PERFORMANCE ANALYSIS OF A RANDOM SEARCH ALGORITHM FOR DISTRIBUTED AUTONOMOUS MOBILE ROBOTS

CHENG CHEE KONG
(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

Firstly, I would like to express my heartfelt appreciation to my project supervisor, Associate Professor Gerard Leng. This project will never be realised without his commendable guidance. Since my Bachelor's dissertation project until this Master's dissertation project, he has always been there to guide me. Particularly in this Master's dissertation project, he has provided me with tremendous assistance in finding my project focus. Despite his busy schedules, he has relentlessly met me at least once a week to obtain updates on the project progress and to ensure that I am progressing in the right direction. He is praiseworthy for his patience in listening to the problems faced during the course of the project and providing valuable suggestions to solve them. Also, not forgetting him for all the birthday parties he has initiated for the postgraduate students, and the many lunches that he has treated us.

Next, I would like to thank my fellow peers, Mr Low Yee Leong and Mr Ng Wee Kiat for their remarkable support and help in making this project a success. Throughout the course of this project, they have provided me with a lot of valuable recommendations and insights when formulating the search algorithm and building the robots.

I would also like to express my earnest gratitude to Mr Cheng Kok Seng, Amy, Mr Ahmad and Pricilla from the Dynamics and Vibration Laboratory, for the generous help that they have rendered.

In addition, I extend my gratitude to DSO National Laboratories for sponsoring part of the project and my DSO colleagues, Mr New Ai Peng and Mr Yeo Ye Chuan for

sharing their views with me. I further thanked Mr Yeo for lending me the workstations to run my simulations.

My family and friends have played important roles in my studies. Their encouragement, concern and support are more than meaningful and heartfelt.

Table of Contents

Summary

Part of the work documented in this dissertation is described in [15]. The paper has been presented in the 2004 IEEE International Conference on Intelligent Robots and Systems (IROS) held at Sendai in Japan.

In this project, there are two objectives. The first objective is to formulate an algorithm for multiple mobile robots to cooperatively search for multiple static targets in an unknown structured environment. The environment is unknown to the robots as they have no *a priori* map information on the environment layout. The second objective is to analyse the system performance of the proposed algorithm.

To fulfil the first objective, we formulated a distributed random search algorithm for a team of autonomous, simple robots. The algorithm is based on five simple behavioural rules and each robot has the same rule set. The algorithm does not need the robots to have self-localization capabilities. In this way, we do not have to deal with localization problem, which is inherent and difficult to solve in the real world.

The algorithm has been implemented on physical robots. It is implemented as five reactive behaviours on the physical robots. In the physical experiments, we deployed five robots to search for three targets located in different rooms in a 4m by 4m mock-up indoor environment with multiple rooms. Ten physical experimental runs are repeated using the same set-up. The robots were able to find all the targets for all ten runs. The mean time taken was 249 seconds. We also performed experiments varying

the environment layout and showed that our algorithm is robust to changes in environment layout.

In addition to physical experiments, we performed multiple simulation experiments to analyse the system performance. The time taken for all targets to be found is used to measure performance. In the simulation experiments, we varied the number of robots from four to twenty robots. We also changed the robots' starting positions and target positions, and the size of the environment. One hundred runs are repeated for each parameter change. Our experiment results show that increasing the number of robots in the robot team and using robots that are smaller in size improves system performance.

Finally, we formulated a benefit function that takes into account cost considerations to evaluate the benefit of increasing the number of robots. We found that ten robots is the optimal number of robots to search in an environment approximately four times the target sensing range for the type of sensors used.

List of Figures

List of Tables

Chapter 1: Introduction

In this dissertation, we give a detailed account of our work described in [15] and further work following it. The paper has been presented in the 2004 IEEE International Conference on Intelligent Robots and Systems (IROS) held at Sendai in Japan. In the paper, we proposed a distributed random search algorithm for a team of simple autonomous robots to search for targets in an unknown structured environment. The proposed algorithm does not require the robots to have self-localization capabilities and has been demonstrated to be effective on actual hardware. In addition, we extended the work and performed multiple simulation experiments for further analysis on the system performance.

1.1 Background

In the last two decades, there has been much research work in the development of mobile autonomous robotic systems. A key driving force is their potential in reducing the need for human presence in dangerous real world applications, such as toxic waste cleanup, clearing of mine fields [17], planetary exploration [4], search and rescue mission, security, surveillance and reconnaissance [28]. The challenge of these applications is the requirement that the robotic systems work autonomously to achieve the human supplied goals. One approach to designing these autonomous robotic systems is to develop a single robot that is capable of accomplishing particular given goals in a given environment. This idea of a single all-powerful robot has been the traditional approach adopted by the robotics research community. A second approach is to design cooperative multi-robot systems. Such a system consists of multiple

autonomous mobile robots working together as a team to accomplish a certain goal. In recent years, there is an increased research interest in the latter approach. This is because cooperative multi-robot systems offer several advantages over the single robot systems [2] [14]:

- The complexity of the mission requirements may be too complicated for a single robot to accomplish. Hence, problems can be decomposed to smaller tasks and allocated among many robots.
- Many robots can be at different places, do many and perhaps different things at the same time. This inherent parallelism in multi-robot systems can improve overall system performance. Hence, cooperating robots have the potential to accomplish a single task faster than a single robot [26].
- Each entity in the team of robots can be simpler than a more comprehensive single robot. Thus, building multiple simple robots can be cheaper or easier than having a single powerful robot.
- A single robot system is itself potentially a single point of failure. Multiple robots can be more flexible and fault tolerant than a single powerful robot. For a multi-robot team, fellow robots can assist a stuck robot or continue without sacrificing the mission.
- Multiple robots have been shown to localize themselves more efficiently, especially when they have different sensor capabilities [22]. This is due to merging of overlapping information, which can help to compensate sensor uncertainty.

Due to these advantages, cooperative multi-robot systems offer the potential of solving large amount of real world applications. This motivated researchers to design multi-

robot solutions and the amount of research work in this field has grown substantially over the years. For these works, they can be broadly categorized into two groups: deliberative cooperation approach and swarm intelligence approach. In the deliberative cooperation approach, robots in the team work together using an explicit cooperation mechanism. Depending on the system architecture design, the robots may or may not follow a leader. There is usually planning involved and a mechanism to perform effective task allocation among the robots. To do this, the robots need to transmit messages to each other using some explicit communications. This usually places high demand on the communication requirements. Hence, cooperation is usually achieved with robots coordinating with each other following some global plan. Swarm intelligence differs from the former approach in that it uses an indirect type of cooperation. Each robot in the team uses simple local rules to govern their behaviours and acts relatively independent from all other robots. They do not follow a leader or to some global plan. The swarm usually consists of large groups of these simple robots and achieves its objectives through local interactions within the entire group. Swarm intelligence is the emergent collective intelligence from these local interactions of groups of simple autonomous entities.

1.2 Project Objectives

There are two objectives in this project: (1) To design an algorithm for multiple mobile robots to cooperatively search for multiple static targets in an unknown structured environment; (2) To analyse the system performance of the proposed algorithm. The environment is unknown to the robots as they have no *a priori* map information on the layout and locations of the targets.

The problem described above poses the following challenges:

- Firstly, how do we manage the many robots running around in the environment? We need to design effective system architecture to control the multiple robots. This system architecture must be capable of controlling a large number of robots and ensure that they work as a team. In addition, it must also be fault tolerant such that a robot breakdown or attrition will not cause the overall system to fail.
- Secondly, we need to design a cooperative mechanism to perform task allocation among the robots. This mechanism must allocate the tasks effectively to the robots and ensure all robots are being employed to achieve the given system mission. Hence, the mechanism should bring about the performance benefit of employing a multi-robot system over a single robot system.
- Thirdly, the unknown environmental layout is another challenge for multi-robot cooperation, since no *a priori* map information is provided to the robots. The robots will have no information that they can use to distribute the task among themselves. We will need to answer the questions of how do we effectively allocate the tasks or resources to the robots such that the overall system performance improves.
- Fourthly, the structured environment is a complex environment for the robots to autonomously navigate through. Most works on autonomous cooperative multi-robot team dealt with cluttered environment. In this type of environment, disconnected obstacles are usually sparsely scattered in the environment. The obstacles may be arranged in a regular array or randomly spaced out in the environment. See Figure 1-1 for an example of a cluttered environment. When

a robot encounters an obstacle, there is usually more than one motion path the robot could take to navigate around it. On the contrary, a structured environment usually consists of connected linear wall-like obstacles. In order to navigate around an obstacle, the robot has to look for discontinuities or openings in the obstacle. For example, the robot has to go through an opening in order to exit a room. See Figure 1-2. Hence, this makes it more difficult for autonomous robot navigation in a structured environment.

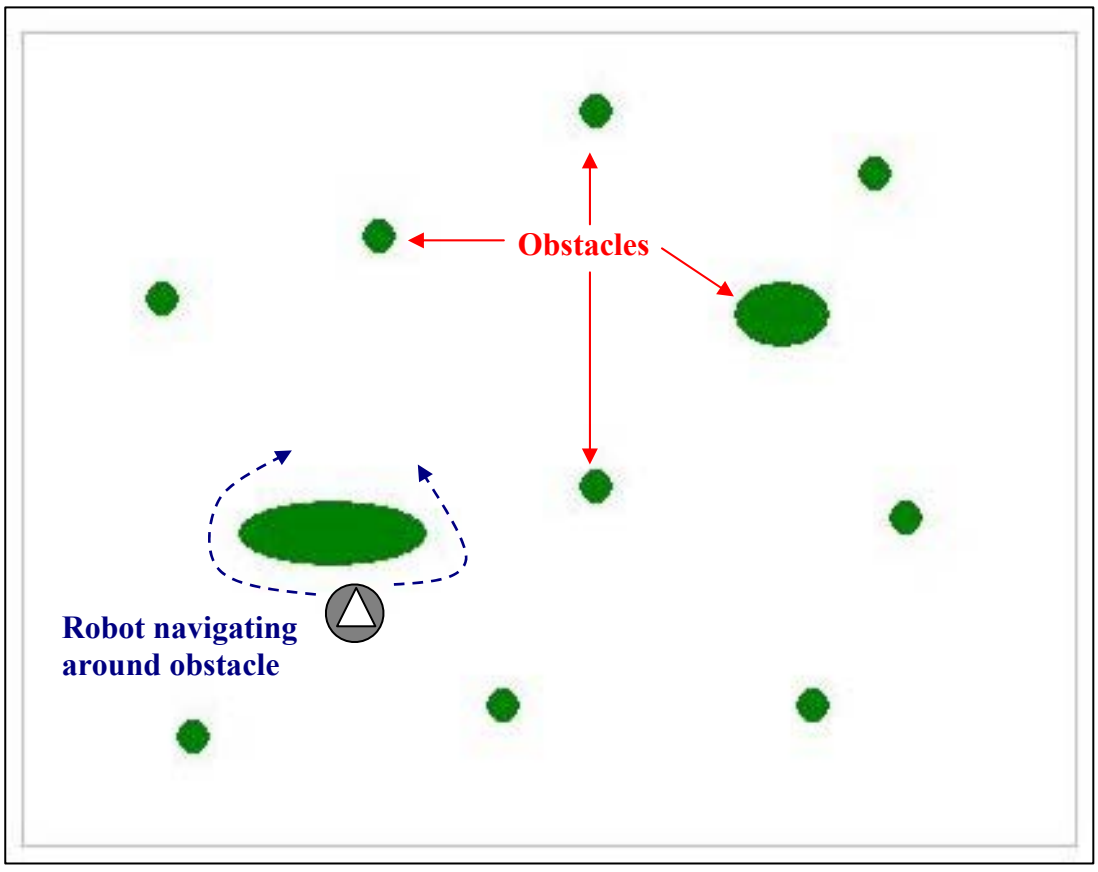


Figure 1-1: An example of a simple cluttered environment

- The fifth challenge is to design an effective search strategy for the multi-robot team. The search strategy should be one that is suitable for multiple robot cooperation. It should also maximize the use of multiple robots such that it

will bring about the benefit of performance improvement over a single robot team.

- Lastly, we need to design a performance measurement to gauge the overall performance of the multi-robot team. Using this performance measurement, we design experiments to analyse the system performance of the proposed algorithm.

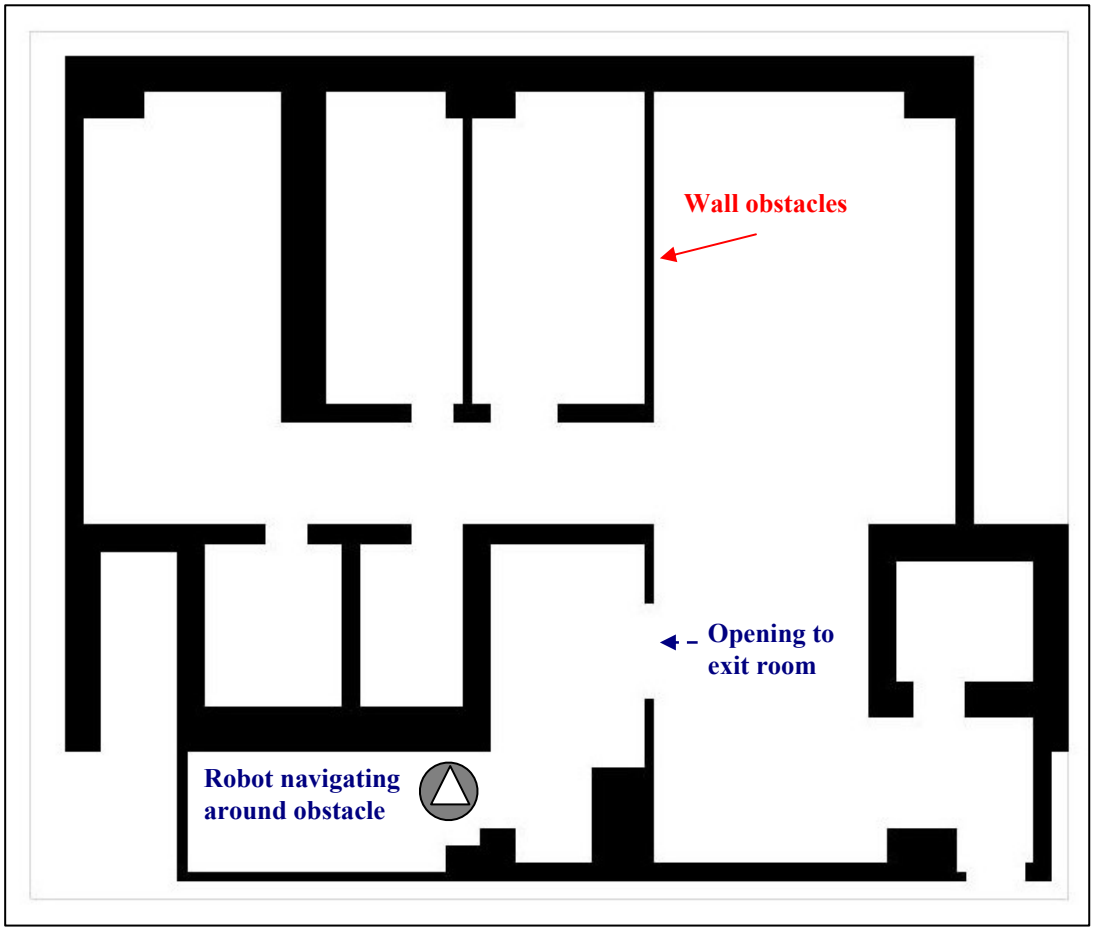


Figure 1-2: An example of a structured environment

In this project, we attempt to solve a search problem using a multi-robot system. Why a search problem? In all the real world problems described earlier, for example a search and rescue mission, security, surveillance and reconnaissance, they all require the robots to perform autonomous navigation and in search for some object of interest

in the environment. Hence, the search problem seems to be the basic problem that all these real world applications have to overcome. Thus, if we can provide an effective multi-robot solution to the search problem, this can lead on to the development of solutions for these real world applications. In addition, the search problem will be an effective test bed for our algorithm on multi-robot control.

1.3 Problem Definition

In this section, we provide a formal definition for the problem described in the project objectives.

1.3.1 Mobile Robot

The design of the control algorithm for the multi-robot team is dependent on the capabilities of the robot platform. For example, the stick pulling experiments described in [36] required at least two robots to coordinate the pulling effort in order to pull out a stick. One robot alone is not able to perform the required task. Hence, it is important to first define the basic capabilities of the mobile robots that the algorithm is intended for. The mobile robots in our multi-robot team are autonomous and independent. They should possess onboard processing capability, motors for mobility, own sensors to provide situation awareness of the environment and other devices that are required for the robot to complete the given task.

1.3.2 Target

In this project, targets are entities of interest in the environment. They emit certain predefined signatures that make them distinct from other entities in the environment. Thus, they can be easily identified and distinguished by our robots equipped with the sensors to detect the emitted signatures.

1.3.3 Search Environment

In this project, the search space is strictly two-dimensional and it is a structured environment. The structured environment is the interior layout of an empty building with multiple rooms. We will simplify the environment by not considering the furniture or other objects that can be found in a building. There are also no doors to block openings from leading into rooms. Thus, the environment is mainly simulated by layout of walls.

In addition, the environment layout is unknown to the robots. This means that no *a priori* map information will be provided to the robots before the start of the mission or throughout the search.

Lastly, the unknown structure environment is bounded. Thus, the robots are constrained to move only within the search space.

1.3.4 Possible Applications

If we are able to design an effective multi-robot system to answer the problem listed earlier, the following far-fetched goals will not be impossible but achievable in the near future:

Search and rescue mission in a disaster sites

Multi-robot systems can be employed to search for survivors in collapsed building. The robots can be fitted with sensors to detect survivors or fitted with cameras to assist rescue workers. Deploying such systems has several advantages. Small mobile robots can replace the rescue workers going into the disaster site. This reduces the risk rescue workers have to bear in performing the rescue mission. Smaller robots can also enter tight situations where a human cannot easily move through. Having multiple robots to search for survivors can potentially reduce the search time needed. This is especially important, as it is a time critical mission. The number of survivors depends on how fast they can be rescued. In fact, using robots in this area is not new. In the recent 2001 September 11 disaster, tele-operated robots are brought into the world trade center site to search for survivors.

Search and clearing of hazardous substances

In view of the growing threat of terror attacks on civilian infrastructures, we can envisage the following scenario. Terrorists planted explosion or toxic chemicals in a shopping mall. We need to find these hazardous entities as soon as possible. Using multiple robots, we can reduce the risk that a human has to undertake. In addition, the robots can be equipped with devices to dispose such items. They can also find these items that may be easily hidden in locations out of reach by a human.

Fighting in build-up area (FIBUA)

The military has recently shown interest in this application, as urban warfare will become a common battlefield in the future. FIBUA is a difficult military operation due to factors such as limited visibility, complex and extensive fortifications, limited intelligence and problems in command and control. These often result in collateral casualties and damages. Because of this, the military has always tried to avoid fighting in an urban environment when possible. The use of multiple robots before the actual operations can provide useful intelligence. They can subsequently serve as surveillance posts to monitor changes in the environment. The robots can also serve to extend the reach of the soldiers during operations, by serving as front scouts and clearing dangerous obstacles obstructing the mission.

1.4 Outline

The work described in this dissertation can be in general grouped into three phases: design, implementation, and analysis. In the design phase, we designed the multi-robot control architecture, the search strategy and the physical robot platform for implementation. For the implementation, we formulated the algorithm into control behaviours in both a sensor-based simulation and the physical robots. Lastly, we performed a series of physical and simulation experiments to study the performance of our random search algorithm.

The contents of this dissertation are outlined as below:

Chapter 2 presents related works that other researchers have contributed in this area. We looked into different approaches for multi-robot architecture, autonomous control of the robot, effect of communications on cooperation and different search strategies.

Chapter 3 presents our random search algorithm. We discuss the requirements of the multi-robot system architecture and look at possible solutions for the architecture design. Then, we present the design of our random search algorithm.

Chapter 4 presents the physical robot that we implemented with the algorithm. We present a detailed description on the design of the physical robot. Besides the physical robots, we also developed a client program for controlling the robot.

Chapter 5 presents the simulation program that we have developed. We describe how we modelled the physical robot and other entities in the simulation program.

Chapter 6 covers our algorithm implementation on physical robots and simulation. We formulated the reactive behaviours to implement the proposed search algorithm.

Chapter 7 describes the physical and simulated experiments to test our algorithm. We present results from the various physical and simulated experiments and discuss the results and observations.

Chapter 8 presents our conclusions and recommendations for future work.

Chapter 2: Background on Previous Work

The first project objective is to design an algorithm to control multiple robots searching for static targets in a bounded structured environment unknown to the robots. The algorithm should be capable of controlling large numbers of robots, perform effective resource allocation to the robots and at the same time be robust to failures. In this chapter, we will review some of the related works.

Cao et al. in [14] provides a critical survey of existing works and discusses open problems in cooperative autonomous mobile robotics, emphasizing the various theoretical issues that arise in the study. The term "cooperative" has been used several times in this dissertation. However, "what is cooperative?" Some explicit definitions in robotics literature include:

- "Joint collaborative behavior that is directed toward some goal in which there is a common interest or reward" in [7].
- "A form of interaction, usually based on communication" in [38].
- "Joining together for doing something that creates a progressive result such as increasing performance or saving time" in [48].

From these definitions, Cao et al. in [14] derived a more formal definition. The authors defined cooperative as: "Given some task specified by a designer, a multiple-robot system displays cooperative behaviour if, due to some underlying mechanism (i.e. the "mechanism of cooperation"), there is an increase in the total utility of the system". In their study, the authors identified five major research axes for cooperative multi-robot systems: (1) Group architecture; (2) Resource conflict; (3) The origin of cooperation; (4) Learning; and (5) Geometric Problems. In addition, the authors

pointed out some promising directions in this field: (1) Development of rigorous formalizations; (2) Formal metrics for cooperation and system performance; (3) Experimental studies might become more rigorous and thorough; and (4) Incorporation of recent ideas in distributed control to achieve oblivious cooperation, or cooperation without communications (e.g. when robots have minimal sensing and communication capabilities).

2.1 Approaches to Multi-Robot Control

Controlling multi-robot systems is a complex problem. Simply increasing the number of robots assigned to a particular task does not necessarily guarantee better performance over single robot systems. Multiple robots must cooperate without destructive interference to produce the benefits over single robot systems. In addition, other issues such as the dynamic environment, malfunctioning robots, imperfect communications, and time and resource constraints add complexity to the problem. Over the years, various control strategies have been proposed. In general, they can be classified in the following three approaches: (1) Centralised Deliberative Approach; (2) Distributed Reactive Approach; and (3) Hybrid Deliberative Approach.

In centralised deliberative approach, there is a central, powerful planner or controller. This central planner gathers information from other robots in the team and forms the global map information of the environment. It then formulates a global plan and allocates various tasks to the each individual robot in the team. While the robots execute the tasks, it monitors the execution, re-plan and re-allocate tasks when necessary. Sometimes *a priori* map information of the environment is required by the

planner to begin. Simmons et al. in [52] described a tiered architecture with a central planner and executive to control multiple autonomous mobile robots. The authors have tested the system in the deployment of teams of robots using different deployment strategies. Li et al. in [34] proposed a centralised planner that uses the hierarchical sphere tree structure to group robots dynamically and perform motion planning for the robots. Burgard et al. in [13] used a centralised planner to coordinate multi-robot exploration. In this work, target points and its utility are assigned to individual robots based on the cost of reaching it. The principal advantage of a central coordinating controller is that an optimal solution can be produced. It can compute a desired position or trajectory for each robot in the system. However, such a system has disadvantages:

- Optimal coordination of the multiple robots is computationally difficult. In addition, the global plan is computed at the central planner. This requires high demands on computation requirements under time constraints on this central planner.
- All relevant information about the robots and their environment are transmitted to a single location for processing. The amount of data transmitted can be enormous and data loss may not be allowed. This leads to stringent and high demands on communication requirements. Rybski et al. in [50] demonstrated how the communication bottleneck reduces the overall system performance. In his work, a multi-robot system on a shared communication channel is shown to perform worse than a single robot.
- The system is not easily scalable in numbers. Adding more robots to the team may require a change in the cooperation strategy. It can also cause an exponential increase in computation and communications requirements.

- The system may not be suitable to operate in a dynamic environment. Any changes to the environment have to be made known to the central planner. It then has to re-plan the global plan. Hence, it can potentially slow down the whole system.
- There is the existence of a single point of failure that can potentially cause the whole system to fail. For example, if the central planner breaks down or there is a break in the communication network, these can cause a standstill in the system. Hence, increasing the risk of mission failure in harsh real world environment.

