	47	360	43
4	41	50	51	34	44	49
5	37	78	56	42	36	45
6	37	55	61	43	53	45
7	50	31	58	50	20	42
8	54	39	62	48	53	36
9	20	39	59	46	20	34
10	42	34	38	38	53	51
11	67	40	16	89	36	43
12	49	46	42	37	50	44
13	36	88	39	69	44	34
14	56	39	42	57	69	43
15	45	40	44	25	20	50
16	19	24	56	20	44	44
17	53	59	34	19	34	45
18	45	71	29	37	50	40
19	33	46	52	29	40	42
20	70	52	43	73	57	49

A.4 Laboratory 1 Target

Table A.21: Non-sharing system results for environment 1 (cluttered).

	2	3	4	5	6	7	8
1	188	105	149	300	300	216	230
2	300	300	189	145	300	290	167
3	300	300	202	300	247	148	300
4	300	126	158	162	300	142	195
5	262	300	197	300	186	152	89
6	300	300	203	300	300	300	98
7	300	60	92	249	94	233	145
8	187	300	233	292	300	218	104
9	300	173	205	300	96	103	117
10	300	208	300	75	168	300	296
11	300	106	92	300	121	166	141
12	300	202	218	160	300	300	225
13	124	149	132	129	102	253	278
14	300	202	300	269	217	300	238
15	300	300	219	300	148	300	97
16	117	300	300	56	91	300	106
17	300	253	237	213	300	300	168
18	300	300	300	300	216	101	108
19	300	127	300	276	300	300	157
20	242	300	300	230	300	300	189

Table A.22: Non-sharing system results for environment 2 (normal).

	2	3	4	5	6	7	8
1	74	131	300	197	300	278	300
2	300	300	293	300	179	172	300
3	300	22	85	300	91	300	265
4	214	300	122	250	105	93	140
5	82	42	300	300	76	90	107
6	300	138	133	209	300	300	300
7	89	129	176	300	70	73	48
8	58	137	57	139	159	74	224
9	115	300	300	258	110	96	233
10	300	180	59	175	95	100	56
11	74	123	191	207	300	177	128
12	300	131	300	218	300	300	62
13	300	89	63	91	300	110	101
14	300	300	300	212	266	300	173
15	82	300	53	266	258	300	125
16	300	76	133	300	100	85	108
17	300	90	84	137	100	287	105
18	300	300	99	72	145	142	300
19	300	300	108	300	119	113	205
20	300	300	45	156	245	300	68

Table A.23: Non-sharing system results for environment 3 (sparse).

	2	3	4	5	6	7	8
1	300	62	300	232	136	48	159
2	154	48	300	300	63	75	26
3	300	102	300	130	127	191	69
4	300	35	109	300	180	112	96
5	218	75	20	61	34	82	51
6	21	263	104	88	300	57	194
7	67	300	76	84	300	52	43
8	300	300	300	53	44	55	54
9	18	300	300	153	57	170	102
10	300	300	150	270	77	110	155
11	117	298	20	25	36	43	300
12	184	68	138	300	60	52	29
13	300	38	55	79	139	32	98
14	57	156	45	90	25	44	69
15	92	300	72	300	49	52	161
16	300	161	259	160	27	300	216
17	300	300	71	102	206	69	44
18	159	37	108	42	119	58	55
19	246	43	46	250	30	33	62
20	54	45	105	78	226	63	59

Table A.24: Pessimistic system results for environment 1 (cluttered).

	2	3	4	5	6	7	8
1	300	93	300	229	104	300	185
2	300	300	147	72	85	195	79
3	300	177	139	182	300	236	106
4	300	300	255	220	300	198	81
5	300	300	176	160	136	300	116
6	300	296	167	151	300	300	238
7	300	300	94	133	300	167	110
8	300	300	95	195	94	211	233
9	300	264	173	70	216	151	300
10	256	300	112	221	192	254	300
11	130	95	126	300	179	300	89
12	131	137	131	71	212	300	96
13	300	201	251	145	204	82	266
14	288	86	193	300	284	198	115
15	300	246	300	159	159	113	300
16	300	106	173	104	159	123	103
17	169	176	167	288	157	80	83
18	300	300	160	237	82	300	183
19	300	300	236	99	106	254	150
20	300	300	162	156	176	195	211

Table A.25: Pessimistic system results for environment 2 (normal).

	2	3	4	5	6	7	8
1	74	96	106	126	74	230	131
2	78	68	140	95	74	184	104
3	293	300	300	84	150	177	94
4	300	71	300	81	127	151	256
5	300	80	144	97	81	241	300
6	300	67	144	123	184	300	300
7	54	162	240	198	290	300	80
8	95	114	84	31	255	174	62
9	68	95	143	88	212	90	98
10	226	105	73	111	117	253	300
11	133	139	205	168	65	122	188
12	300	164	86	60	84	300	187
13	300	216	87	121	117	130	82
14	59	300	64	300	300	300	93
15	103	156	69	77	271	71	137
16	119	113	180	67	144	94	104
17	191	277	109	79	204	52	101
18	127	300	180	90	173	300	43
19	300	167	55	190	300	125	113
20	230	245	94	122	65	185	300

Table A.26: Pessimistic system results for environment 3 (sparse).

	2	3	4	5	6	7	8
1	300	30	30	41	111	26	35
2	40	107	18	44	72	29	59
3	300	300	181	54	300	63	91
4	300	192	28	300	188	69	35
5	300	50	135	267	198	81	75
6	300	300	300	49	63	28	97
7	300	75	300	120	70	143	29
8	46	300	42	194	289	147	106
9	300	300	300	45	252	98	81
10	300	30	99	160	40	256	162
11	47	71	153	234	72	89	28
12	54	300	300	37	86	125	70
13	29	184	186	164	23	248	138
14	300	86	97	46	42	73	55
15	300	132	160	27	79	57	300
16	290	216	300	300	95	173	20
17	140	97	220	300	34	156	124
18	300	300	212	111	47	21	184
19	242	300	73	86	39	49	294
20	148	178	300	298	42	112	19

Table A.27: Optimistic system results for environment 1 (cluttered).