Distributed reactive approach can address the above problems through distributing the planning among the robots in the team. There is no global plan to coordinate the robots. Each robot is an autonomous independent entity, acting on information that is locally available through its sensors. Cooperation in the team emerges through the local interactions among robots and the environment. As the field of artificial life emerged, researchers have begun to model systems by applying nature-inspired principles such as swarm intelligence to robotics. Swarm intelligence is the emergent collective intelligence from the local interactions of groups of simple autonomous entities. It was first introduced by Beni in [8] on the concept of cellular robotics. Subsequently, proven working models in nature (ants, bees, etc.) have motivated researchers to show considerable interest in swarm intelligence [9][21][56][59][61]. Parunak in [45] summarised several studies of such systems, and derives from them a set of general principles that artificial multi-agent systems can use to support overall system behaviour significantly more complex than the behaviour of individuals agents. Dudek et al. in [21] presented a swarm robot taxonomy of the different ways in which

such swarm robots can be characterised. Reynolds in [49] demonstrated flocking behaviour in birds using just three simple behavioural rules. In his simulated flock, the birds worked independently trying to stick together and avoid collisions. The flocking behaviour emerges from these independent behaviours. Hackwood et al. in [27] proposed a model where simple robots act under the influence of "signpost robots". Many aspects of the collective activities of social insects are self-organized. Successful models of self-organization capabilities of ant colonies have inspired many researchers to design ant-liked systems. Ants and other insects are known to use chemicals called pheromones for various communication and coordination tasks. Payton et al. in [46][47] modelled these chemical pheromones with their virtual pheromones of infrared messages. They have successfully demonstrated this concept in their work on pheromone robotics through physical simple robots interacting with each other using the virtual pheromones. Wagner et al. in [59] had the ant-robots performing distributed covering of an un-mapped building using evaporating traces that gradually vanish with time. Kube et al. in [32] demonstrated cooperative box pushing by a group of robots just using simple ant inspired behavioural rules. Bonabeau et al. in [10] identified that self-organisation relies on four basic ingredients: (1) Positive feedback; (2) Negative feedback; (3) Amplification of fluctuations; and (4) Multiple interactions. This distributed reactive approach allows fast response to dynamic conditions and decrease the communications requirement. Typically, little computation is required since each robot plans and executes its own activities. Moreover, the whole system is more robust and the approach scales easily to accommodate large number of robots. However, the principal drawback of this approach is that they often result in highly sub-optimal solutions because all plans are based on local information. In addition, completeness cannot be assured and generally

large numbers (or infinite time) is the best guarantee to obtain high probability of "completing" the task.

In hybrid deliberative approach, cooperation is deliberately planned for. Unlike the centralized approach, there is no central planner. Information gathered by different robots is exchanged whenever possible and the robots use that available information to generate individual plans. These plans can be individual robot activities or multi-robot activities. Better connectivity among the robots allows better cooperation and hence results in better system efficiency. To achieve cooperation, many groups adopted strategies similar to Contract Net Protocol, first introduced by Smith in [54]. It is an approach to negotiation in multi-agent systems inspired by a market-liked model. Simmons et al. in [53] extended their earlier work of a centralized tiered layered architecture [52] to a hybrid one. Each robot now has a complete three-layered architecture and the layers can interact directly with the same layer of other robots. This approach has the two disadvantages: firstly, negotiation protocols and mapping of task domains to appropriate cost functions can complicate the design of a control-architecture; secondly, negotiation schemes can increase communications requirements.

2.2 Robot Control

Brooks [12] presented a robust and flexible robot control system. Layers of control systems are built to let the robot operate at increasing levels of competence. These layers operate asynchronously and higher-level layers can subsume the roles of lower level layers. Mobile robots designed using the behaviour-based paradigm have shown

good performance in adapting and operating in open environments. The approach has been praised for its robustness and simplicity of construction. One of the pioneering works is Reynolds's flocking behaviour in [49]. Balch et al. in [6] demonstrated multi-robot formation keeping using reactive behaviours. Mataric in [40] presented three examples of behaviour-based control robots performing navigation and path finding, group behaviours, and learning election.

2.3 Communication

For robots to cooperate, some forms of communication may be required. In general, there are three types of communication. In the first type, the environment itself is the communication medium. There is no explicit communication among the robots. Stigmergy is an example of such communication principles where indirect interactions among the entities are through modifications of the environment to achieve collective behaviour. It was first described by Grasse to explain how social insect colonies can collectively produce complex behaviours [10]. The second type is interaction through sensing where the robots are able to distinguish themselves from the environment. Lastly, the robots communicate directly with one another. Hence, robot communication can be implicit through interaction with the environment or explicit where intended messages are directed or broadcast to other robots. Although, Arkin in [1] has demonstrated that cooperation is possible without communication, he does not make the claim for all tasks. The effect of communications on the system performance has been studied in [5][20][37][55]. In general, these works concluded that some simple local interactions among robots would improve the system performance.

2.4 Search Strategy

The problem of exploring an environment has several applications like planetary exploration, reconnaissance, rescue, etc. An effective search algorithm should not be environment dependent [42]. In general, there are two types of search strategies: a perfectly plan-based coordinated search pattern [13][29][42][46][47], and a random search [20][24][25][55].

Burgard et al. in [13] assigns target locations to robots, taking into account the cost of reaching it and its utility. Typically, plan-based strategy requires accurate localization capability. However, in urban environments, accurate localization using Global Positioning System (GPS) is generally not possible. While landmark-based approaches may be inaccurate, this is particularly true in disaster scenarios, where the dynamic environment may undergo structural modifications [29]. Other plan-based approaches in [29][46][47], overcomes this constraint by having the robot entities in a tightly coordinated formation through line-of-sight relationships with one another. However, such approaches may not fully exploit the parallelism advantage in multi-robot systems.

For the random search strategy, Gage in [25] presented the chord strategy by McNish. In the chord strategy, the searcher travels as far as possible between changes of direction and is guaranteed not to visit any point twice during transit. A diffusion reflection algorithm to determine the next chord direction can reliably provide uniform coverage. However, the chord strategy requires the localization of the robot and geometry of the search area. Other randomised search algorithms described in [25] do not claim to provide complete coverage and they have only been explored in

simulations. Gage in [24][25] proposed that multi-robot systems consisting of many inexpensive simple robots may tend to use randomised search strategies for two reasons: (1) the effectiveness of a coordinated search strategy decreases with the capability of the search sensor, and (2) the cost of implementing a coordinated search strategy is higher.

2.5 Related Work

In this dissertation, we proposed a distributed random search algorithm. The multi-robot control architecture of our algorithm uses the distributed reactive approach. In this way, there is less demand on the computational and communication capabilities of the robots. Hence, we can use multiple simple robots to solve the posed problem. Moreover, this approach allows us to scale the number of robots easily and is robust to single point of failure.

In our proposed algorithm, each robot is controlled by simple behavioural rules using the behavioural-based approach. The difference of our work from previous similar works is that we have added behavioural rules to promote local interactions. We believed that these rules add benefits as previous studies on communication have shown that having some form of simple local interactions would improve the system performance. In addition, these local interactions are required for the robot to complete the search problem.

Our proposed search algorithm uses the random search strategy. As discussed earlier, randomised search is more suitable for multi-robot systems that use simple robots.

The analysis on randomised search strategies in earlier works was mostly done in simulation and dealt with cluttered environments. Unlike these works, our random search algorithm is implemented in both physical robots and simulation for a structured environment. In addition, it is robust to changes in the environment.

Lastly, our proposed random search algorithm does not require the robots to localize themselves. As discussed earlier, good accurate robot self-localization in an indoor environment is difficult to achieve on real physical robots. Many works on cooperative multi-robot systems could only be implemented in simulation as they assume that robots have the self-localization capabilities. Hence, we do not make this assumption here.

2.6 Chapter Summary

In this chapter, we have looked at the various multi-robot control architectures that have been proposed by researchers over the years. In general, there are three approaches: (1) Centralised Deliberative Approach, (2) Distributed Reactive Approach, and (3) Hybrid Deliberative Approach. Each of these multi-robot control architectures has its advantages and disadvantages. There is no "the one" architecture that is perfect for all multi-robot systems. However, based on the system requirement, we can apply the techniques from these approaches to design an architecture that brings out the benefits of our multi-robot system.

Some form of communication is required for cooperation among the robots. The type of communication also affects the system architecture. For example, swarm

intelligence uses implicit communications as cooperation and explicit communications is more suitable for deliberate control. In general, the works surveyed suggests that some form of simple interactions will improve system performance.

We also surveyed some of the search techniques employed. In general, there are two approaches: plan-based and random search. Plan-based techniques require more capabilities of the robots, such as self-localization and better sensors, compared to random search strategies.

Lastly, we formulated our random search algorithm using the findings of these earlier works. We also presented the differences of our work from these works. Mainly, our algorithm has behaviour rules to provide local interactions and do not require robot self-localization capabilities.

Chapter 3: Designing the Multi-Robot System Architecture

In the earlier chapter, we have looked at some related works. Over the years, researchers have proposed different multi-robot control architectures and different search strategies to tackle this autonomous robot search problem. From their work, we learned the problems associated with multi-robot control and real world environment implementation complications. In this chapter, we will discuss the requirements of the multi-robot system architecture to solve our posed problem. Following this, we will look at possible solutions for the architecture design. Finally, we will present our random search algorithm for multiple autonomous independent robots to solve the indoor search problem.

3.1 Architecture Requirements

The first objective of this project is to design a cooperative search strategy for multiple autonomous robots searching for targets in an unknown structured environment. The first step to provide a solution for the problem is to design the multi-robot control architecture for the system. Hence, we will define some characteristics that the multi-robot system architecture should possess:

- The multi-robot system should be economically cheaper compared to a single robot system. This is to bring in the added benefit of using multiple robots. Each robot should be relatively cheap and allows them to be sacrificed. For

example, a robot can itself be carrying a bomb and take out a target by exploding against it.

- The system should be fast and responsive. This is important for time crucial tasks such as locating a bomb in the building. The robots cannot spend too much time waiting to compute the next step to move.
- Easily scalable in numbers. The system should allow increasing the number of robots without much work needed to change the multi-robot control system.
- Robust to failures. There should not be any failure points in the multi-robot systems that can potentially cause the whole system to fail. We want a system that is capable of handling robot "attrition" such that the system will still operate even when it is down to a single robot.
- Homogeneous composition. We would like a system that is homogeneous, that is, all the robots are the same, having the same capability. In addition, each robot has the capability of performing a given task alone. This is different from some multi-robot system where robots need to coordinate to perform a task. For example, in the stick pulling experiments, two robots are needed to pull out a stick.

3.2 Inspiration From Nature

In our effort to design an effective multi-robot system, we decided to take a step back and look at nature for ideas. The reason being that nature itself has lots of proven working examples of real life cooperative systems. How does a wolf pack coordinate and organize the pack in a hunt to make the wolves such efficient hunters? How does a flock of geese organize themselves to fly in formation during migration such that

they save energy and have better chance of survival? How does a school of fishes swim together in formation to fool their predators? How does a swarm of army ants that can easily make up to a few hundred thousands in numbers organize themselves in a hunt foraging for food? Living organism in nature has been constantly evolving for the past millions of years and nature has an effective way of improving them. Through nature selection, the better systems will have higher chance of survival and those inferior systems need to improve or face extinction.

Among these social organisms that display cooperative behaviour, the foraging behaviour of the ant colony interests us the most. The ant colony is well known to be efficient searchers, even in terrain that is unknown to the colony. The ants demonstrate this capability in their food foraging behaviour. Not only are they able to find the food source that can be located some distance, but also find the shortest path leading to the food source from the nest. How do these simple social insects achieve such complicated collective behaviour? The answer lies in the ants' capability to self-organize efficiently.

Deneubourg et al. in [19] showed that path selection to a food source in the Argentine ant is based on self-organization. In their simple and elegant experiment set-up, a food source is separated from the nest by a bridge with two equally long branches. After some time, a single dominant trail of ants formed on one of the branches. They replaced the branches with one branch longer than the other and performed the same experiments. Initially, there were two trails of ants on the branches. After some time, the trail on the shorter branch dominated. Hence, the ants were not only capable of finding the food source, but also able to find the shortest path to it.

Aron et al in [3] have shown that the Argentine ant could solve the minimal spanning tree problem. In their laboratory experiment, three or four nests are connected by cardboard bridges. The resultant traffic of ants was such that the ants were travelling on a set of paths connecting all the nests. The set of paths formed a minimal spanning tree, that is, the ants did not use redundant bridges.

Army ants are among the largest and most cohesive societies [18]. Their foraging systems coordinate hundreds of thousands of individuals and cover a thousand square meters in a single day. There is no centralized control, each individual acts on its own behaviours. These swarm raids, comprised of individuals that are virtually blind, are fascinating examples of powerful, totally decentralized control. This is achieved through self-organization, which was shown in Deneuborg et al.'s [18] self-organization model of the army ant raid patterns.

3.3 Proposed Algorithm

We are inspired by the amazing collective foraging behaviour of the ants that results from just simple individual ant behaviours. Hence, we attempt to design our algorithm using a similar approach. Like the individual ants, we design simple behavioural rules for the robots, based on what simple individual will do intuitively when searching in an environment. Firstly, the individual robot needs to wander around the unknown environment to explore it. Through wandering in the environment, the robot will be able to discover new grounds and explore them. When moving in the environment, it will surely encounter obstacles or other robots. Hence, the robot needs certain

collision avoidance logic to prevent collisions. In a search task, the individual robot obviously must have certain logic to find the targets in the environment. Lastly, for the robots to be cooperative and work as a team, they must have some means of communication with each other. Therefore, putting all this together, our algorithm consists of the following five behavioural rules:

1) **Avoid obstacles and fellow robots.**
2) **Find targets and alert neighbouring robots.**
3) **Respond to neighbouring robots' messages.**
4) **Follow external commands.**
5) **Wander in the environment.**

Rule 1 is essentially obstacle avoidance behaviour. The robot will avoid any robots or obstacle in its motion path. This rule does not require the robot to distinguish fellow robots from obstacles.

Rule 2 allows the robot to find any target within its detection range. It will also alert neighbour robots (if any) of the target presence relative to itself. This is achieved through broadcasting a message and any robots within the communication range can receive it.

Rule 3 allows the robots to react to messages from fellow robots. The way the robots cooperate depends on the reaction of the robot. For example, if the robots move away

from the robot emitting the messages, this will result in scattering behaviour of the robots.

Rule 4 allows the robots to receive messages from an external command and acts on them. These commands can be used to control or change the robots behaviour. The robots can follow these commands and perform different sub tasks. For example, initially the robots can be given the order to move in a group and assemble at a certain location, start the mission at a certain time and finally regroup when the mission is over.

Rule 5 is actually the default rule. It is activated when the above four rules is not active and is dependant on the mission requirement. In a search mission, we will like the robots to wander in environment and explore unseen places.

The rules are prioritised, with rule 1 having the highest priority. They provide local interactions among the robots for cooperation. For example, when a robot avoids a fellow robot, it changes its search path. Cooperation to find all targets is achieved through the local interactions triggered by rule 2 and 3. These two rules can ensure that a target is only found by one robot. Each robot is independent and controlled by the five behavioural rules, that is, all the robots have the same intelligence.

3.3.1 Algorithm Characteristics

The proposed algorithm has the following characteristics:

- Distributed Control. Each individual robot works independently and is controlled by the behavioural rules without waiting for instructions from a central controller.
- Simplicity. Each robot is governed by just the five simple behavioural rules. The rules do not have high computational requirements. Hence, the robot can be simple and low cost.
- Fast and responsive. The simple behavioural rules do not demand high computation capability. Thus, all computations can be completed relatively fast and be responsive to changes in the environment.
- Homogeneous. All robots are physically the same and are controlled by the same behavioural rule set.
- Scalable in numbers. The algorithm does not require tight coordination among the robots and the system is homogeneous. Hence, robots can be added or removed easily without the need to change the algorithm.
- Robustness. The system is distributed. Thus, there is no single point of failure.

3.3.2 Uniqueness of Algorithm

The behavioural rules do not require the robot to know its position or the environment layout. This is an important characteristic as robot localization in an indoor environment is a difficult task and is itself an area of research. This is because it is not possible to use the GPS in the indoor environment. We cannot rely on the robot's odometer as it accumulates errors from slippage and uneven terrain. In addition, landmark-based localization techniques have high computation requirements and do not work well in dynamic environments. Dynamic environments, for example a

disaster site, may contain moving entities or changes in the layout. In this work, fellow robots moving in the environment will cause problems for landmark-based localization techniques.

Lastly, the randomised search strategy is employed here. Based on the architecture requirements discussion, the random search is suitable for our algorithm. This is because of the characteristics of our proposed algorithm. Random search is suitable for simple distributed control architecture that is reactive to dynamic changes, and where low cost robots can be easily added or removed from the system.

3.4 Chapter Summary

In this chapter, we proposed a distributed random search algorithm for a team of autonomous simple robots. We have also identified certain key requirements for the algorithm. They are: low cost robots; fast and responsive; scalable in numbers; homogeneous and robust to failures.

In designing our algorithm, we looked at working cooperative systems from nature. In particular, we are interested in the foraging behaviour of the ant colony. Through simple local individual behaviours, the ants can produce emergent complex system behaviour. Thus, we are inspired by the ants and propose an algorithm comprising of five simple behavioural rules. Our algorithm employs a random search strategy and does not require the robots to localize themselves. The behavioural rules also provide local interactions among the robots for cooperation.

Chapter 4: Designing a Physical Robot Platform

In this chapter, we present the mobile robot platform that we designed and built to implement our random search algorithm. We will discuss the design criteria for the robot. Following this, we give a description of the sensors, actuators, communication devices and other components on the robot. Finally, we developed a client program to control the robot using a Pocket PC. It should be noted that the focus of this research is not designing the physical mobile robot. We designed the mobile robot for the purpose of implementing our search algorithm. The physical robot is used to demonstrate that the search algorithm is robust enough to function in a real world environment.

4.1 Mobile Robot Design Criteria

Simplicity is one of the main characteristics of the proposed distributed search algorithm. It does not have high demand on the capability of the robot platform. Hence, simple and economically cheaper robots can be used. Gage [23] defined a simple robot as one possessing: (a) a measure of mobility, (b) sensor capability to measure its position with respect to its nearest neighbouring elements, (c) mission capable sensor, (d) communications capability, and (e) on board processing capability.

Our simple robot must be capable to meet the demands of the proposed algorithm. Thus, we define the following design criteria:

- Mobility. Each robot must be capable to move about in the environment without help from other robots. Since the robots do not need to have localization capability, there is no need for position encoders on the robots.
- Sensor capability. The robots must be equipped with sufficient sensors to provide situation awareness of the environment to the robot. This is required for successful autonomous robot navigation in the environment.
- Mission capable sensor. Since the mission here is to search for targets, which are represented by light beacons. Hence, light detectors are required for the robot to find targets.
- Communication. The algorithm requires some means of communication among the robots. Thus, the robot must be equipped with some communication device.
- Processing capability. The robots are autonomous. Hence, there must be some on-board processing capability in the robot.

4.2 Inspiration From Nature

Since the design of our random search algorithm is nature-inspired, we could also look at nature for ideas to design our mobile robot platform. In nature, organisms are living in a harsh world where the rule "survival of the fittest" applies. Having good sensory and motion capabilities is important for survival. Nature selection and evolution has taken place through millions of years. Hence, the organism in nature must have evolved to some very effective design. For example, biologists discovered that bats started out as ground rodents and have evolved wings for flight to hunt for food.

However, the focus of this research is not on the robotic platform. We will not look into those fanciful actuator capabilities but gather insight on the sensory part.

Have you ever tried swatting a fly? Notice how difficult it is to swat one. The fly's sense is very well developed. Its compound eyes with an "ommatidium" as basic unit cover a wide angle and are particularly good in detecting quick assault movements. This sensory capability combines with the fly's ability to manoeuvre itself into intricate flight patterns makes it difficult to swat a fly. Another example is spiders. Spiders have six or eights eyes, all looking in different directions. This allows them to spot preys, predators or potential danger easily. Hence, allowing them to react responsive to the specific situation.

Drawing ideas from these insects, we decided to design a robot that has sensors to give it wide coverage of the environment around it. Preferably, the robot should have sensors all round to detect for any changes in the environment.

4.3 Robot Platform Description

The focus of this research is not in designing a robot platform. We are not proposing creative novel designs for the robot. The purpose of the robot is for us to implement the proposed algorithm and demonstrate that the algorithm works. We set out to design our simple mobile robot based on the criteria listed and the ideas obtained from nature. We named the mobile robot "CoSyBot". Figure 4-1 shows the CoSyBot. We have built and assembled five CoSyBots to implement our random search algorithm and conduct physical experiments.

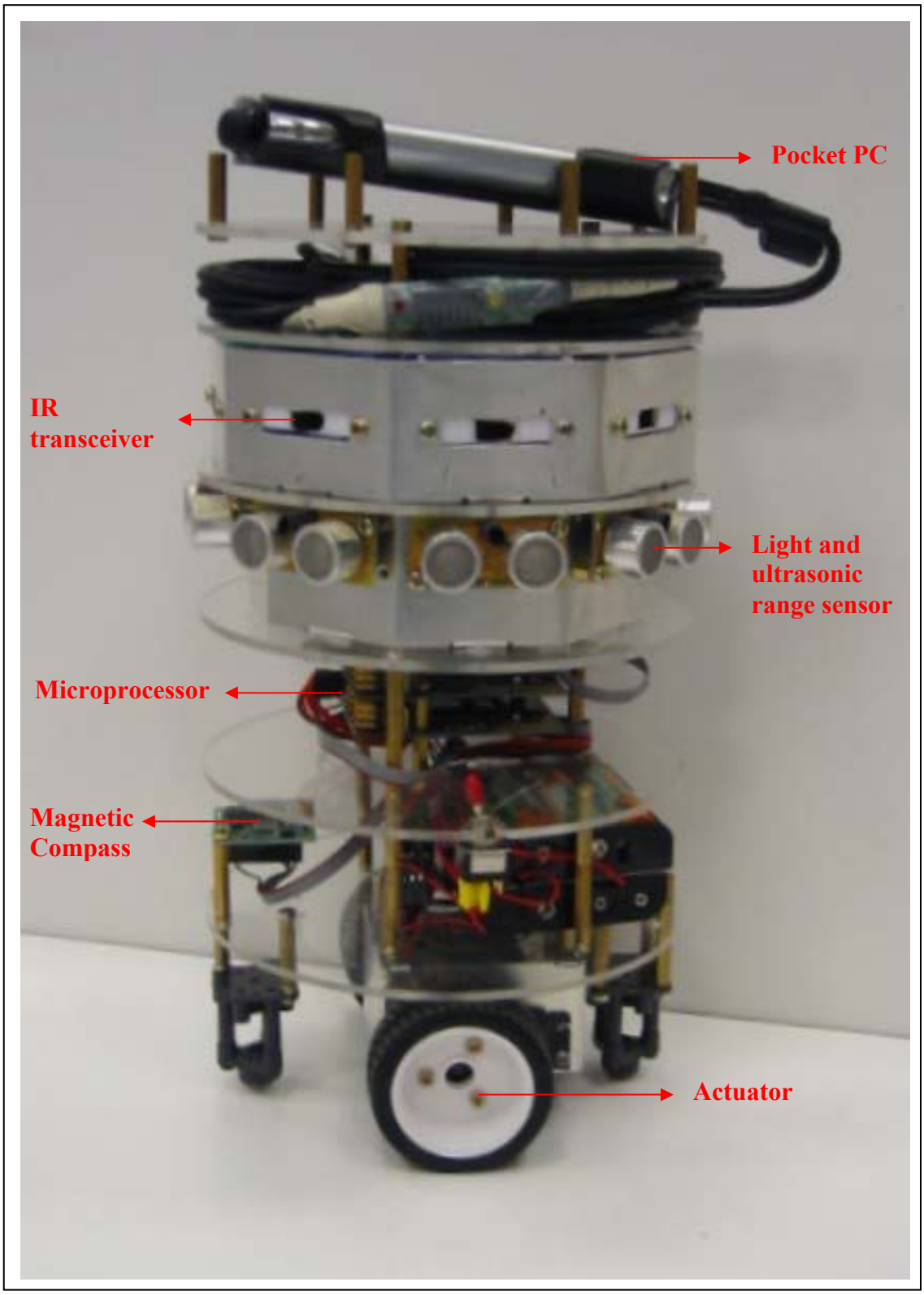


Figure 4-1: CoSyBot robot platform

4.3.1 Features of CoSyBot

4.3.1.1 Physical Structure

CoSyBot has a modular structure. Layers or modules can be changed or added to reconfigure the robot according to the mission requirements. For example, the light sensor module can be replaced with a pyroelectric sensor module to detect heat source. The robot has a circular footprint of 150mm in diameter with a height of 300mm. There are two power supplies for the robot: six 1.5Volts AA batteries to power the sensors and microprocessor; and four 1.5Volts AA batteries for the servomotors. All power is supplied directly to the microprocessor boards, which then relay it to the other devices.

4.3.1.2 Mobility

Locomotion of the robot is provided by two modified continuous servomotors with a wheel each and supported by two ball transfers serving as caster wheels. The wheel axis passes through the centre of the robot with the wheels symmetrical about the centre axis. See Figure 4-2. This allows the robot to turn on the same position using differential drive without any swing radius. Since, the environment is a two-dimensional structured workspace with a flat terrain, this locomotion is sufficient for the robot to manoeuvre in it. The servomotors are driven by the motor controller on the microprocessor board.

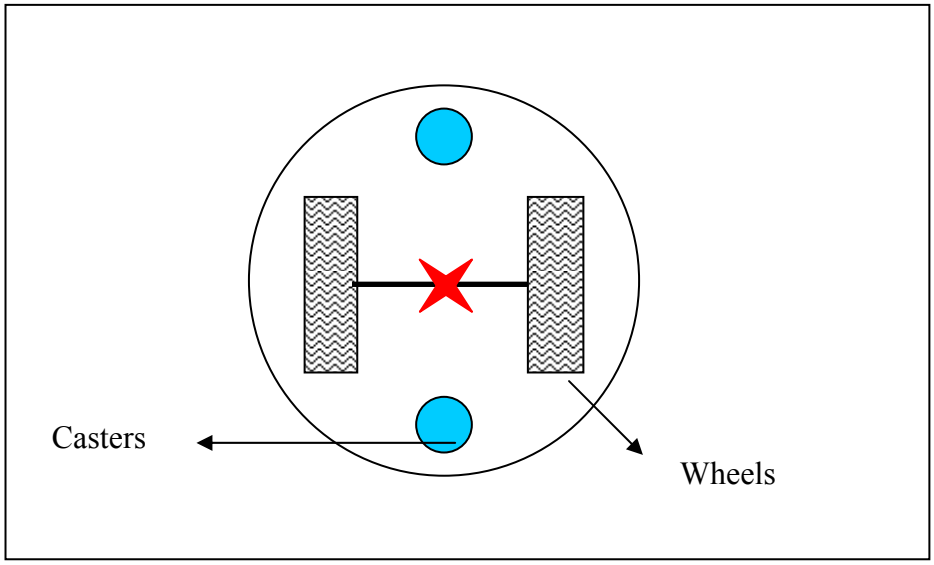


Figure 4-2: CoSyBot actuator layer

4.3.1.3 Sensors

Firstly, a Devantech magnetic compass gives the heading information of the robot in the environment. This compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earth's magnetic field. The output is the absolute heading value in the range of 0-360° with a resolution of 0.1°. The compass output 0° when it is pointing in the direction of the Earth's magnetic north. It is connected to the microprocessor board using the industrial IIC bus.

Secondly, the ultrasonic range sensor layer consists of eight Devantech SRF08 range sensors. Technical data of these ultrasonic sensors can be found in Appendix A. They are placed 45° apart in a circular array, giving 360° all round sensing for the robot. Each SRF08 is an ultrasonic range sensor and provides range information of the environment around the robot. They are connected to the microprocessor board using the industrial IIC bus. Each SRF08 has a unique address, which allows the microprocessor to talk directly to.

Thirdly, each SRF08 has a built in light sensor. The light sensor is to detect the targets (light beacons) in the environment. Hence, they are the mission capable sensors for the robot. Together, the sensors provide situation awareness of the environment.

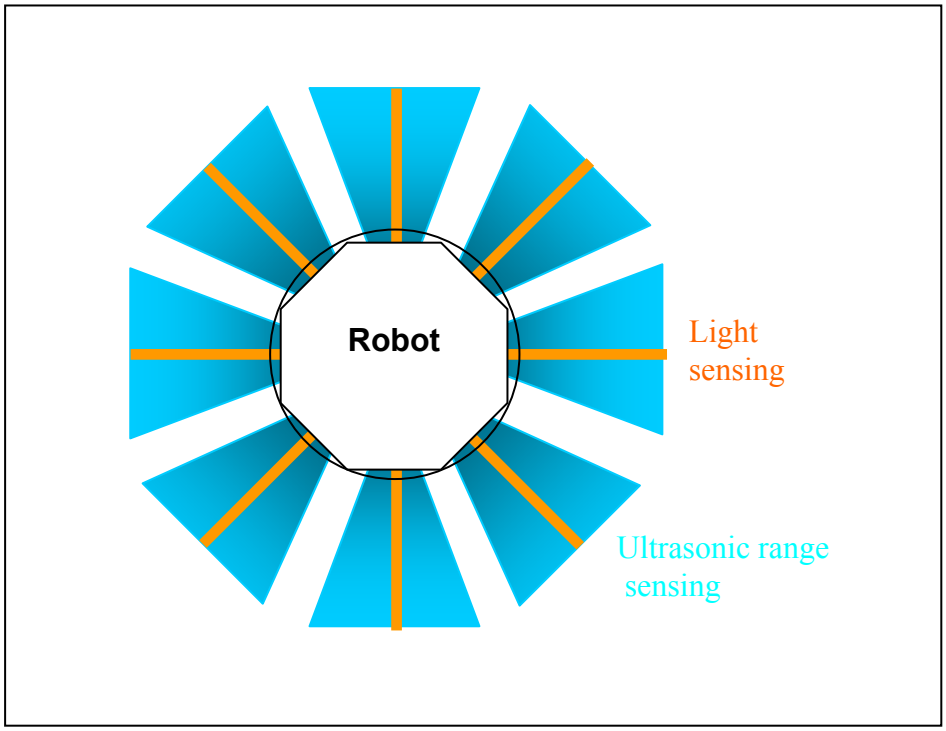


Figure 4-3: SRF08 sensors arrangement

4.3.1.4 Communication

There are two types of communication devices on CoSyBot to provide local implicit communication and global explicit communication. The IR transceiver layer on the robot provides the implicit communication. It consists of eight IR transceivers placed 45° apart in a circular array, similar to the light and ultrasonic range sensor layer. These IR transceivers are standard IrDa 1.0 compliant. They are directional and allow communication via the IR channel. Hence, robots can send IR messages to other robots within the transceiver line of sight range. This layer allows local interaction between neighbour robots, similar to the "dance" which honey bees perform to

communicate with each other. We designed the circuit for these IR transceivers, as it was not available as an additional option for the BrainStem GP 1.0 board. They are fabricated using commercial off the shelf components. The circuit board is responsible for encoding and decoding the IR messages and communicates to the microprocessor board via the RS 232 serial interface.

The wireless LAN communication network provides the explicit communication. This is achieved using the IEEE 802.11b wireless device on the Pocket PC. The developed communication network uses the UDP protocol, and allows both broadcasting and peer-to-peer communication among the robots. In this way, explicit communication can be achieved among the robots in the network to enhance cooperation among them, e.g. when one robot found a target, it can inform all the robots in the team. In addition, an operator can send external commands to the robots via this network.

4.3.1.5 Processing

Firstly, the robot has two BrainStem GP 1.0 microprocessor boards networked together. This is the part of the robot that manages connections to the rest of the physical devices on the robot, i.e. servomotors, ultrasonic range sensors, IR transceiver etc. Simple TEA programs can also be loaded and run from the microprocessor boards. Technical specifications of the BrainStem GP 1.0 microprocessor board can be found in Appendix B.

Secondly, the robot can be operated in slave mode with a host computer as the master. A Pocket PC is mounted on the robot and served as the host computer via RS 232 serial connections. This arrangement allows the robot to exploit the better processing

power of the Pocket PC. In our set-up, we used the HP iPAQ H5450 Pocket PC, which has a 400 MHz XScale processor. The algorithms for the behavioural rules are implemented in the host computer to control the robot. Figure 4-4 illustrates the architecture of CoSyBot.

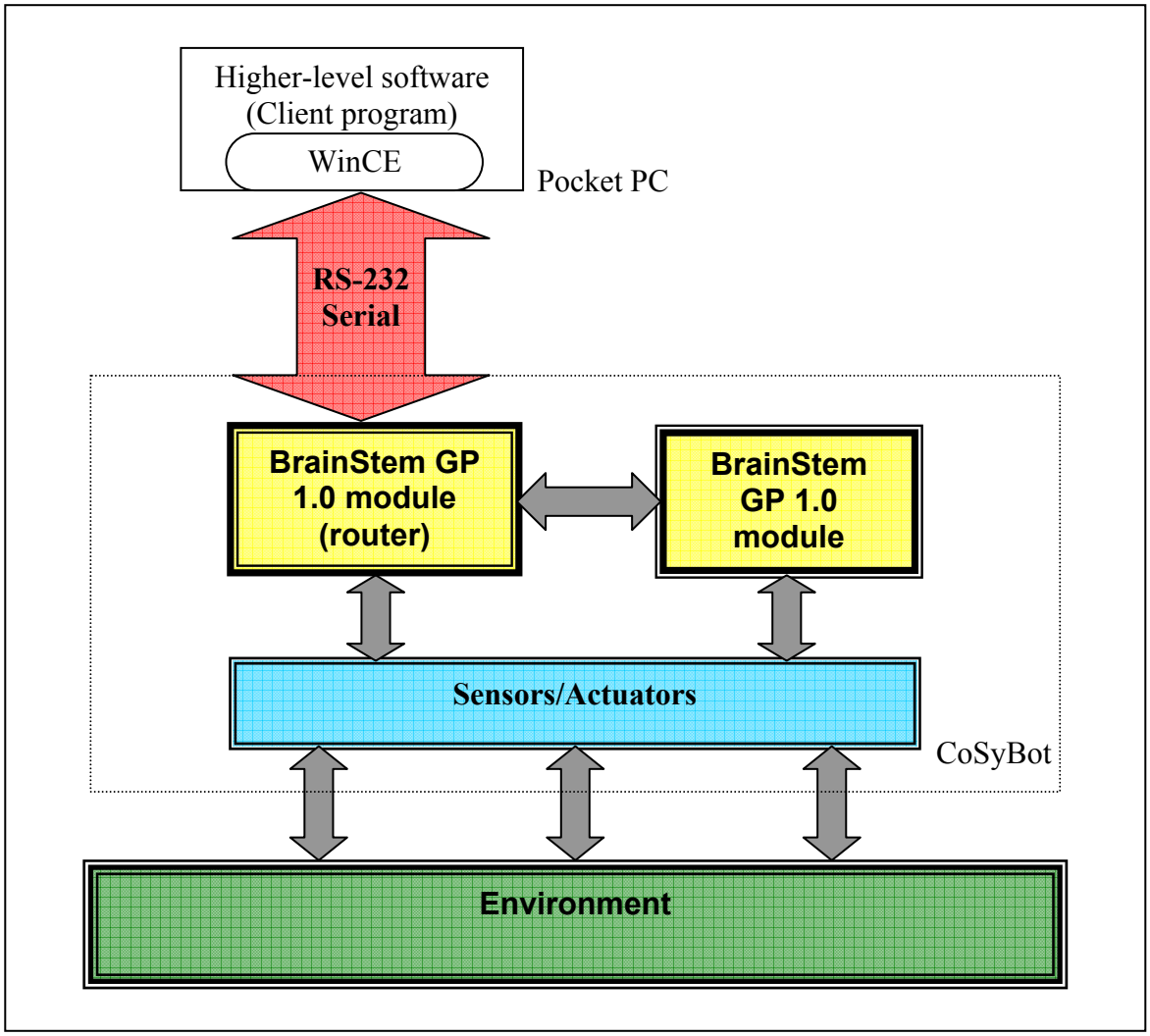


Figure 4-4: Architecture of CoSyBot

4.4 Client Program

We developed a client program for the CoSyBot using Microsoft Embedded Visual C++. The design of this client program promotes reusability and portability. Figure 4-5 illustrates the architecture of client program. It consists of the following 3 layers:

- **Device Abstraction Layer**. This layer abstracts the robot's control code away from the physical robot platform. In other words, it is an interface layer between the control codes and the physical robot. It decouples the control codes from the physical robot. Requests or commands are sent to this layer, which in turn relay them to the physical robot and back to the source. The advantage of having this layer is that it allows the client program to work with other robotic platform. This can be achieved through providing the relevant device controllers to the device abstraction layer. Hence, the user needs only to work in this layer, while reusing his algorithm control codes without major changes to it.
- **Application Layer**. The algorithm control codes reside in this layer. It mainly consists of two components. Perception module accesses the robot's raw sensor data through the device abstraction layer and processes them into useful interpretable information for the robot. Behaviour module accesses this information and triggers the appropriate control behaviour. The control behaviour then sends the actuator commands to the physical robot through the device abstraction layer.
- **User Control Console**. This layer provides an interactive display for the user to control the robot. He can start and stop his control program from here, access the processed information from the Perception module or execute the behaviours in the Behaviour module. It also has an output console for logging data while the robot is in operation. This is handy for the user to debug or troubleshoot his control codes. The user can also access the robot's physical devices directly via the device abstraction layer.

The Microsoft Embedded Visual C++ toolkit also allows the client program source codes to be complied to work with different embedded systems. In our set-up, the client program runs on the HP iPAQ H5450 Pocket PC with a 400MHz xScale processor. The Pocket PC acts as the host computer and controls the robot in slave mode. Connection between the Pocket PC to the robot is via the RS-232 standard serial port.

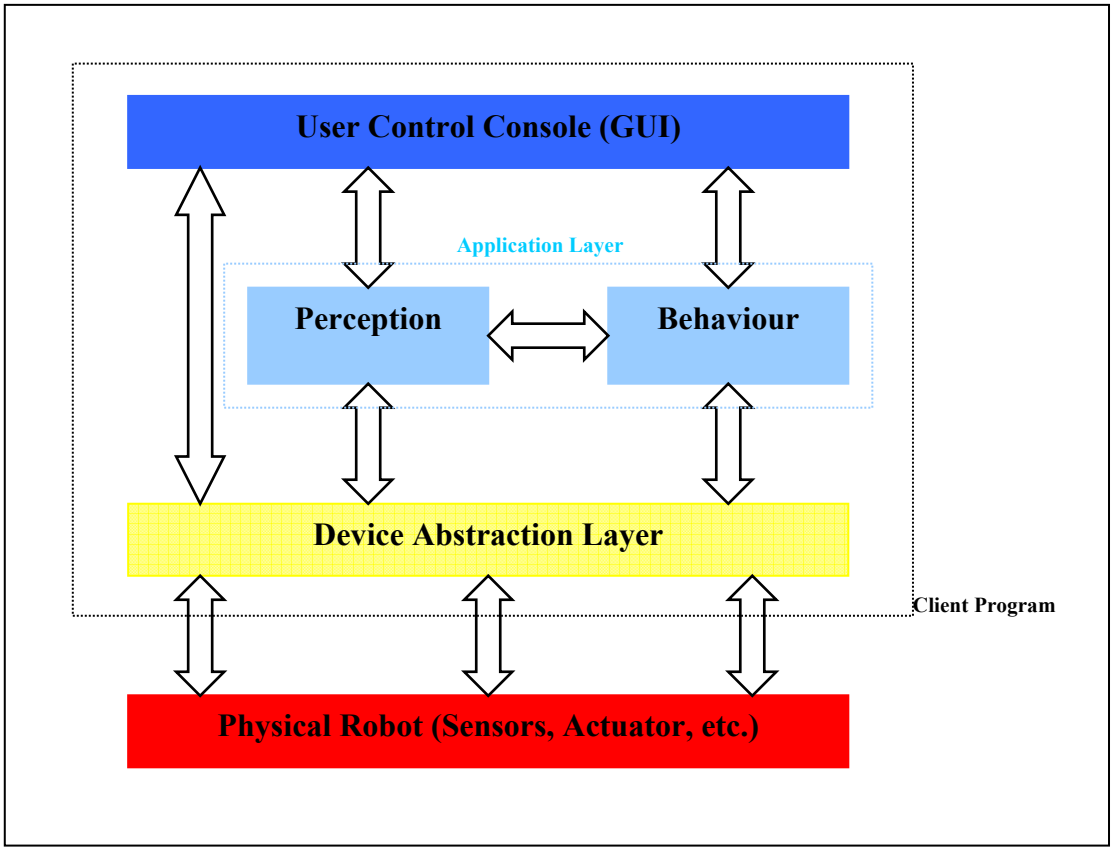


Figure 4-5: Architecture of CoSyBot client program

4.4.1 Features of the Client Program

Features of the client program include:

- Algorithms are implemented onto the robot through the client program using C/C++ programming language. This is useful as C/C++ is a widely used programming language.
- It handles the serial communication link with the robot.
- It supports the wireless network implementation for the robot.
- It has a hardware diagnostics tool to check the robot's sensors and actuators.
- A GUI interface for easy interaction with human operator. See Figure 4-6.

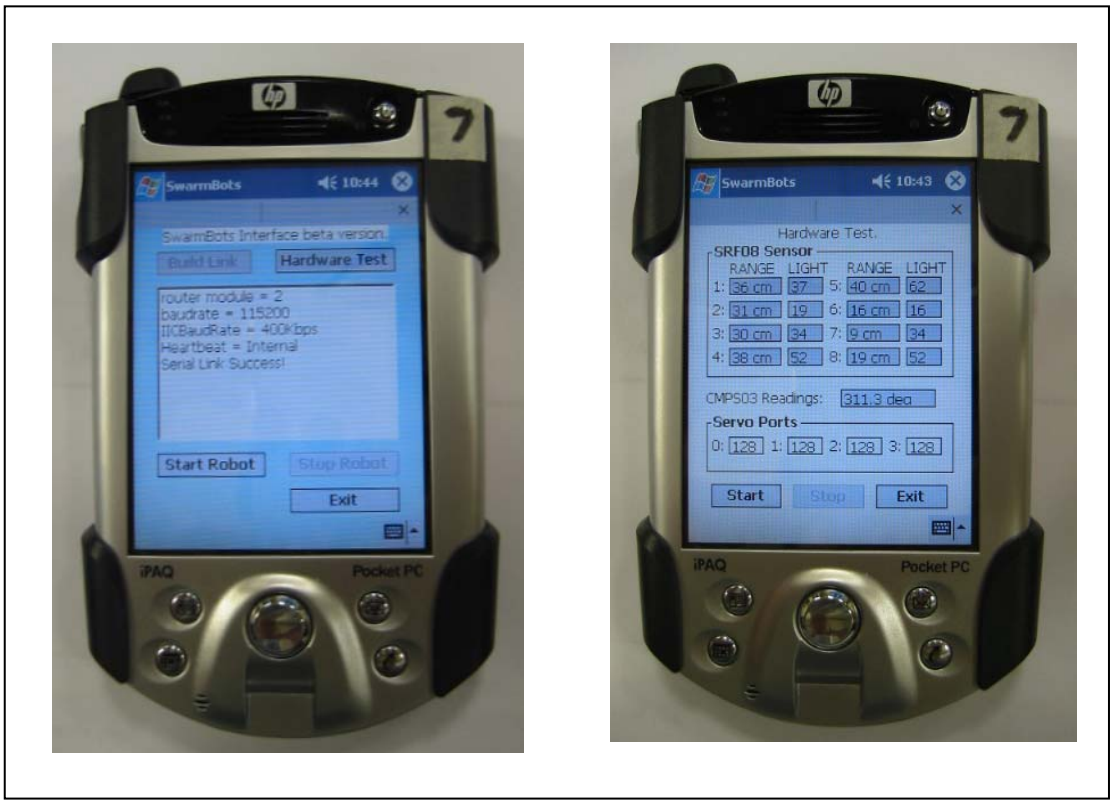


Figure 4-6: GUI of the client program. Main window (left) & Hardware diagnostic window (right)

4.5 Chapter Summary

In this chapter, we presented the physical robot CoSyBot that we designed and built. CoSyBot is designed following the criteria that we had laid out so that it could meet

the demands of our proposed random search algorithm. In general, it must possess sufficient sensors, motors for mobility, means for communication and on-board processing capabilities, allowing it to operate independently. CoSyBot uses ultrasonic sensors to sense the environment, light detector to detect targets (light beacons), IR transceivers for local interactions, and servomotors for mobility. We have built and assembled 5 CoSyBot for our physical experiments. In addition, we also developed the client program to control the robots using a Pocket PC.

Chapter 5: Modelling the Physical Robot and Structured Environment

Earlier, we described the physical robot that we have designed and built to implement our proposed random search algorithm. We also developed the client program to control the robot using a Pocket PC. The intention is to demonstrate our proposed algorithm on physical robots. In addition to the physical robot and client program, we also developed a computer simulation program. We modelled the physical robot and the structured environment in the simulation program. In this chapter, we present the simulation program and how we modelled the individual entities in it.

5.1 CoSyBot Simulation

This simulation program serves 2 main purposes. Firstly, we can use the simulation program to design, develop and test our robot behavioural codes before implementing on the physical robot. Hence, we are not developing it directly on the physical robot. This makes sense because we would have to deal with real world problems if we work on the physical robot directly. This takes more time and makes it difficult to isolate any problems encountered during the development process. Secondly, it is technically not feasible to perform multiple experiments using physical robots for studying the algorithm. We can use the simulation program to perform multiple simulated experiments. The simulation program will allow us to generate more results in a shorter time compared to performing the physical experiments. The above two

purposes can be achieved through developing a simulation program that has high fidelity to the physical robot.

The simulation program is a two-dimensional graphical simulator that simulates the topological view of the environment. It can run on any Microsoft Windows based machine. The program is written in C/C++ programming language and using Windows programming for the graphics. Approximately six thousand seven hundred and seventy lines of codes are written for this simulation program.

5.2 Modelling the CoSyBot

A model of the CoSyBot robot is created in the simulator. We modelled it closely to the actual robot, having similar physical characteristics. These are namely the physical body, motion drive, navigation and target sensors, and communications capabilities. They are described in detail as follows.

5.2.1 Physical Body

The physical CoSyBot robot has a circular footprint of 150mm diameter and stands 300mm high. All physical devices on the robot are bounded within this circular footprint. In other words, there are no physical devices protruding out of the 150mm circle and physical contact on this circular body is considered robot collision. We modelled this physical structure of CoSyBot in the simulator. On the simulator GUI, a circle object represents the CoSyBot robot. The circle size scales with the simulated environment dimensions, accordingly to the actual CoSyBot in a real environment.

The simulated robot will collide with simulated objects on the circle circumference and not pass through them in the simulator.

5.2.2 Motion Drive

The physical CoSyBot robot uses differential drive with zero swing radius for motion. Hence, it is capable of making turns on the same location. We model this motion capability of the robot on the simulator. The model has two parameters for motion: translation velocity and rotation velocity. These parameters can be configured accordingly to the physical robot's speed.

5.2.3 Sensors

There are three sensors on CoSyBot. The first sensor, which is the magnetic compass, gives the robot's heading. This is easily achieved in simulation using an absolute reference frame and output the simulated robot's heading with respect to it.

The next sensor, which is the SRF08 ultrasonic range sensor, gives range information of objects from the robot. Figure 5-1 shows the sonar pattern graph of the SRF08 ultrasonic range sensor provided by the manufacturer. As illustrated by the sonar pattern graph, the sensor has an effective field of view of 40° (20° left and right of the sensor central axis). The sensor has built-in processing capability to process the raw sonar data and outputs the detected range directly. It has an effective range of 3cm to 6m. However, this depends on the height of the sensor mounted from the ground. This is because the sonar wave emitted is approximately conical in shape and sonar reflections from the ground will decrease the effective range. The effect reduces as the

sensor is mounted higher. On the CoSyBot, the sensor has an effective range of 1.2m. We verified these physical characteristics (field of view and range) of the sensor through physical experiments of measuring the range and shifting obstacles away from the sensor central axis. The experiment results can be found in Appendix C. We first created a simulated model of the ultrasonic sensor with these physical characteristics. Then, we positioned eight of this simulated ultrasonic sensor in the CoSyBot model. They are placed in the same position as the physical CoSyBot. Therefore, the range values obtained from them in simulation are similar to the actual physical robot.

The third sensor, which is the light detector, is capable of detecting any light beacons that is within direct line of sight. On CoSyBot, we used the light detector with a binary output to detect the targets. These characteristics are modelled on our simulated light detector to detect line of sight simulated light beacons. Similarly, they are placed at the same positions as on the physical CoSyBot.

5.2.4 Communication

CoSyBot uses IR transceivers for local implicit line of sight communication. The communication range is limited to 1m for our application. We modelled these physical characteristics in our simulated IR transceiver. Eight of this simulated IR transceivers are created in the CoSyBot model. Similarly, they are placed in the same position as the physical CoSyBot.