	2	3	4	5	6	7	8
1	300	300	300	217	182	134	72
2	300	300	214	88	300	222	192
3	300	300	243	88	192	90	94
4	300	300	217	220	87	208	300
5	290	206	300	181	300	228	270
6	211	158	241	100	164	300	167
7	300	300	144	250	136	300	82
8	300	300	300	198	190	273	292
9	300	150	300	300	214	133	204
10	219	300	115	283	272	94	112
11	300	110	300	210	273	300	99
12	300	300	220	242	159	300	223
13	300	247	182	157	168	300	178
14	300	300	208	237	158	300	120
15	300	300	103	300	102	111	189
16	300	236	300	148	160	174	82
17	300	300	243	118	300	180	149
18	300	149	96	294	226	300	182
19	300	207	230	145	139	300	77
20	300	100	179	165	173	259	300

Table A.28: Optimistic system results for environment 2 (normal).

	2	3	4	5	6	7	8
1	300	171	50	99	181	76	119
2	300	31	63	36	213	300	133
3	246	203	227	300	300	240	235
4	54	300	59	300	192	139	63
5	300	274	96	59	196	83	300
6	300	167	85	300	117	300	85
7	300	295	32	281	81	99	300
8	300	163	266	300	300	95	300
9	300	300	89	143	300	80	127
10	300	300	42	300	40	110	209
11	89	168	218	238	122	106	196
12	49	46	264	154	80	216	84
13	300	174	300	300	110	300	160
14	52	300	94	77	300	187	89
15	88	163	300	115	76	228	168
16	55	291	32	217	300	300	92
17	300	300	300	143	76	196	300
18	300	300	59	109	68	250	197
19	91	166	46	112	86	150	300
20	300	129	67	95	210	300	106

Table A.29: Optimistic system results for environment 3 (sparse).

	2	3	4	5	6	7	8
1	62	300	300	44	59	131	88
2	300	300	158	57	74	86	125
3	94	278	163	41	75	32	41
4	300	300	84	79	189	84	52
5	300	27	114	65	56	43	300
6	239	48	32	51	133	23	111
7	300	60	123	67	29	235	94
8	139	115	20	32	36	151	60
9	190	69	194	116	69	89	31
10	300	109	22	33	46	16	77
11	99	163	179	55	16	225	18
12	300	46	54	41	57	58	61
13	19	300	44	226	27	96	81
14	300	138	256	184	24	246	43
15	300	246	48	74	66	35	49
16	42	116	62	95	34	124	28
17	267	16	300	65	77	300	300
18	216	300	65	157	92	92	84
19	72	300	86	163	73	222	81
20	300	30	36	300	68	106	300

A.5 Hybrid System 1 Target

Table A.30: Hybrid system results for environment 1 (cluttered).

	2	3	4	5	6	7	8
1	300	300	300	300	193	236	219
2	300	300	181	300	190	300	300
3	300	154	300	240	95	300	129
4	300	300	246	300	300	286	140
5	295	300	207	300	158	164	170
6	300	271	95	217	156	156	131
7	99	300	111	220	182	181	300
8	113	300	300	131	187	234	300
9	300	300	170	144	300	203	300
10	300	300	300	118	300	300	300
11	208	300	300	300	133	154	226
12	300	300	235	300	300	300	184
13	300	169	300	300	225	300	291
14	300	300	300	300	137	300	300
15	219	300	300	300	286	300	129
16	158	130	300	300	105	300	300
17	300	158	300	298	300	202	157
18	300	300	300	115	208	300	170
19	300	275	122	297	138	152	300
20	300	300	300	300	300	263	300

Table A.31: Hybrid system results for environment 2 (normal).

	2	3	4	5	6	7	8
1	300	147	150	300	117	300	59
2	275	271	161	300	300	90	266
3	300	300	300	300	135	300	127
4	110	300	100	175	96	56	204
5	289	300	122	200	242	300	235
6	129	118	104	138	187	77	132
7	148	300	111	137	300	129	153
8	117	110	157	83	178	179	83
9	300	300	262	95	254	75	130
10	300	300	300	141	300	143	300
11	167	300	229	300	144	80	300
12	126	94	140	248	300	163	204
13	88	128	120	188	269	300	103
14	300	70	110	140	300	104	120
15	300	69	121	244	167	110	300
16	174	300	195	300	136	117	102
17	300	300	95	183	125	93	148
18	192	300	300	300	240	139	55
19	117	300	100	114	300	123	93
20	183	300	201	125	300	222	113

Table A.32: Hybrid system results for environment 3 (sparse).

	2	3	4	5	6	7	8
1	51	61	82	222	300	90	86
2	300	76	126	300	184	92	125
3	300	215	300	300	121	120	65
4	188	87	116	98	67	54	95
5	300	89	77	300	58	52	66
6	78	300	205	138	179	112	72
7	86	139	120	43	100	91	195
8	57	98	176	208	41	157	133
9	65	98	56	300	215	300	67
10	120	94	75	125	75	88	161
11	300	260	101	300	133	66	233
12	72	91	93	300	140	79	206
13	103	300	153	77	99	104	113
14	43	111	140	54	113	89	77
15	64	143	123	83	300	120	165
16	300	102	87	136	66	118	127
17	94	157	104	253	55	203	71
18	68	81	94	79	87	85	124
19	300	300	43	128	117	122	87
20	244	47	300	92	80	172	89