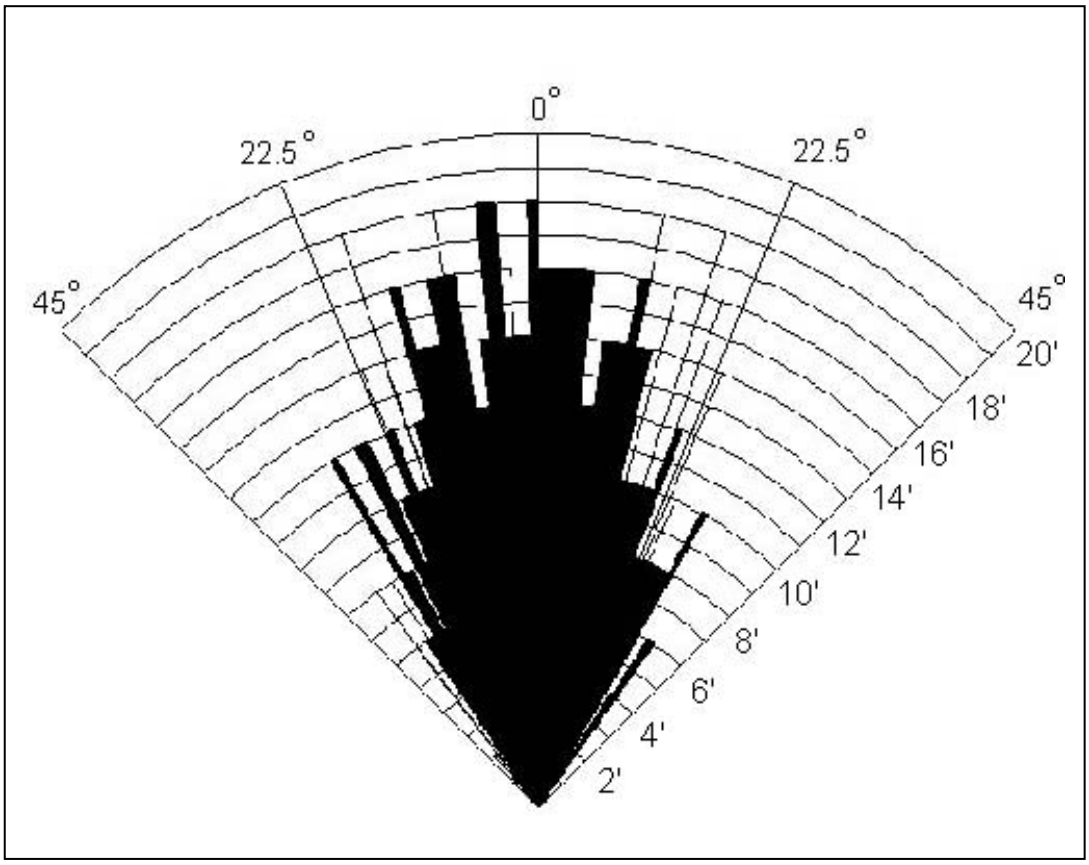


Figure 5-1: SRF08 sonar pattern graph

Therefore, a simulated model that is close to the actual physical robot is created. In addition, the algorithm is developed following the architecture of the client program in Figure 4-5. Program codes for both perception and behaviour are separated from the simulator through a device abstraction layer. The effort in doing this is to ensure that the algorithm developed in simulation can be easily implemented on the physical robot without major changes.

5.3 Modelling Target

Targets are modelled as circular objects in the simulator. They return true to the simulated light detectors on the CoSyBot when they are within the prescribed detectable range.

5.4 Modelling the Structured Environment

Walls and obstacles are modelled as either polygon or circular objects in the simulator. Like CoSyBot, they exist as simulated physical objects in the simulator. Any contact with the surface of these objects is considered collision. Hence, the CoSyBot simulated object cannot move through them. The structured environment is created using a combination of wall objects in the simulator.

5.5 Input File

A text file is used to specify the set-up in the simulator. In the input file, the user can specify the number of robots to use and their positions, the structured environment layout and dimensions, the number of targets and their positions. The simulator reads in these inputs and creates the set-up in the simulator. Hence, different experiment set-ups can be easily modelled using different input files.

Putting all these together, Figure 5-2 shows the GUI display of the simulator. The blue circles with arrowheads represent the CoSyBots and their heading. The black polygon objects represent the walls or obstacles. Yellow circle objects represent the targets and they change to red circles when found.

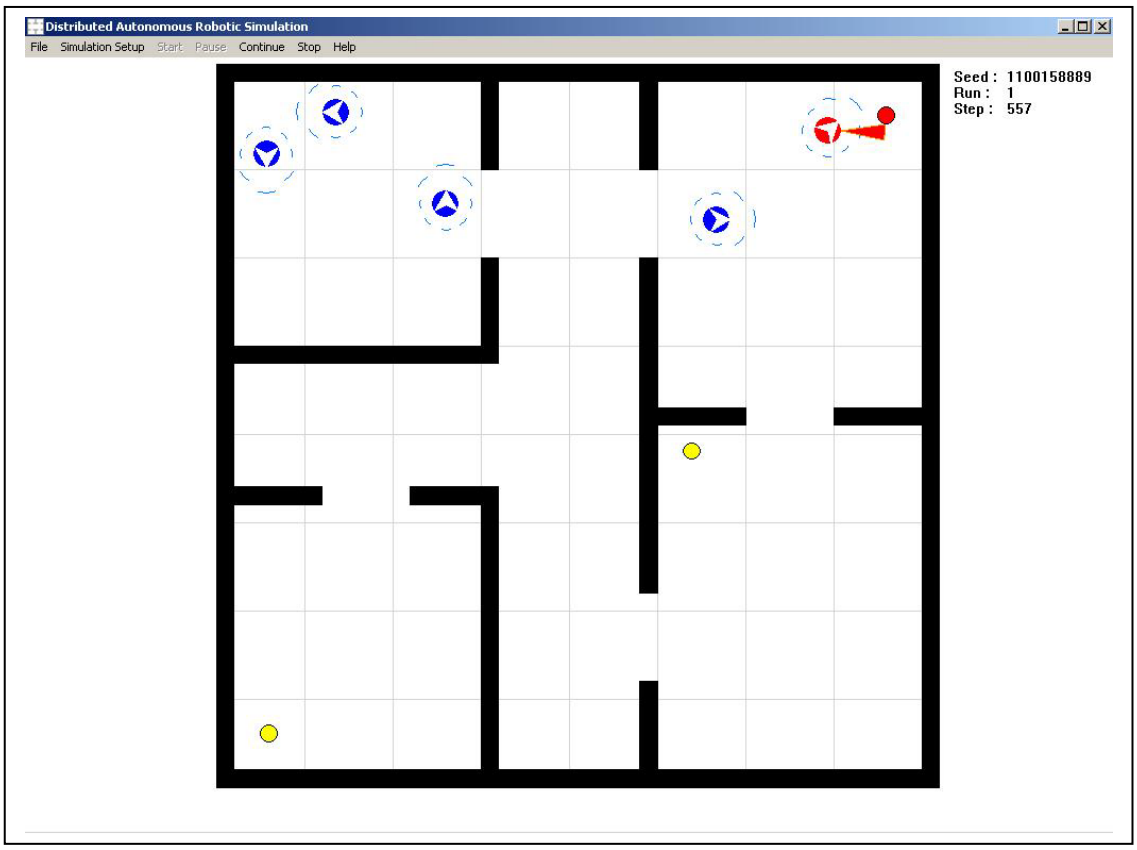


Figure 5-2: Simulator GUI

5.6 Chapter Summary

In this chapter, we described the two-dimensional simulation program that we have developed. We modelled the CoSyBot, targets (light beacons), and the structured environment in the simulator. It reads an input text file to set-up the simulated environment. The program is written in C/C++ programming language and runs on any Microsoft Windows based machine. Approximately six thousand seven hundred and seventy lines of codes are written for this simulation program.

Chapter 6: Algorithm Implementation

In this chapter, we present the implementation of our proposed random search algorithm. We formulated the five behavioural rules into five reactive behaviours for the CoSyBot. These reactive behaviours are developed using the simulator described earlier. Then, they are tested and refined on the physical CoSyBots. We describe the algorithm behind each individual behaviour and illustrate how they interact together to solve the posed search problem.

6.1 Mobile Robot Navigation

Autonomous mobile robot navigation is a key problem to successful applications of mobile robot systems. In addition, avoiding collision with other entities in the environment is important for successful mobile robot navigation. Hence, all mobile robots feature some form of collision avoidance. These range from primitive algorithms that stop the mobile robot in short of a detected obstacle to complex algorithms that enable the robot to detour obstacles. The latter approach may result in non-optimal paths, since no prior knowledge about the environment is used. This brings no added benefit of designing complex obstacle avoidance algorithms as they usually have high demands on sensors and computation requirements. Hence, our algorithm is a simple local obstacle avoidance behaviour that suffices in preventing collision and selecting a safe direction for the robot to navigate in the environment.

The second factor to consider is the navigation sensor used by the robot. Sensors such as a laser range finder that has long range and high accuracy for resolution of 0.5

degree can provide more detailed and highly reliable sensor information of the environment. They are excellent for algorithms that are highly sensitive to sensor accuracy. The CoSyBot uses eight SRF08 ultrasonic sensors for navigation. Ultrasonic sensors present many shortcomings [11]: 1) Poor directionality limits the accuracy in determining the spatial position of the obstacle; 2) Frequent mis-readings are caused by either ultrasonic noise from external sources or stray reflections from neighbouring sensors; and 3) Specular reflections can cause an obstacle to be not detected or "seen" as much smaller that in reality. Hence, we need to design an obstacle avoidance algorithm suitable for using ultrasonic sensors.

From the literature, there are a number of obstacle avoidance algorithms available. One popular obstacle avoidance method is based on edge detection. In this method, an algorithm tries to determine the position of the vertical edges of the obstacle and then steer the robot around either one of the "visible" edges [16][33][60]. A common drawback of edge-detection approaches is their sensitivity to sensor accuracy.

Khatib in [30] suggested the idea of imaginary forces acting on a robot. In this method, obstacles exert repulsive forces, while target applies an attractive force to the robot. A resultant force vector, comprising the sum of a target-directed attractive force and repulsive forces is calculated for a given robot position. Further works using this technique can be found in [31][41]. Common to these methods is the assumption of a known and prescribed world model, in which simple, predefined geometric shapes represent obstacles and robot's path is generated off-line.

Borenstein et al. in [11] developed the Vector Field Histogram (VFH) method. It looks for gaps in locally constructed polar histograms. VFH employs a two-stage data reduction process. In the first stage, it constructs a reduced one-dimensional polar histogram from a local grid around the robot. In the second stage, it selects the most suitable sector from all polar histogram sectors with a low polar obstacle density and aligning the robot to that direction. Using this technique, the robot is able to travel at faster speeds without becoming unstable and is less likely to get trapped in a local minima. Borenstein et al. demonstrated the VFH method on a mobile robot using ultrasonic sensors. Ulrich et al. proposed the VFH+ method in [57]. VFH+ is an improved version of the VFH. It takes into account of the width of the robot and the robot trajectory. This results in less trajectory oscillations and also an improved direction selection using a cost function. The VFH* algorithm proposed in [58] combined VFH+ with the A* search algorithm to overcome problematic situations inherent with purely local obstacle avoidance algorithms.

The results and potential of the VFH algorithm inspired us in the algorithm design of our reactive behaviours for "CoSyBot". We find the similarity of the "CoSyBot" physical characteristics and the VFH technique. In particular, the sector selection and aligning the robot to that direction is suitable for CoSyBot.

6.2 Reactive Behaviours

In chapter three, our proposed algorithm consists of five behavioural rules: 1) Avoid obstacle and fellow robots; 2) Find targets and alert neighbouring robots; 3) Response to neighbouring robots' messages; 4) Follow external commands; and 5) Wander in the

environment. CoSyBot has no self-localization capability, which itself poses a major research problem. Hence, it will not be able to map the explored environment. Most global plan-based algorithms will not be possible without map knowledge of the environment. Local reactive approach is simple and fast. It connects the appropriate actions for the robot to take directly to the available sensor information. See Figure 6-1. This is suitable for dynamic unknown environment. In addition, this approach has less demand on the physical capability of the robot platform. Thus, it is suitable for the simple CoSyBot platform.

The development of the robot behaviours is done using the CoSyBot simulator. This is useful as the simulator contains a realistic model of CoSyBot. Both the simulator and the client program are written in C/C++ programming language, and have similar architecture. Therefore, the behaviour codes developed in the simulator could be directly ported to the client program on the physical robot for testing. A local reactive behaviour is implemented for each of the five behavioural rules.

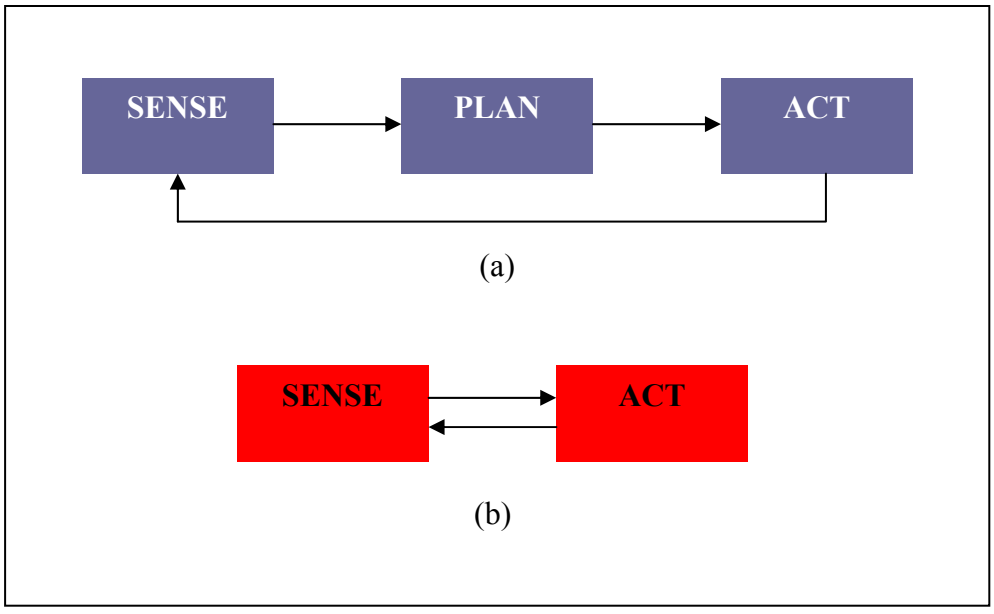


Figure 6-1: (a) Plan-based approach versus (b) Local reactive approach

6.2.1 Obstacle Avoidance

Obstacle avoidance is the first behaviour on the list. As discussed earlier, obstacle avoidance is important for autonomous mobile robot navigation. It is responsible for preventing the robot from physical collision with other entities in the environment and at the same time determines the robot's new direction of motion. Therefore, it has the highest priority.

This behaviour relies on the eight SRF08 ultrasonic range sensors to provide situation awareness of the environment. The robot's perception of the environment local to the robot is divided into eight sectors, each covered by a sensor respectively. CoSyBot always moves forward in a direction coincident with the central axis of sector 0. Hence, sectors 1 and 7 are forward facing, sectors 2 and 6 are side facing, and the rest are rear facing sectors. See Figure 6-2. Each ultrasonic range sensor simply returns the range reading of the nearest object. Sectors with range reading less than a prescribed trigger distance will be considered blocked. A blocked sector to the robot implies that there is an obstacle in that particular sector or region of the environment. Hence, it is not "safe" for the robot to transverse into that region. The region is now considered inaccessible. When the obstacle avoidance behaviour is triggered, the result is a change in the direction of motion or collision with an obstacle, if the choice of the trigger distance is not properly chosen.

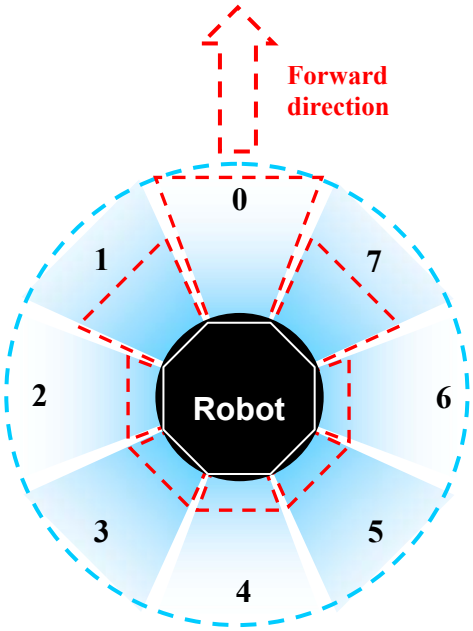
A larger trigger distance would suggest more "intelligent" obstacle avoidance behaviour, as the robot is able to start avoiding obstacles that are some far distance away. However, for local obstacle avoidance algorithms, a larger trigger distance may

cause the robot to be overly sensitive and perform manoeuvres unnecessarily early. Moreover, it may also lead the robot to no longer detect existing openings or falsely report a trap situation. This is due to the field of view of the ultrasonic sensor, which increases away from the sensor. However, too small a trigger distance may result in the robot not sufficiently responsive to dynamic changes in the environment. For example, having a second robot suddenly moved into the robot's motion path. Hence, the choice of the trigger distances is important. A set of trigger distances that performed well is obtained through experiments using the physical CoSyBot; refer to table in Figure 6-2. The dashed red lines in the figure represent these trigger distances. Observe that they are not uniform. There are two reasons for this. Firstly, for the CoSyBot set-up, uniform range will cause the robot to be trapped in continuously turning situation or overturning if it is to ignore the rear sectors, see Figure 6-3. Secondly, the forward facing sectors is assigned a larger trigger distance for higher safety considerations. The trigger distances are obtained through positioning obstacles in the respective sectors with the robot moving towards them and able to avoid them safely without collision. The largest trigger distance is 15cm in sector 0, which coincidently is the robot width.

The robot changes its direction of motion when a forward facing or side facing sector is blocked. The robot stops first and proceeds to select a new direction. During the selection stage, the trigger distances for all sectors are changed to 15cm. Each central axis of an unblocked sector is a possible new candidate direction. The robot will then choose the unblocked sector closest to sector 0, illustrated in Figure 6-4. Hence, sectors 1 and 7 have higher priority over the other sectors. In symmetrical situations, the robot will choose one randomly. If all the sectors are blocked, the robot will stop

and wait for one to be clear. This is because the robot should always be able to turn back to the previous direction, unless a fellow robot has moved behind it. Hence, it can only proceed only after this fellow robot moves away. This is a first in, last out policy. In this way, our algorithm can address the issue of multiple robots in a dead-end narrow passageway. This is important for multi-robot systems since it will be an inherent problem. The algorithm is illustrated in Figure 6-5.

Similar to the VFH algorithm, our local obstacle avoidance also looks for gaps in the local environment. This is achieved without the need of maintaining a local grid and constructing the polar histogram. Thus, our technique is less computational intensive.



Sector	Safety Range / cm
0	15
1	11
2	7
3	7
4	7
5	7
6	7
7	11

Figure 6-2: Sector representation of the local environment around robot

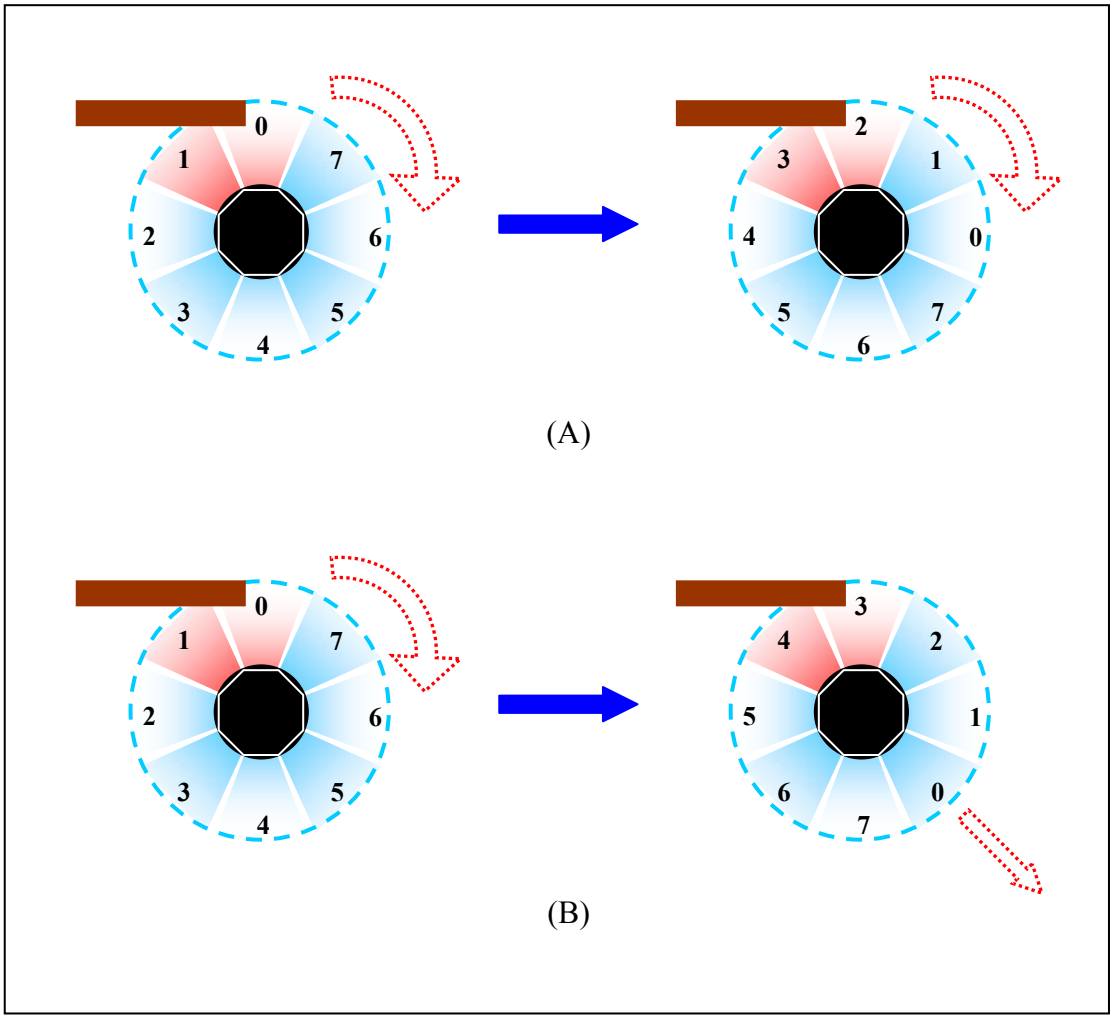


Figure 6-3: Uniform ultrasonic range. (A) Continuously turning, (B) Overturning

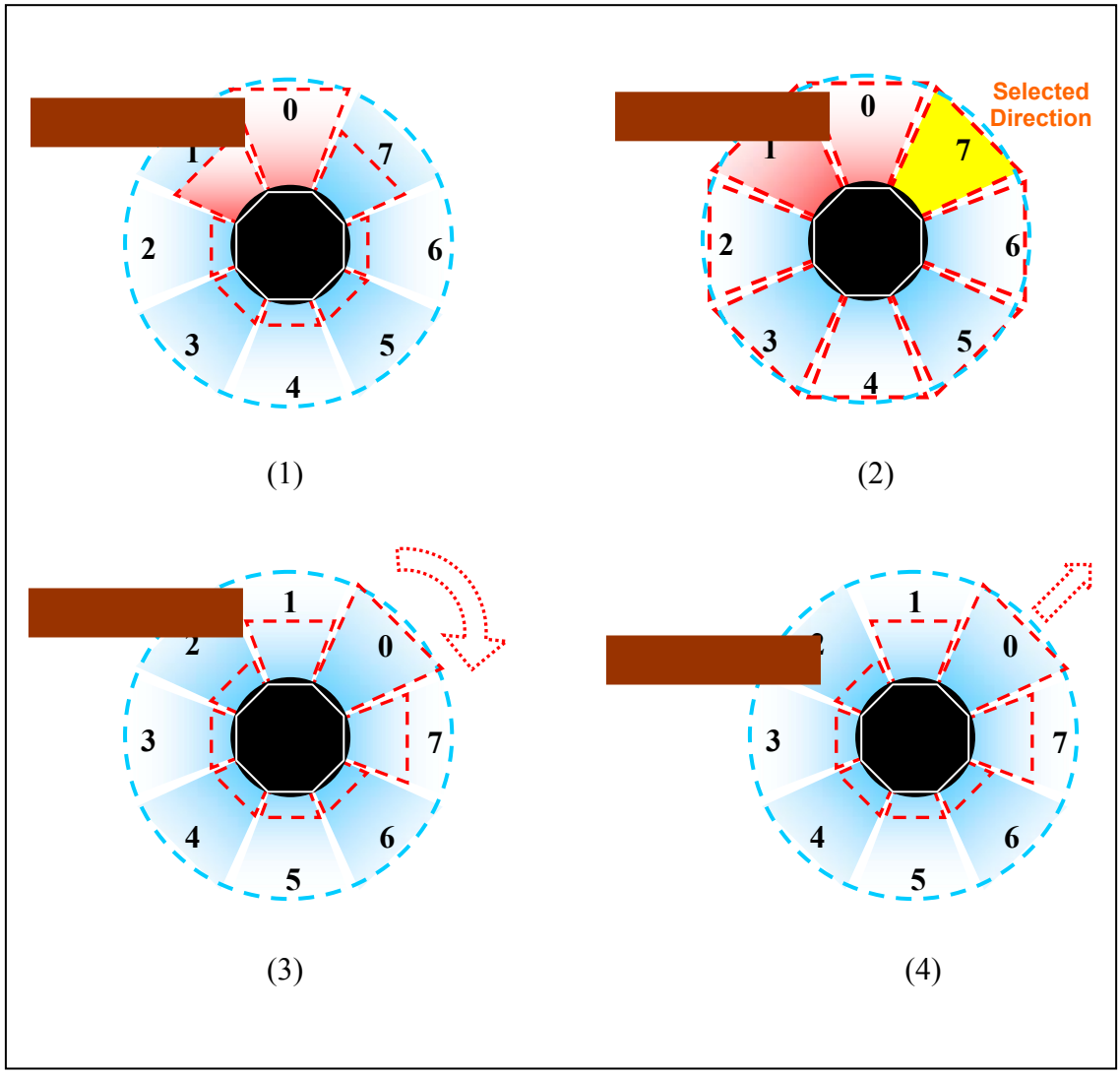


Figure 6-4: Illustration of obstacle avoidance behaviour

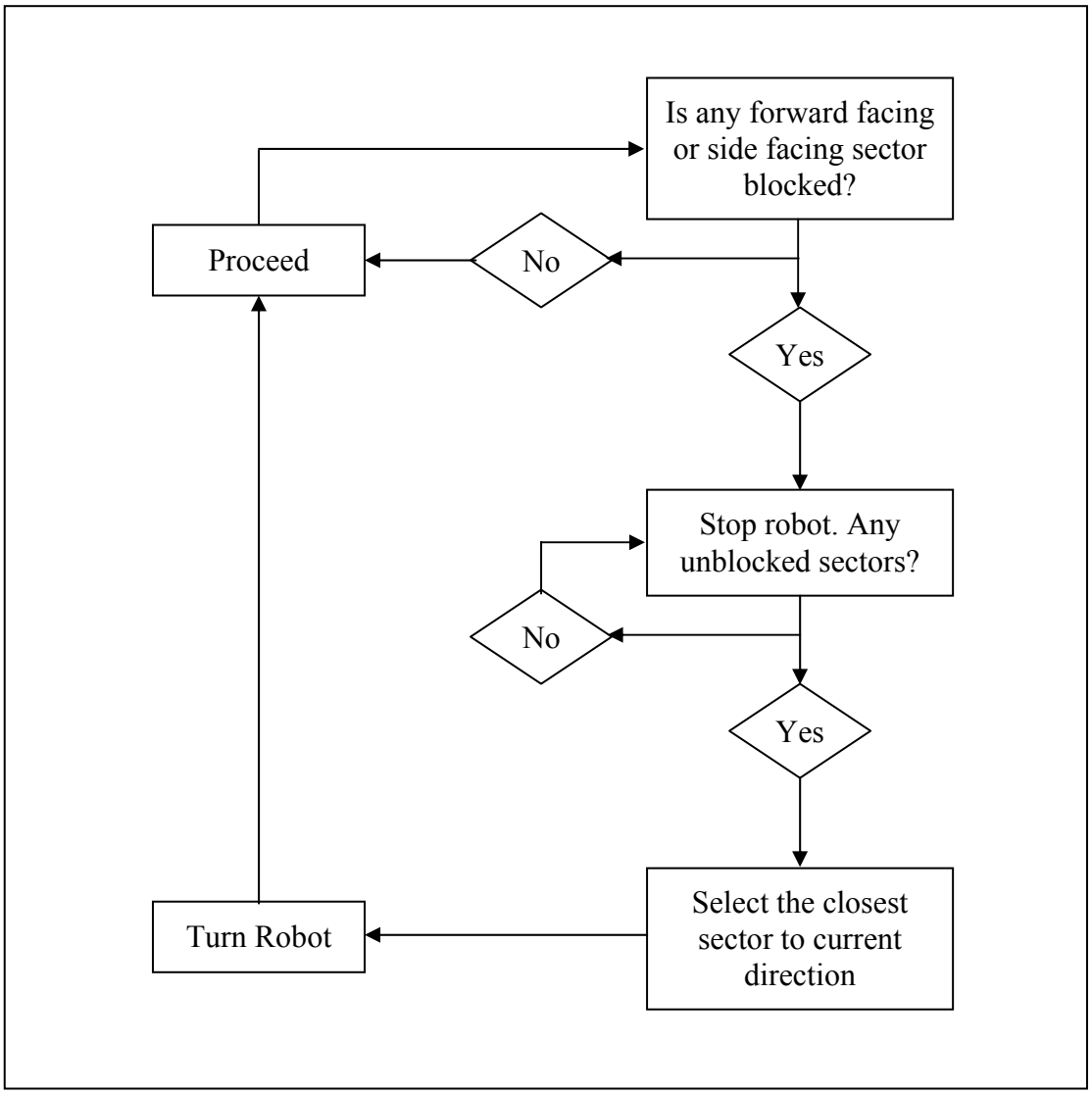


Figure 6-5: Obstacle avoidance behaviour algorithm

6.2.2 Target Detection

The robot will be able to perform autonomous navigation with the obstacle avoidance behaviour in place. Next, the robot requires the mission capable behaviour for it to complete the required task. Target detection is responsible for finding the targets and its priority follows after obstacle avoidance.

This behaviour relies on the eight light detectors positioned around the robot. See Figure 6-6. Similarly, eight sectors are considered. The light detectors can only detect line of sight light beacons and do not have wide-angle span like the ultrasonic sensors. When a light detector detects the light beacon, the respective sector is turned active.

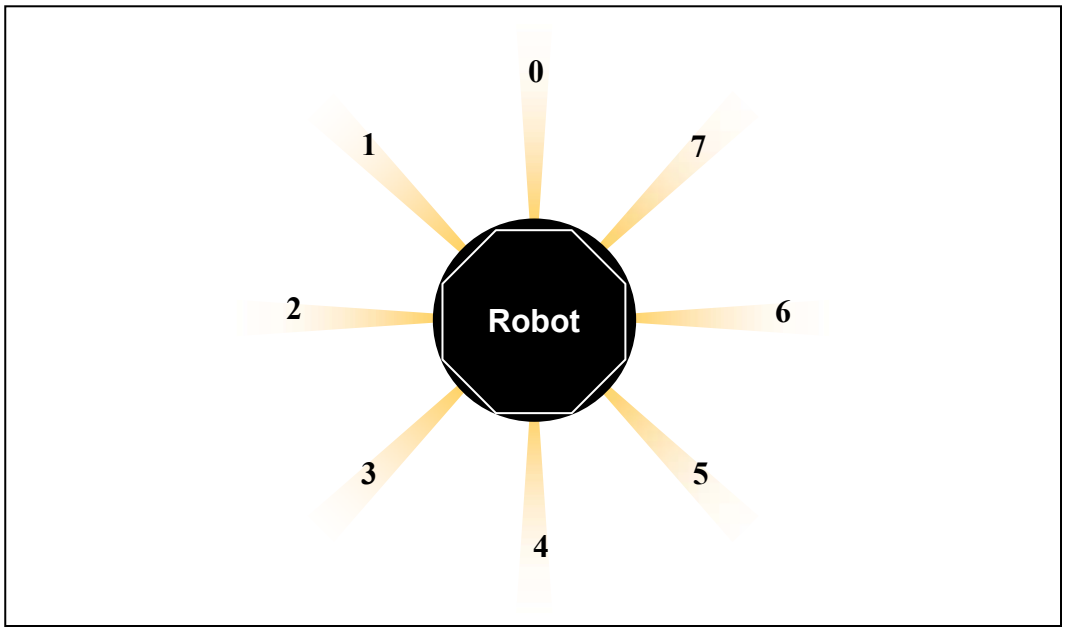


Figure 6-6: Light detectors around robot

Like obstacle avoidance, target detection behaviour also changes the robot's direction of motion. However, targets are now attractive instead of the obstacles repulsive effect. Each central axis of an active sector is a possible new candidate direction. The behaviour will randomly choose one if there are more than one active sector. Targets are considered found when it is within a certain range from the robot, fulfilling the definite range law in [25]. The robot stops and broadcast the "found target" message to other robots via the line-of sight infrared transceiver. See Figure 6-7. The algorithm is illustrated in Figure 6-8. In the CoSyBot physical set-up, obstacles

between the robot and target will block off the light rays. Hence, the target detection attractive effect will not conflict with obstacle avoidance's repulsive effect.

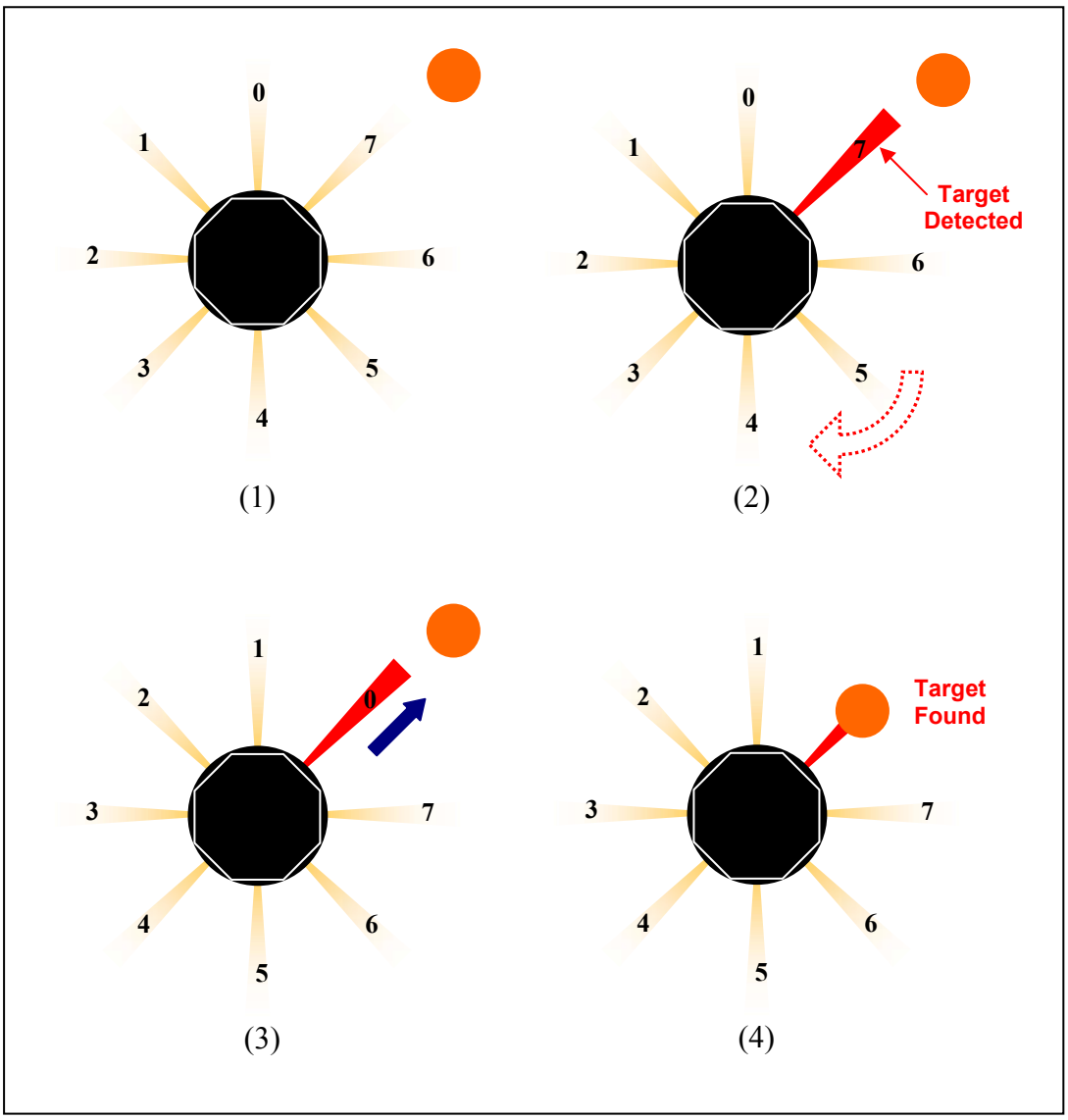


Figure 6-7: Illustration of target detection behaviour

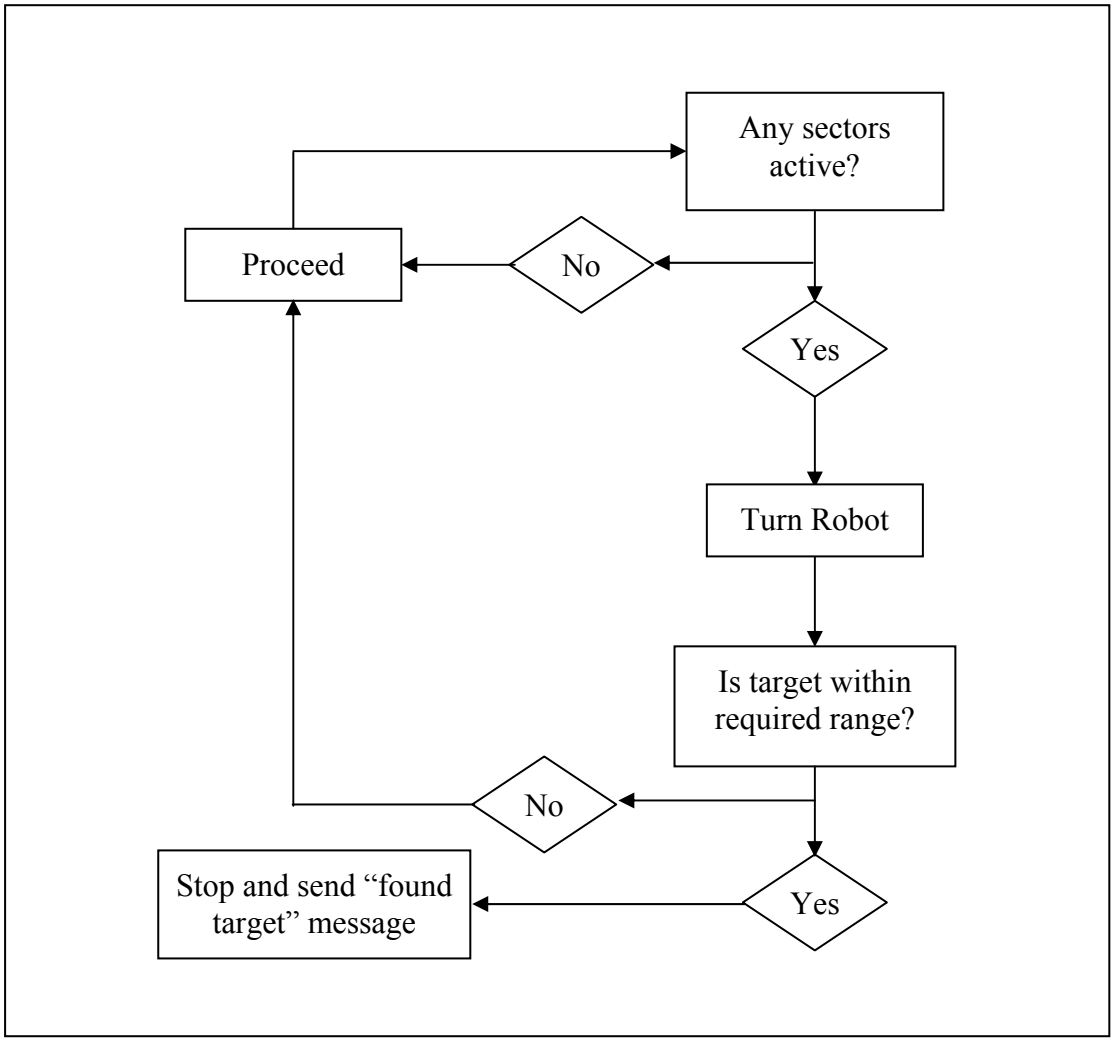


Figure 6-8: Target detection behaviour algorithm

6.2.3 Respond to Neighbouring Robot's Message

For multiple robots to be cooperative, some form of communication is required. Communication allows the robots to cooperate effectively, improving the overall performance. Hence, it is an important component for multi-robot systems. In our survey on the works by robotics researchers, they concluded that in general some simple local interactions among robots will improve the system performance. This behaviour is designed to provide for the local interactions among the robots.

The behaviour uses the eight line-of sight infrared transceivers for implicit communication among the robots. See Figure 6-9. Earlier, a robot will send the "found target" message when it has found a target. If the infrared transceiver in sector 0 receives this message, this implies a robot has found a target ahead of it. This behaviour will stop the robot and change its direction of motion. In selecting a new direction, the robot randomly chooses from the sectors that do not receive such messages from other robots. Again, the candidate directions are the respective central axis of the sectors. There are two reasons for this behaviour. Firstly, this prevents the scenario of having more than one robot finding the same target. Secondly, in this way the robots are compelled to explore other areas, increasing the probability in exploration of unknown space. See Figure 6-10. The algorithm is illustrated in Figure 6-11. The combination of this behaviour and target detection behaviour provides the local interactions for cooperation to find all targets.

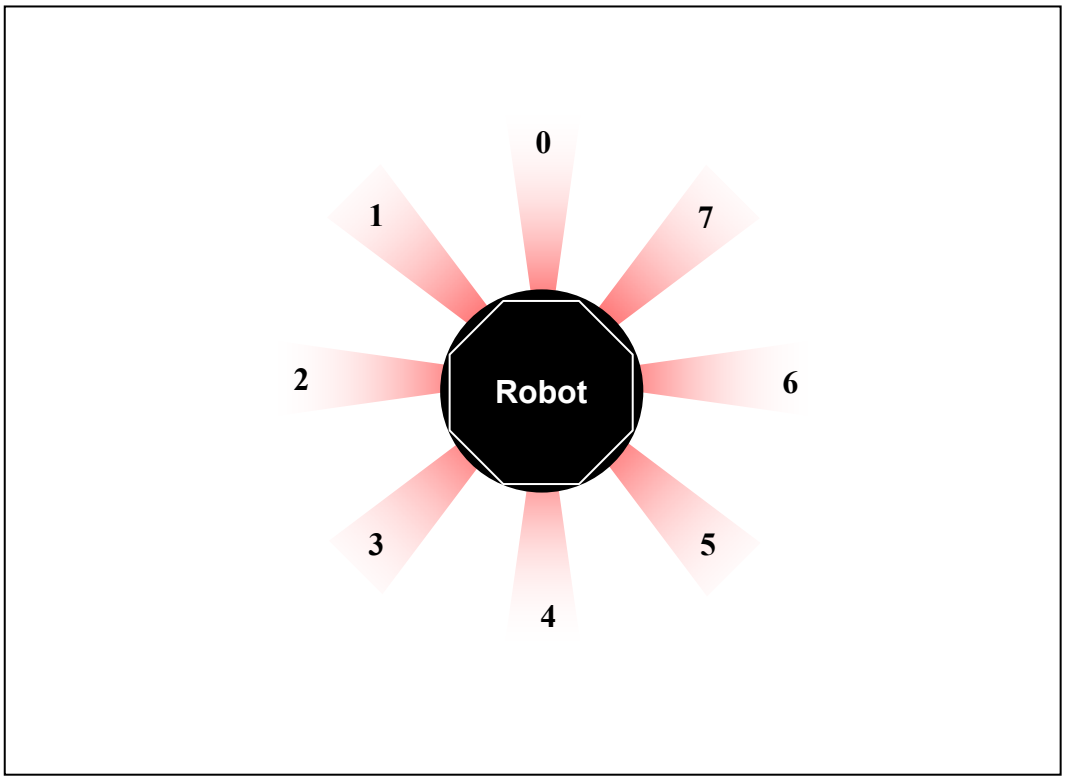


Figure 6-9: IR transceivers around robot

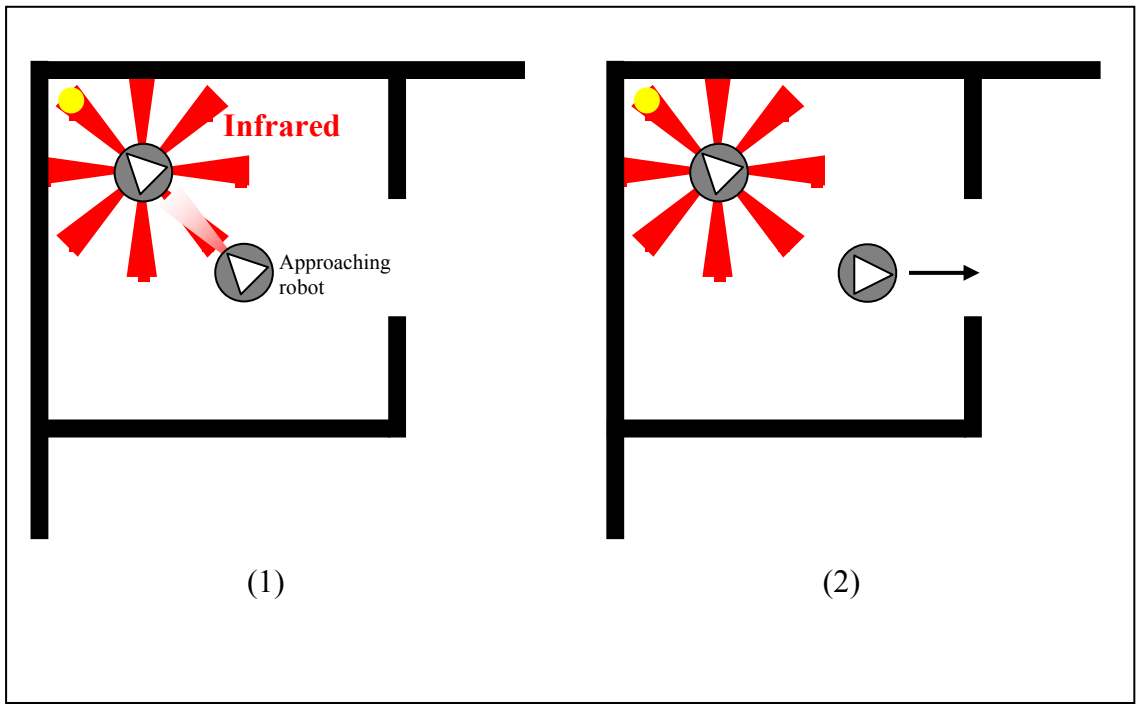


Figure 6-10: Illustration of respond to neighbouring robot's message behaviour

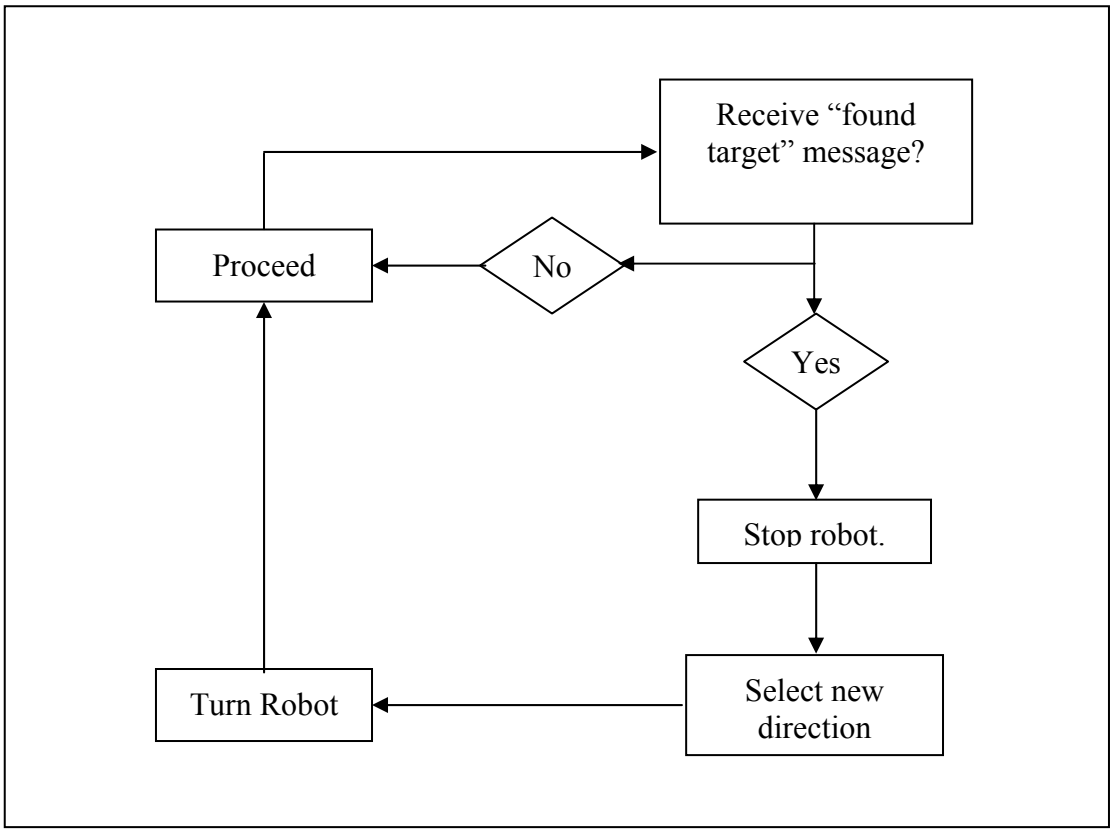


Figure 6-11: Responding to neighbouring robot's message algorithm

6.2.4 Follow External Commands

Follow external commands behaviour does not contribute directly to the autonomous control of the robots. It is not required for the robots to complete the mission but is a useful feature to the multi-robot system. It provides an avenue for the commander or user to intervene or control the robots. For example, the commander can inhibit the target detection behaviour at the beginning. This will cause the robots spreading out in the environment without the distraction from the targets. He can later activate this behaviour to complete the mission.

This behaviour uses the wireless network for global communication. This is achieved using the UDP protocol in winsocks network programming. Currently, the usage of this behaviour is limited to starting and stopping the robots. This is useful as starting and stopping large number of robots can be a difficult task.

6.2.5 Wander

Wander is the default behaviour for the robot. When none of the previous behaviour is active, wander is responsible for moving the robot in the environment. The robot just continues moving in the current direction. In addition, it also looks for openings to move into. This is useful as it increases the possibility of the robot moving into potentially unexplored areas.

This behaviour uses the ultrasonic range sensor to sense for openings to move into. Referring to Figure 6-2, it makes use of sectors 1, 2, 6 and 7. It senses for openings through detecting a large jump in the range readings for these sectors. If openings are

detected, the behaviour will select, from the candidate directions, the new direction to turn to or maintained the current direction. The decision is made randomly using a random number generator. In doing this, it prevents the robot from being trapped in a particular room. If there are no openings, the robot maintains its current forward motion. See Figure 6-12. The algorithm is illustrated in Figure 6-13.

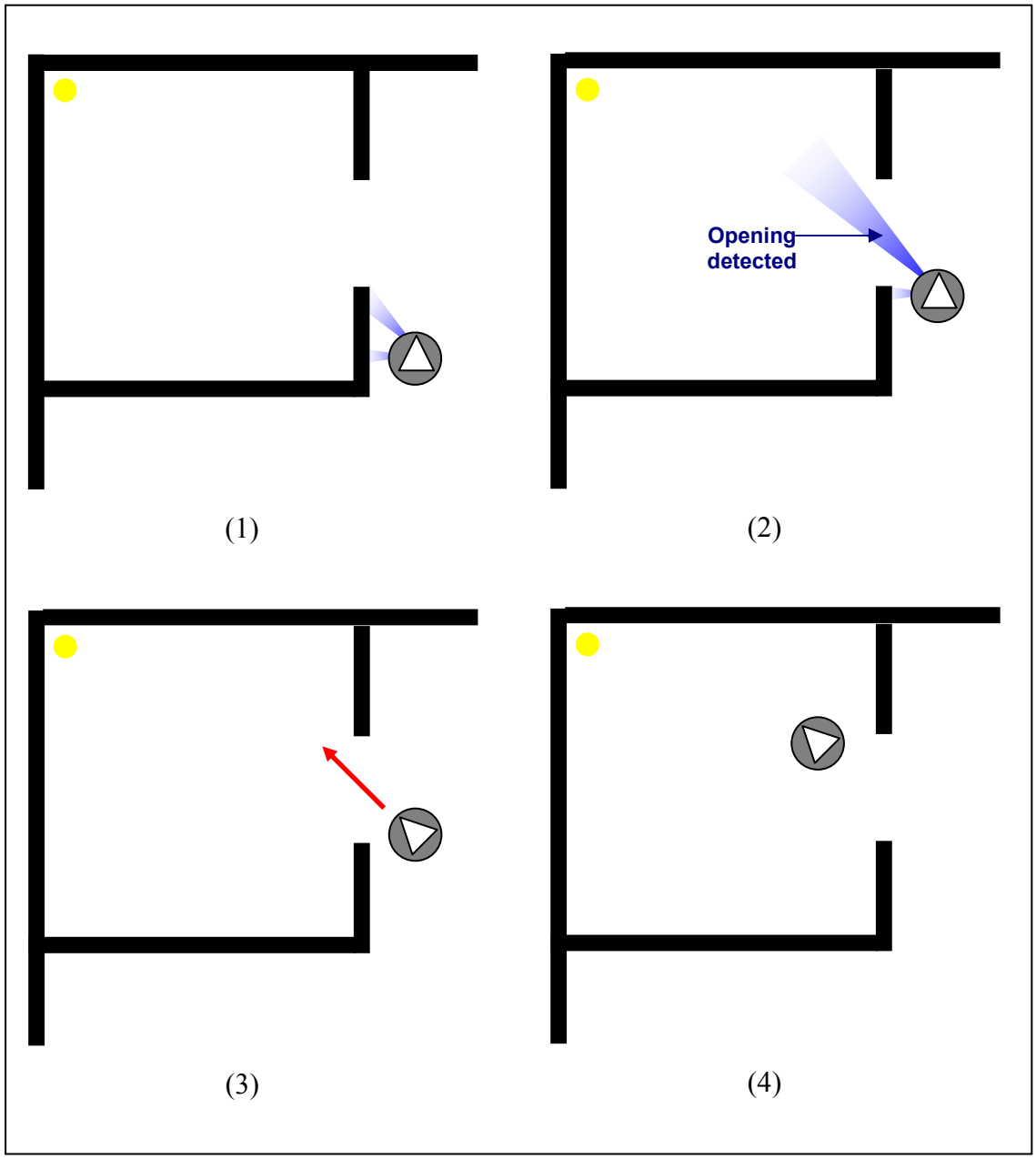


Figure 6-12: Illustration of wander behaviour

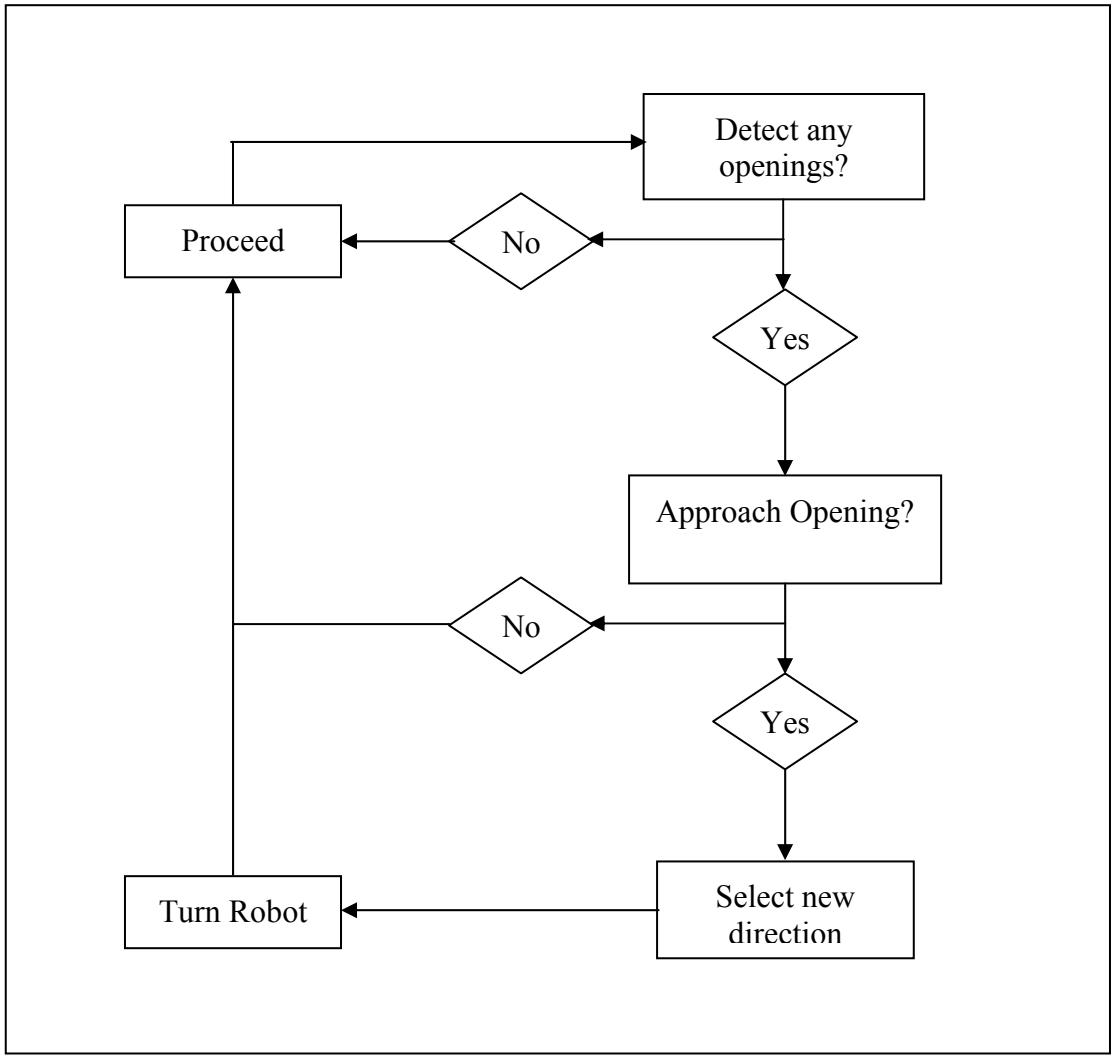


Figure 6-13: Wander behaviour algorithm

6.3 Implementing the Reactive Behaviours

The five reactive behaviours are implemented on five physical CoSyBot robots and the simulator. For all five behaviours, the robot does not require to know its position in the environment. Each robot is independent and has identical set of behaviours. The behaviours provide the local interactions among robots. For example, when a robot avoids a fellow robot, it changes to a new search direction.

The robot executes the behaviours in a sequential flow as shown in Figure 6-14. Behaviours that acquired sensors information fulfilling all its condition will trigger and send action commands to the robot. This allows the robot to respond quickly to changes in the environment. Since the reactive behaviours are simple and each requires little computation, sequential execution in real time is feasible. The detailed interactions between the behaviours are illustrated in Figure 6-15.

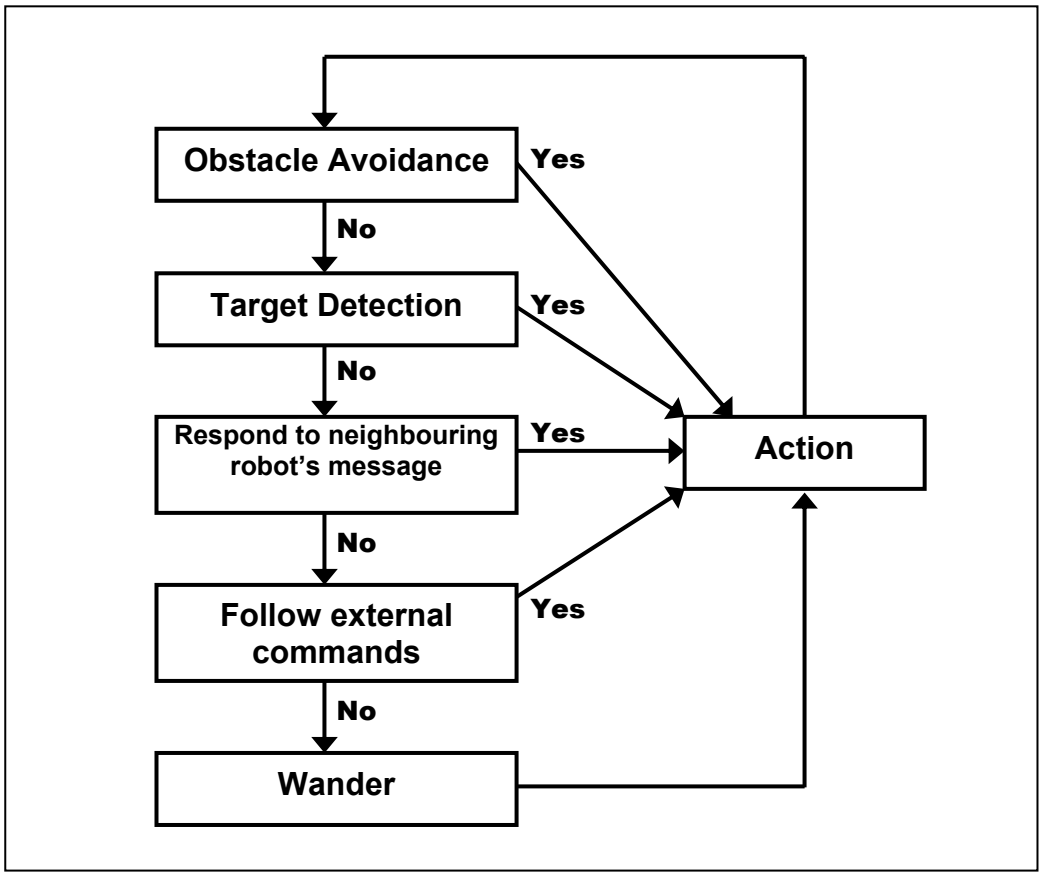


Figure 6-14: Sequential execution of the behaviours

How do we know that the five behaviours are sufficient to solve the search problem? In the earlier work by Balch et al. in [5], he has demonstrated in a similar foraging task that three behaviours are sufficient. The behaviours are: Avoid; Forage and Wander. This is similar to our behaviour set. Our additional behaviours are: Respond to

neighbour's messages and Follow external commands. Follow external is used only to start or stop the robots. As discussed earlier, respond to neighbour's messages behaviour provides the addition local interactions to improve systems performance. In addition, it ensures that one target is found by only one robot. Hence, all robots are employed to search for different targets and all targets can be found as long as there are more robots than targets.

6.4 Chapter Summary

In this chapter, we implemented the proposed random search algorithm into five reactive behaviours. The behaviours are: (1) Obstacle Avoidance, (2) Target Detection, (3) Respond to neighbouring robot's message, (4) Follow external commands, and (5) Wander. We described the algorithm for each of the individual five reactive behaviours and illustrated how they integrated with the capabilities of CoSyBot. The behaviours are executed in a sequential flow in order of their priorities. This is possible to control the robot in real time as the behaviours are simple and require little computation. Lastly, we are confident that these behaviours are sufficient because similar works previously have demonstrated a foraging task with three behaviours: Avoid; Forage and Wander. We have additional behaviours for local interactions to improve system performance and ensure that all targets will be found.

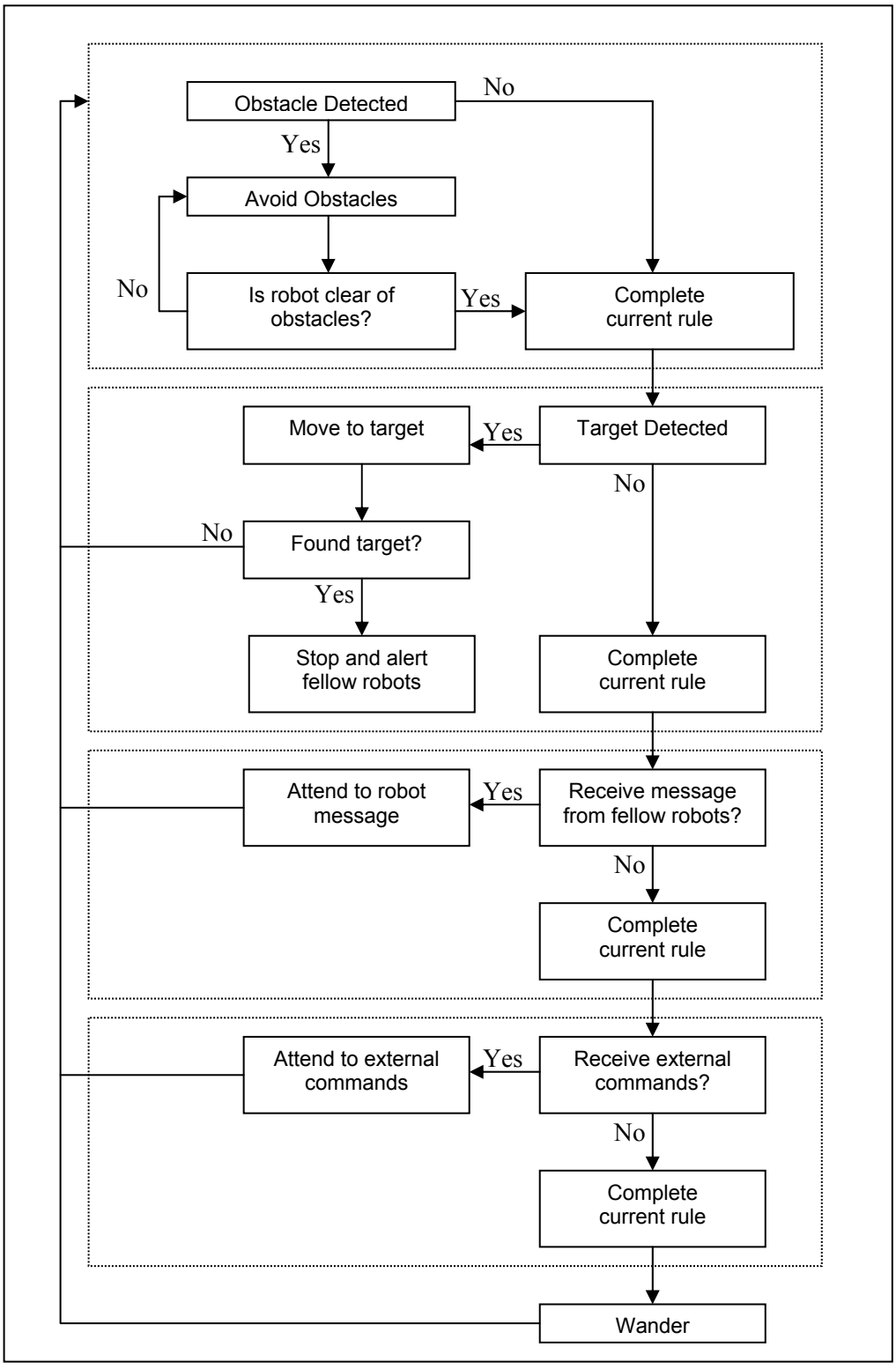


Figure 6-15: Interaction of the behaviours

Chapter 7: Analysing the System Performance

In the previous few chapters, we covered the design and implementation phases of this project. In this chapter, we demonstrate that our proposed random search algorithm can solve the search problem. We integrated the five behaviours together in a simulation experiment to verify that our algorithm works. Then, we moved to physical experiments to demonstrate that our algorithm works on real robots in a real world environment.

Continuing to the next project objective, we performed multiple simulation experiments to analyse the performance of our algorithm. In these experiments, we varied the number of robots, robot starting positions and the size of the search environment. Finally, we discuss the results and observations from these experiments.

7.1 Testing the Algorithm in Simulation

In the previous chapter, we reasoned that the five reactive behaviours are sufficient to solve the posed search problem, supported with findings in previous works by other researchers. In this section, we will test the five reactive behaviours in simulation to demonstrate that they work.

The set-up in the simulation is designed to simulate the indoor of a building with multiple rooms. We make the following assumptions: (1) Size of openings leading to

rooms is wide enough for the robot to move through, (2) the number of robots must be greater or equal to the number of targets, and (3) targets should be located in the environment that is accessible to the robots. This set-up will also be used in our physical experiments.

7.1.1 Experiment Set-up

In this simulation test, we deployed five CoSyBots to search for three targets in a structured environment. The environment created is a 4m by 4m bounded building with multiple rooms. Figure 7-1 shows the simulation set-up. There are three targets (yellow circles), each placed in separate different rooms. While, all five robots (Blue circles with arrowhead) start from the same room. The speed of each robot is set to 0.3 meters per second. The range of the target sensor is approximately 1 meter, which is less than the shortest distance from the opening to the target of all rooms. In other words, the robots must enter the rooms to find the targets. The time taken for all three targets found is used to measure the system performance. The aim of this experiment as stated earlier is to verify that our proposed random search algorithm works. A hundred simulation runs is repeated to generate a sufficiently large sample size.

7.1.1.1 Results and Analysis

The results obtained for the hundred simulation test runs is illustrated in **Figure 7-2** and listed in Table 7-1. All three targets are found in all hundred simulation test runs. Therefore, our proposed random search algorithm is sufficient to solve the search problem.

From the results in **Figure 7-2**, we observed that the performance of the algorithm fluctuates greatly for the hundred runs. The mean time is 216 seconds with standard deviation of 105 seconds. This is due to two reasons. Firstly, the randomised strategy in the algorithm resulted in robots taking different motion paths even for the same set-up, resulting in different results for each run. Secondly, there are few robots in the environment to provide consistent local interactions for cooperation.

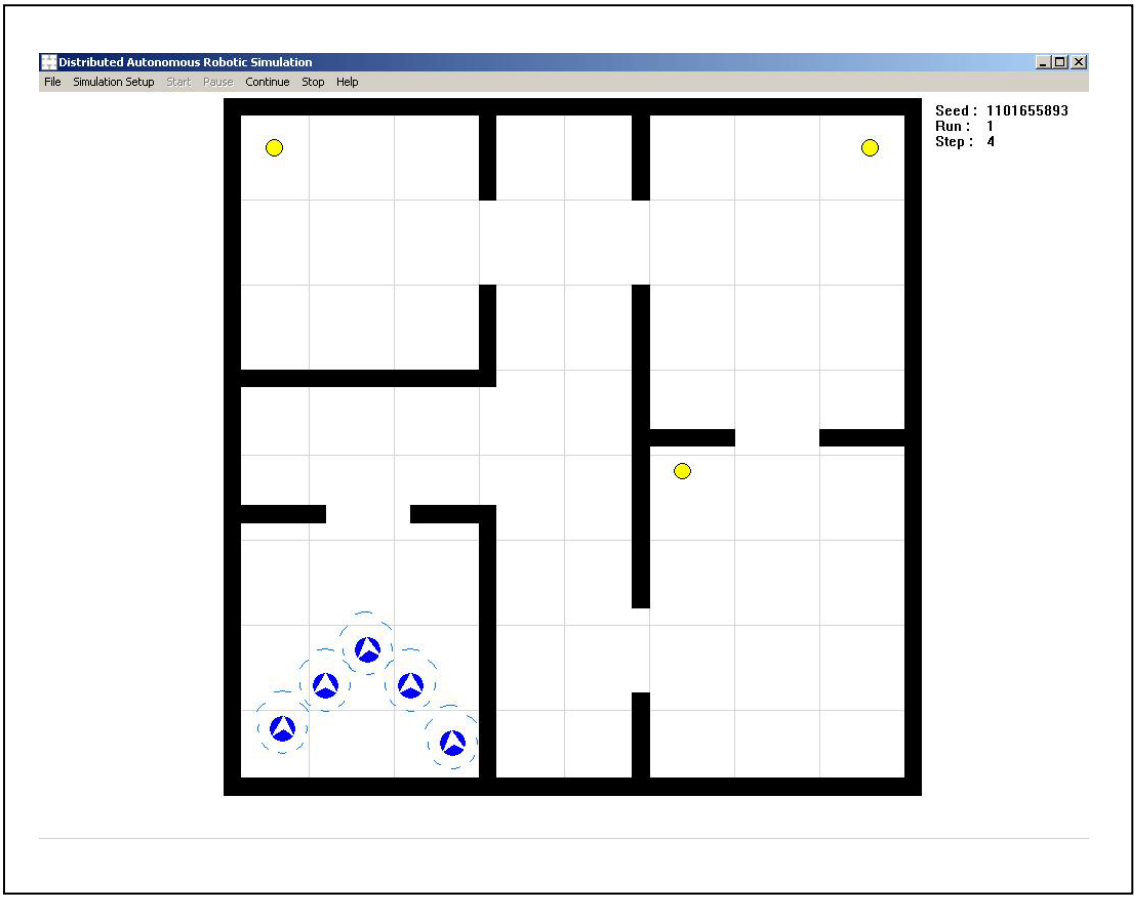


Figure 7-1: Simulation test set-up

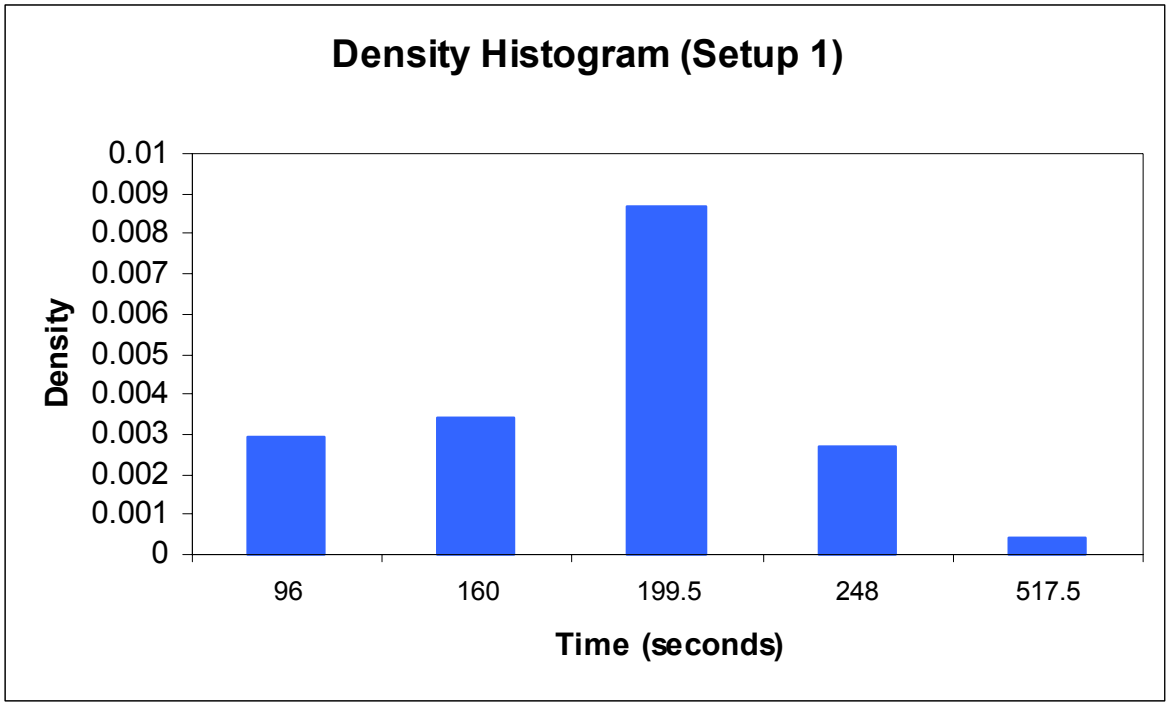


Figure 7-2: Results of 100 simulation test runs

Table 7-1: Results of simulation test

Five robots team	Mean, μ /s	Standard Deviation, σ / s
100 Simulation runs	$\mu_s = 216$	$\sigma_s = 105$

7.2 Physical Experiments

The proposed search algorithm has been shown to work in our simulation test. Hence, we proceed to implement it on the physical robots. Physical experiments are conducted to demonstrate that the developed reactive behaviours are feasible on actual physical hardware.

7.2.1 Experiment Set-up

We duplicated the simulation test set-up, described in the earlier section, in the physical experiment. See Figure 7-3. The physical robots also have similar capabilities. There are two aims for this experiment. First is to demonstrate that the proposed algorithm works on physical robots in a real world environment. Secondly is to verify the fidelity of the simulation program to the physical experiments. Ten physical experiment runs is repeated for this set-up.

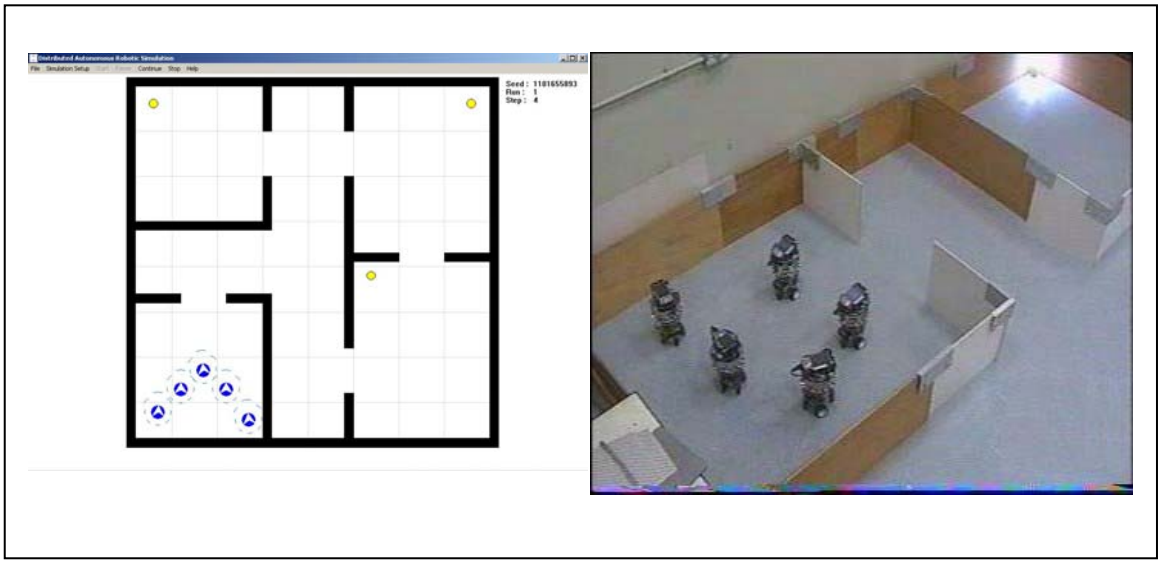


Figure 7-3: Physical experiments layout

7.2.2 Robots Searching for Targets

Figure 7-4 shows screenshots of a video clip captured in one physical experiment run. The screenshots show only portions of the set-up because the ceiling in the lab area is not high enough for our video camera to capture the full set-up. Screenshot (1) shows five CoSyBots at the starting position in one room and a target represented by a light beacon in another room. We start the robots using a separate Pocket PC to send commands to the robot through the wireless network. The user or commander just

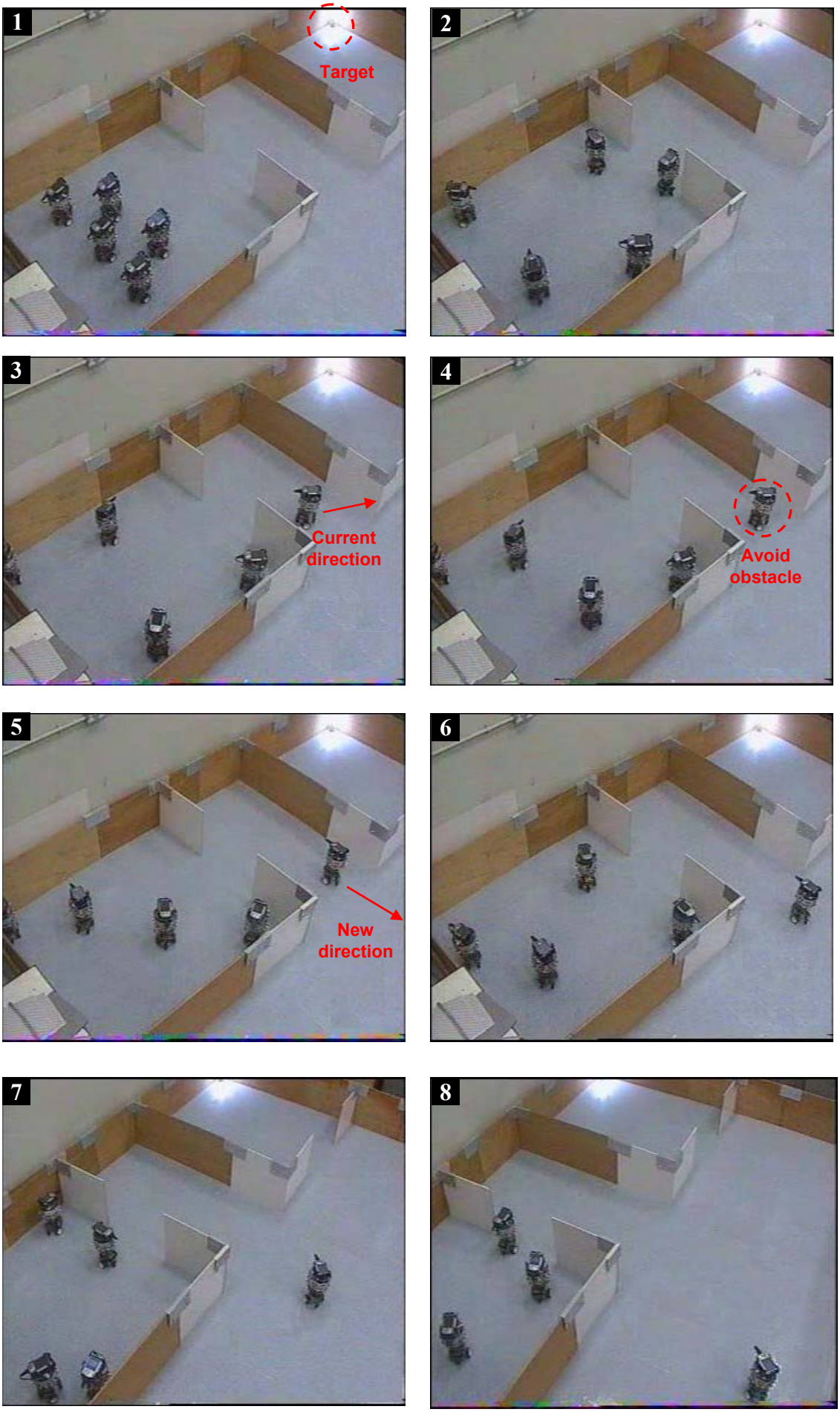
needs to send a single "start" command, which is broadcasted to all the robots. And the robots will start together upon receiving the command. Similarly, this applies to "stop" and other commands. In addition, the user could also send commands to a specific robot. To do this, he just needs to include the intended robot's identification number in the commands. In our set-up, each robot has a unique identification number and the robots will ignore the commands if their identification number does not match. These demonstrate the "Follow External Commands" behaviour on the physical robots.

Screenshots (3), (4) and (5) illustrate the robot performing the "Obstacle Avoidance" behaviour. The robot detected the wall within the trigger distance of its forward sensing sectors. It then selected the new direction to turn to according to the algorithm described earlier. Screenshot (6) shows the robot had successfully avoided the wall. Throughout the physical experiments, the robots were able to avoid collisions using this behaviour.

The "Target Detection" behaviour is illustrated in screenshots (13), (14) and (15). When the light detectors on the robots detected the target, it triggered the behaviour to approach the targets. The robots considered the targets found when they are within a prescribed range away as shown in screenshot (15). Then the robots stopped and start sending IR messages that the targets in the rooms are found. This interacts with the "Respond to Neighbour's Messages" behaviour to expel other robots away. In this way, there will be only one robot to each target found.

When there are no obstacles, targets or IR messages, the robots will just wander in the environment with the default "Wander" behaviour as shown in the screenshots. If the robot detects an opening, this behaviour will also randomly decide to guide the robot to move through the openings. In screenshot (17), the robot was moving parallel to the wall. It detected the opening when moving past it and the "Wander" behaviour turned the robot to move into the room illustrated in screenshots (18) and (19).

Each physical experiment terminated when the last target is found, for example in screenshot (20). The screenshots show that all targets are found, each by one robot. This is similar for other physical experiment runs. Therefore, the physical experiments demonstrated that our proposed algorithm is capable of solving the required problem and is feasible to be applied on physical robots.

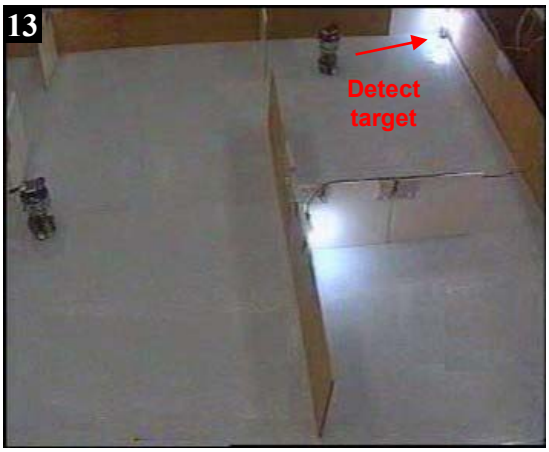

1
Target
2
3
Current direction
4
Avoid obstacle
5
New direction
6
7
8


9

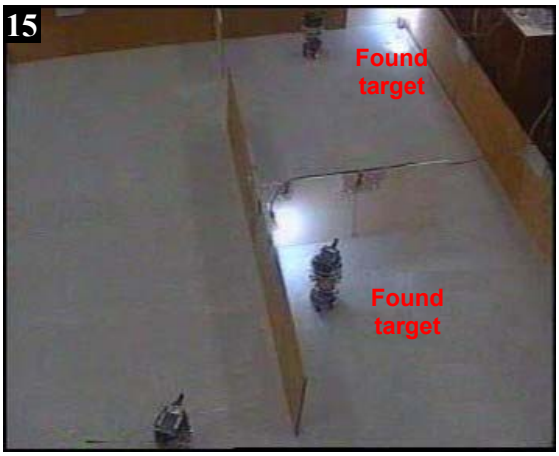

Target
10


11
Target


12


13
Detect
target


14
Detect
target


15
Found
target
Found
target


16

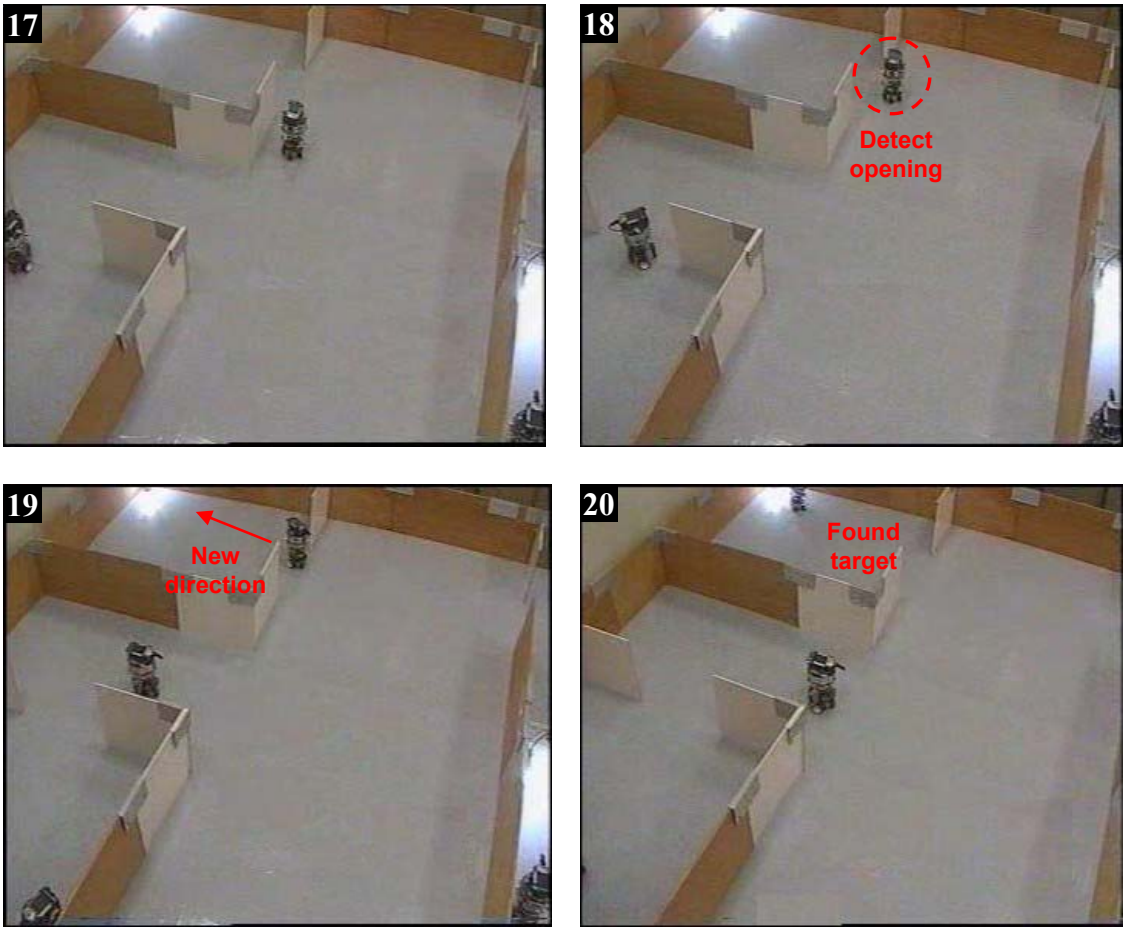


Figure 7-4: Screenshots of a physical experiment

7.2.3 Physical Experiments Results and Observations

The robots found all three targets in all ten physical experiment runs. The results are listed in Table 7-2. The mean time taken is 249 seconds.

Table 7-2: Results for ten physical runs

Runs	1	2	3	4	5	6	7	8	9	10
Time taken /s	254	191	320	237	127	304	423	175	219	238
Mean, μ / s	249			Standard Deviation, σ / s			84			

During the runs, we observed that the robots have random motion paths. The robots do not repeat the same motion path even though they start from the same positions for each run. This is similar to the simulation test runs because of the randomness nature present in the behaviours.

In addition, we also did physical experiments with different layouts, target locations and starting positions. The results for these experiments are not tabulated. This is because the main motive is to verify that the search algorithm is not unique to solving the layout shown in Figure 7-3. In these experiments, the robots were still able to find all the targets. Hence, the search algorithm is demonstrated to be robust to the robot starting position and the layout of the environment.

In a few experiments, there were instances of robot 'attrition'. For example, a robot stopped moving due to failure in the hardware. However, this did not affect the rest of the robots, they still continued to search for targets and found all targets. We also performed experiments beginning with four robots, and adding one robot later during the experiment. This did not affect the system. We can conclude that the algorithm is robust to robot failure and scales in numbers easily.

7.2.4 Comparing with Simulated Test Results

The results obtained for the five robots team for both the simulated test and physical experiment are listed in Table 7-3.

Table 7-3: Simulation test and physical experiment results for five robots team

Five robots team	Mean, μ /s	Standard Deviation, σ / s
100 Simulation runs	$\mu_s = 216$	$\sigma_s = 105$
10 Physical runs	$\mu_s = 249$	$\sigma_p = 84$

The average time for five robots in the simulation test runs is 216 seconds, which is close to the 249 seconds obtained in physical experiments. The physical experiment mean is $0.31\sigma_s$ from the simulation experiment mean. This is much smaller than one standard deviation away. Hence, the simulation experiment results are reasonably close to the physical experiments.

7.3 Simulation Experiments

Further experiments on the algorithm are done using the simulation program. The aim of these simulation experiments is to analyse the performance of the proposed random search algorithm. Various system parameters are varied for the analysis and simulation experiments will allow us to perform the analysis more rapidly. Moreover, it is not feasible and practical to perform many explicit physical experiments, as the system parameters cannot be easily varied and time consuming having each run at real time for large sample runs. For example, changing the number of robots or the environment size and repeating the experiments for a hundred runs. The results for the simulation experiments can be found in Appendix C.

7.3.1 Varying the Number of Robots

In any multi-robot system, one important system parameter to consider is the number of robots in the team. This parameter has a direct influence on the system

performance. Intuitively, having more robots in the team should improve the performance. However, it can also cause the whole system to fail if the multi-robot control system is unable to handle the number.

7.3.1.1 Experiment Set-up

Using the same set-up in Figure 7-1, we varied the number of robots from a four robots team to a twenty robots team at the same starting position. There are two aims of this experiment. First is to analyse the effect of the number of robots on the system performance. Second is to test whether the algorithm is scalable in number. Similarly, a hundred simulation runs is repeated for each number of robots to produce a sufficiently large sample size for the analysis.

7.3.1.2 Results and Analysis

Figure 7-5 shows the results with the mean time displayed on a logarithmic scale. The graph shows the mean time, over hundred runs, taken for each of the robot team sizes. Two observations can be made from the graph. Firstly, the system performance improves with the number of robots. This is expected as having more robots in the team means there are now more robots performing the task. It increases the parallelism advantage of the multi-robot system. Hence, it suggests that increasing the number of robots will increase the probability of success. Secondly, the system performance reaches a point where there is no significant improvement with increasing number of robots. From Figure 7-5, the number is about ten robots for this given environment. After ten robots, the graph tends to a horizontal line. This is expected, as having a larger number of robots with no change in the size of the environment will lead to over-crowding. This increases the amount of interference each robot exerts on

fellow robots. The robots are spending more effort avoiding collision with each other than performing productive work to complete the mission. Hence, it reduces the efficiency of the system.

Throughout the experiments, no changes are required on the algorithm to increase the number of robots in the team. We simply add the robots, each with the same identical set of behaviours, to the team. Therefore, our algorithm is easily scalable in numbers.

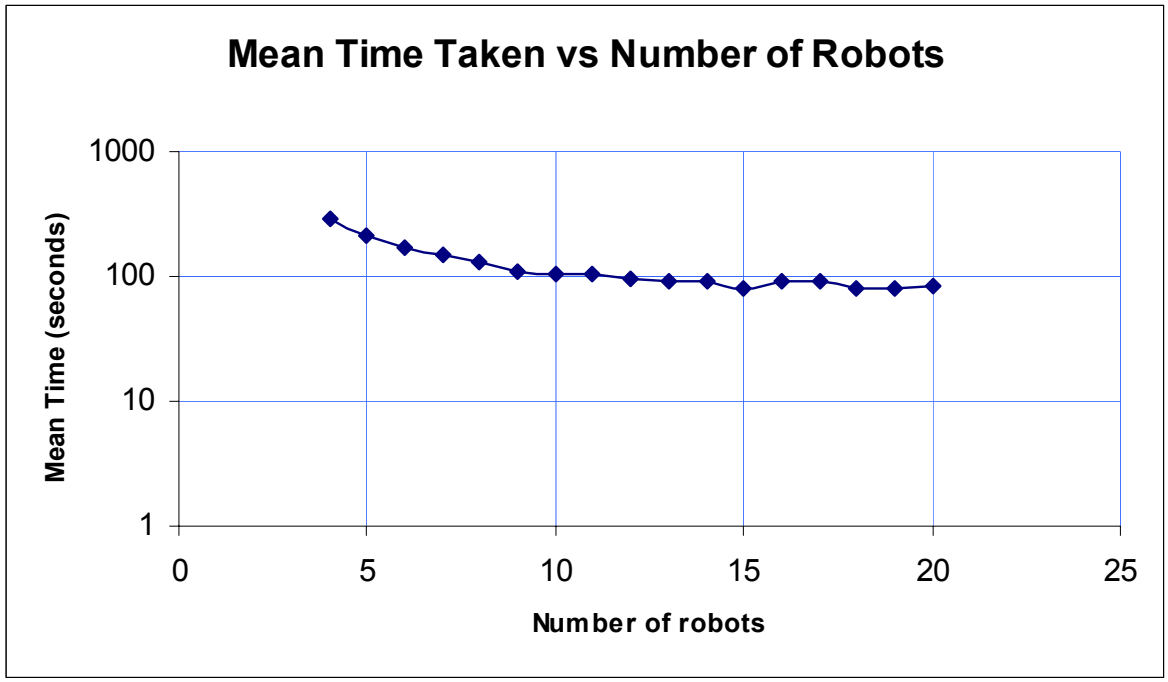


Figure 7-5: Graph of mean time (on logarithmic scale) taken to find all targets against number of robots

Lastly, in section 7.1.1.1, we stated that one of the reasons for the highly random results is that there are too few robots in the environment to provide consistent local interaction. From Figure 7-6, the standard deviation for the results obtained decreases with the number of robots. Hence, this result support our hypothesis made earlier.

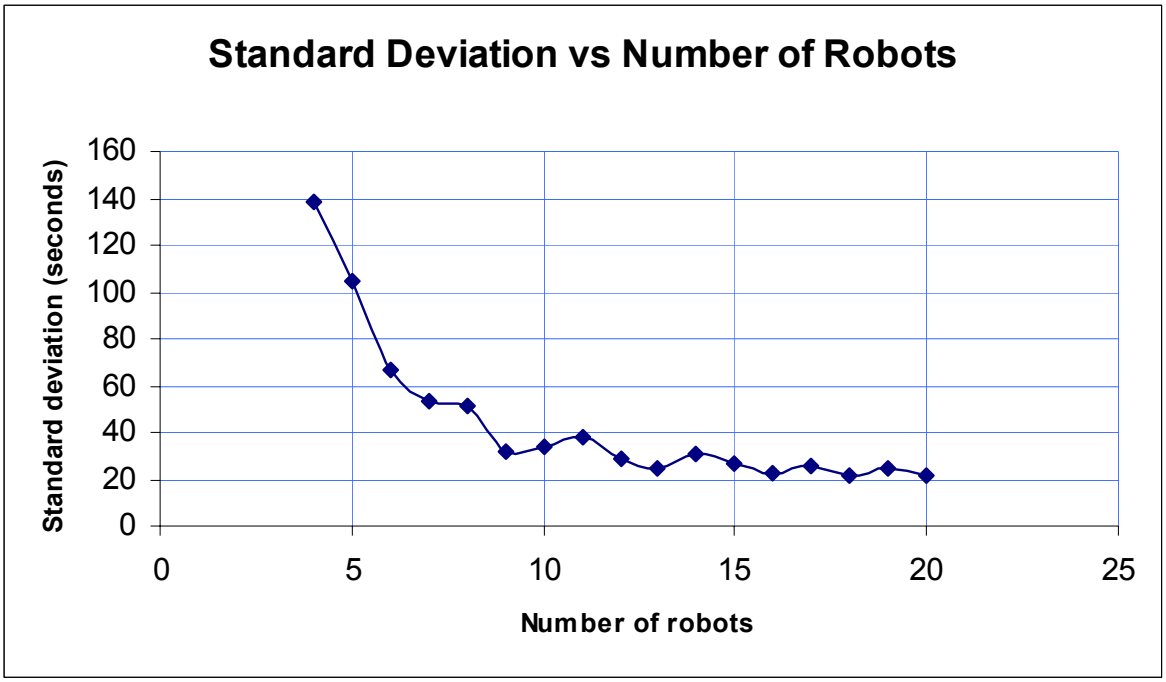


Figure 7-6: Standard deviation against number of robots

7.3.2 Varying the Starting Positions and Targets' Positions

In this set of experiments, we varied the robots' starting position and targets' positions in the same environment layout.

7.3.2.1 Experiment Set-up

The aim of this experiment is to verify that the algorithm is not dependent on the robots' starting position and targets' positions. Figure 7-7 shows the different set-up used. Set-up (1) is used in the previous experiments. From the figure, the robots start in a different room with the targets shifted accordingly for each set-up. The environment layout is kept the same because it affects the complexity of the environment. Thus, for a fair comparison, we kept the layout and size unchanged. In

addition, we also varied the number of robots from four to twenty. Similarly, a hundred simulation runs is repeated for each set.

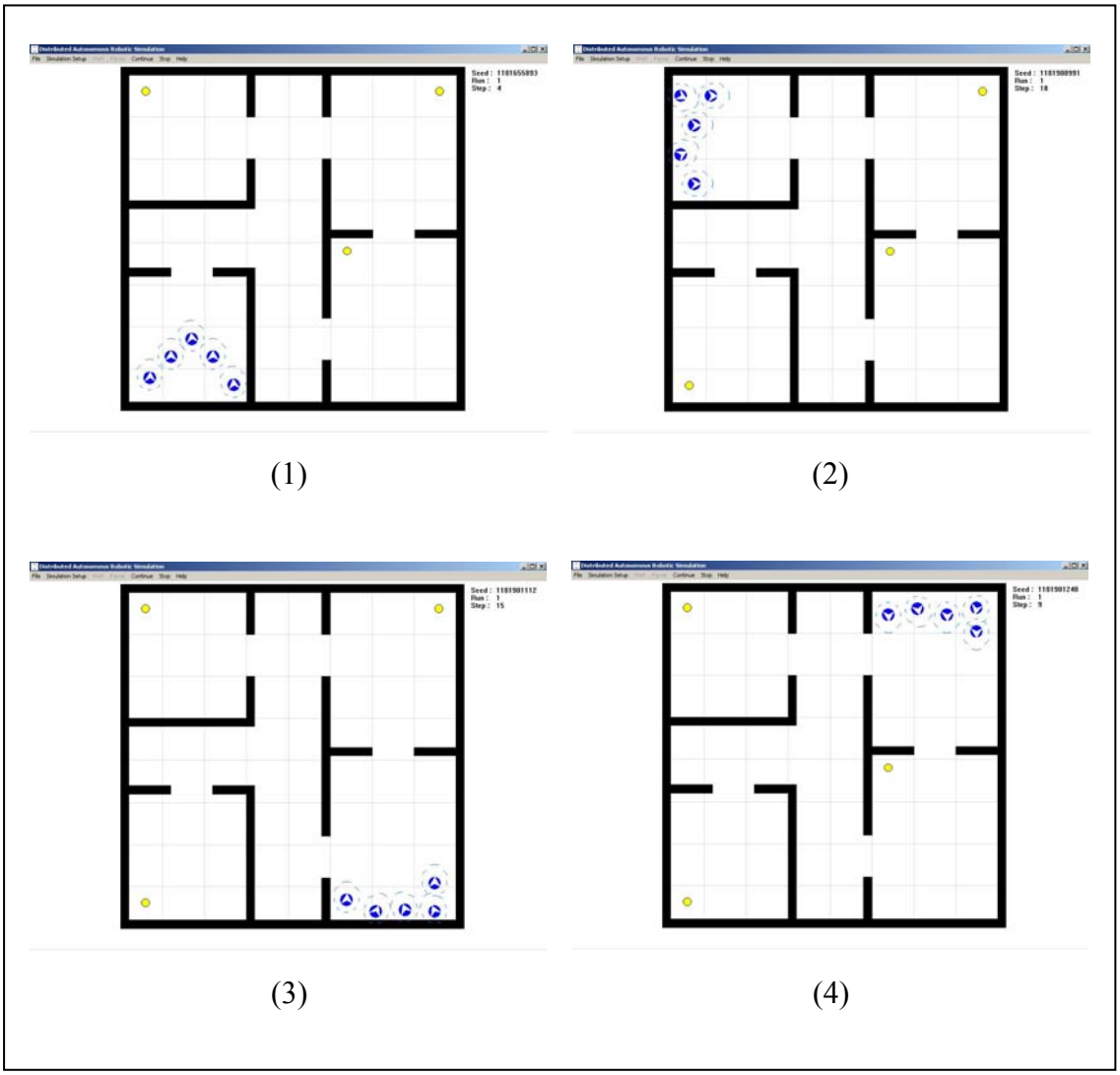


Figure 7-7: Different robots' starting position and targets position

7.3.2.2 Results and Analysis

Figure 7-8 shows the results with the mean time displayed on a logarithmic scale. The graphs show the mean time taken to find all targets for different number of robots in each of the four set-ups. A few observations can be made from the graphs. Firstly, all

four set-ups show a similar trend. They show an initial sharp decrease in the time taken to find all targets with increasing number of robots. This decrease in the mean time, i.e. improving system performance, gradually becomes insignificant after a certain point. Secondly, the system performance for the same number of robots differs for each set-up. This is evident from the graphs. For example, the four robots team in set-up four took the longest mean time, approximately 1.5 times more than the rest. This is because the complexity of the environment changes with the robots' starting position. The robots may be starting in some positions that have difficulty accessing other rooms. This suggests that for the same environment layout and size, the number of robots, their starting positions and target positions are factors affecting the system performance. Thirdly, the effect of robots' starting position and target positions on system performance decreases with increasing number of robots. From the graph, the system performance for the four set-ups is consistently closer to each other after ten robots. More robots suggest more local interactions among them and increasing the possibility to explore new areas.

Although the system performance varies for each set-up, the robots were able to complete the required task. Hence, the algorithm is robust and not dependent on the robots' starting position and targets' positions.

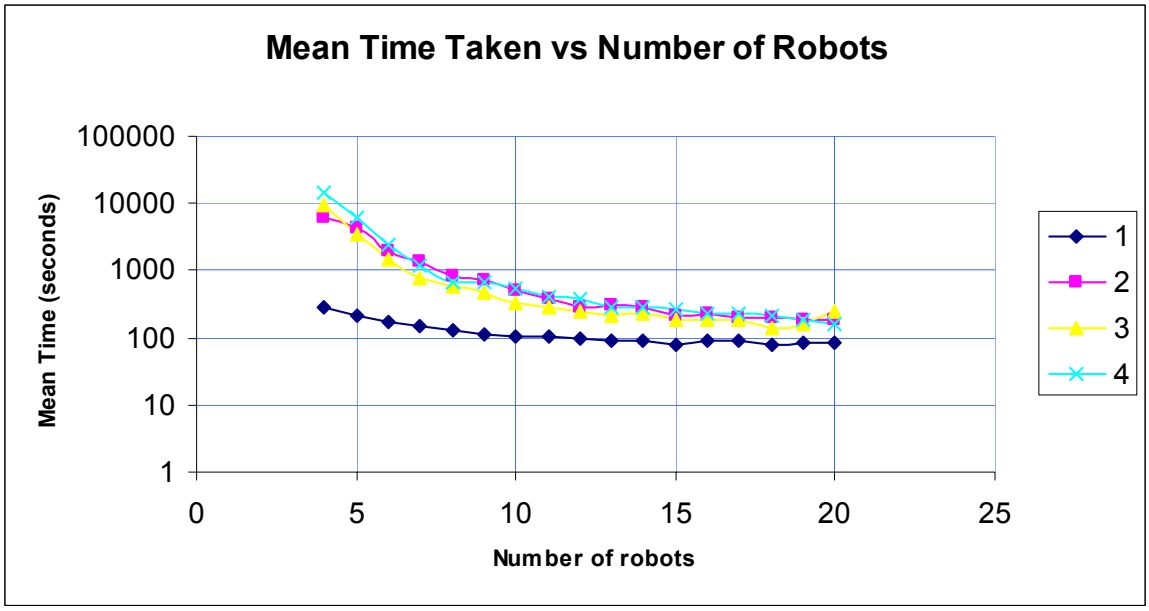


Figure 7-8: Experimental results of different robots' starting position and targets' positions

7.3.3 Increasing the Environment Size

In this set of experiments, we increased the size of the environment.

7.3.3.1 Experiment Set-up

The size of the environment is scaled up two times while keeping the same environment layout. See Figure 7-9. Comparing to Figure 7-7, noticed that the environment is now twice its original size and the robots' starting position and target positions are similar. The robot's physical characteristics, such as sensing range and speed, are kept the same. The aim of this experiment is to study the effect of the environment size on the system performance. Similarly for a fair comparison, we kept the layout unchanged. The number of robots is varied from four to twenty and a hundred simulation runs is repeated for each set.

7.3.3.2 Results and Analysis

Figure 7-10 shows the results with the mean time displayed on a logarithmic scale. The graphs show the mean time taken to find all targets for different number of robots in each of the four set-ups. We would expect the robot to take longer to find all targets, since the environment area is now twice as large. The robots have to traverse a longer distance to find the targets. Surprisingly, the robots performed better in a bigger environment. This is clearly observed when comparing the results in Figure 7-8 and Figure 7-10. We made certain observations from the simulation to explain for this better performance. Firstly, the robots are now smaller relative to the environment. This allows the robots to navigate through narrow passageways and openings to rooms easily. These difficult environment features are now less tight in space for the robots. Hence, the robots could enter rooms to find targets with lesser difficulty. Secondly, the bigger free space in the environment reduced the interference among robots. The robots have a bigger free space to move about, reducing the encounter times with other robots. In this way, the robots spend less time avoiding fellow robots and more effort exploring the environment. To support this hypothesis, we can compare the number of obstacle avoidance behaviour routine calls for each set-up in both environment sizes. For a fair comparison, we should consider the number of obstacle avoidance behaviour routine calls relative to the total number of reactive behaviour routine calls in the robot team. Table 7-4 lists the ratio of the relative number of obstacle avoidance behaviour routine calls in a ten-robot team for the double size environment to the original size environment. The ratios for all the four set-up are less than one. Hence, there are lesser obstacle avoidance behaviour routine calls in the robot team for the bigger environment. This is inline with our hypothesis.

The size of a real world environment is fixed and usually followed certain standards. We cannot change the environment size. Hence, the results suggest that using small robots is useful for the proposed algorithm. This is inline with the objectives of the algorithm, which is to use simple cheap robots. Our algorithm uses local reactive behaviours. It is not able to perform optimal path planning with only local environment information. However, using small robots will allow it to navigate through tight areas with lesser difficulty. This is similar to the argument of having shorter trigger distance for obstacle avoidance discussed earlier.

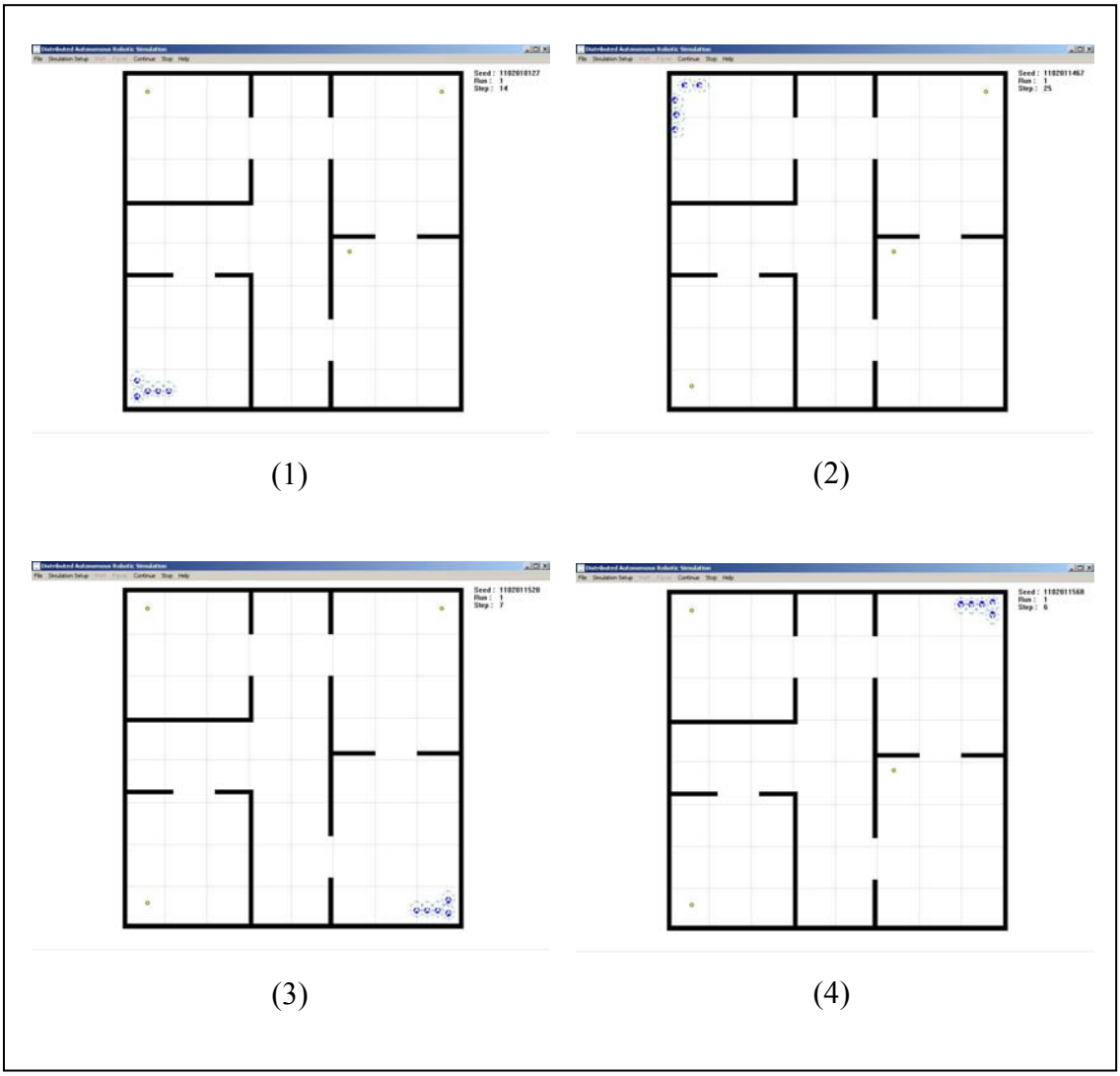


Figure 7-9: Set-up for scaled environment experiments

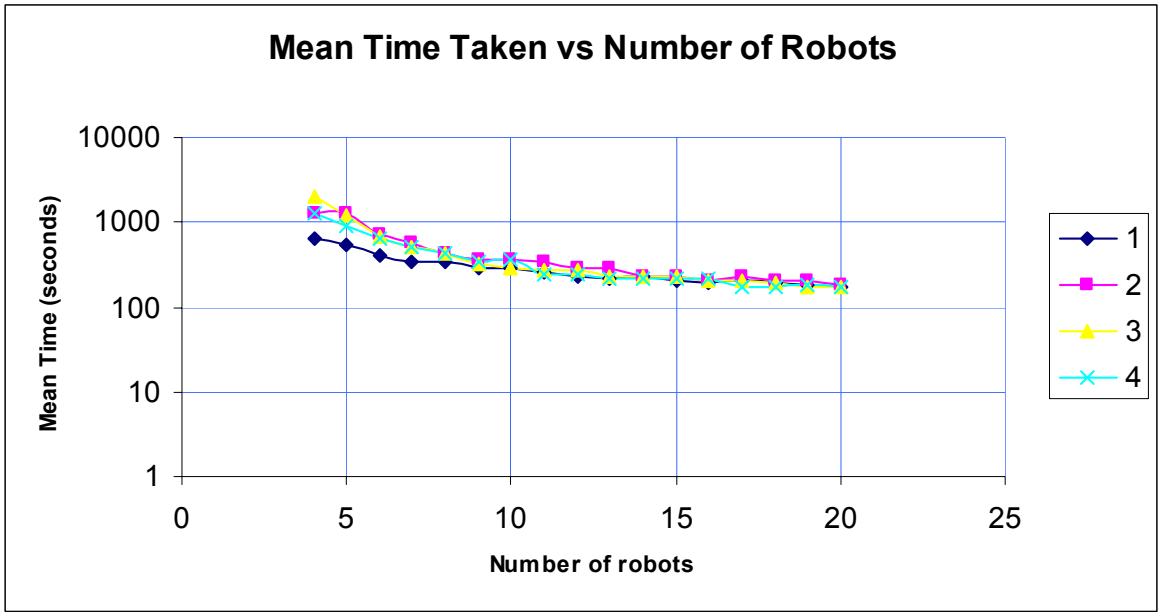


Figure 7-10: Experiment results for scaled environment experiments

Table 7-4: Ratio of the relative number of obstacle avoidance behaviour routine calls for the four set-ups

Ten-robot team	Ratio for double size to original size
Set-up 1	0.6
Set-up 2	0.4
Set-up 3	0.3
Set-up 4	0.4

7.4 Discussions

From the experiments, we have identified that the factors affecting the system performance are the number of robots, the robots' starting positions and targets' positions, and the size of the robots. A larger number of small robots in the team will reduce the effect of robots' starting position on the system performance and yet yield better performance. This can be deduced from the graphs in Figure 7-10.

In almost all multi-robot works, increasing the number of robots in general will always improve the system performance. Thus, from the performance viewpoint, having more robots is good. However, there are associated costs with increasing the number of robots. Such as the robot's physical monetary value, power consumption, communications overhead, etc. These costs are usually not taken into consideration in accessing the benefit of adding more robots. This is because most of these costs are subjective parameters. For example, to access whether the monetary price of the robots is expensive, is subjective to the respective individual. Some of these costs may be specific to the particular system. If the relevant costs involved are put into consideration with improved system performance. There may be no added benefit in increasing the number of robots, though this may improve the system performance. The improvement may not always outweigh the costs involved.

We propose a simple function to evaluate the benefit of increasing the number of robots:

$$Benefit, B(n+1) = \frac{P_{n+1} - P_n}{P_n} - C ,$$

Where P_n is the performance of the current number of robots, P_{n+1} is the performance of the adding one robot to the team and C is a constant. The first term in the function is a dimensionless rate of change of the system performance. The constant term C is the cost of adding one robot to the team. In the equation, it is a constant because the cost of adding one more robot is subjective. Hence, it will be up to the human designer to decide on its value. This equation simply balances the performance improvement benefit with the incurred costs. Thus, as long as the value is positive and non-zero, there is benefit to add the additional robot.

Applying the equation to the results in Figure 7-5 and choosing C = 0, we obtained the benefits of increasing the number of robots in the first set-up. The results are illustrated in Figure 7-11. We chose C = 0 because in our multi-robot systems, the main cost incurred with adding more robots is the physical monetary value of our robot. Since our algorithm aimed to use simple inexpensive robots, we can ignore the cost and assigned it to 0. From the graph, the benefit of adding the eleventh robot is a negative value. It is the first instance where the benefit falls below zero. In this case, we should not add the eleventh robot and stopped here since there is a break in bringing benefit to the system with adding more robots. The target sensing range of the robot is 1 meter, while environment size is 4m by 4m in this set-up. Therefore, our results suggested that ten robots is the optimal team size in an environment approximately four times its target sensing range.

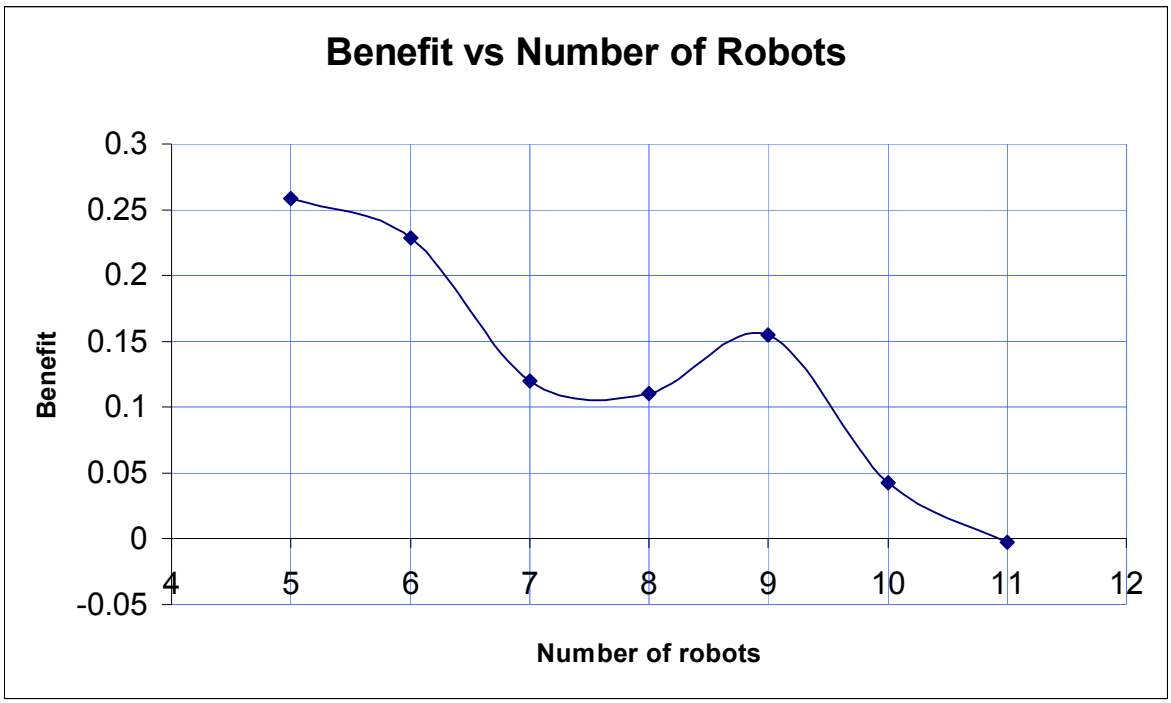


Figure 7-11: Benefit against number of robots

7.5 Chapter Summary

In this chapter, we showed that our proposed random search algorithm is able to solve the indoor search problem using a simulation test set-up. Following this, we demonstrated the algorithm on five physical CoSyBots with the same environment set-up. We found that the results from both simulation test and physical experiments are reasonably close. The mean time taken for the physical experiments is 249 seconds that is 0.31 standard deviation from the simulation test mean of 216 seconds. Hence, we could use the simulation program for experiments to estimate physical experiment results. We also changed the environment layout, robots' starting positions and targets' positions in the physical experiments. The robots found all targets for all the physical experiment runs. Hence, our proposed algorithm is robust to changes in the environment.

Further experiments are done using the simulation program. We varied the number of robots from four to twenty, changed the robots' starting positions and targets' positions, and double the size of the environment. A hundred simulation runs are repeated for each parameter change. We obtained some findings from these experiments. Firstly, we found that the system performance improves with the number of robots and reaches a point where there is no significant improvement. Secondly, the system performance varies with different robots' starting position. The difference decreases with increasing number of robots. Thirdly, the robots performed better in the bigger environment. This suggests that small robots work well with our proposed algorithm.

Finally, we proposed a benefit function to evaluate the benefit of increasing the number of robots. The benefit function takes into account the cost considerations in increasing the number of robots. Our results suggested that ten robots is the optimal team size in an environment approximately four times its target sensing range for the type of sensors used.

Chapter 8: Conclusions

8.1 Dissertation Conclusions

In this dissertation, we have designed a distributed random search algorithm that cooperates a team of simple autonomous robots to search for targets in an unknown indoor environment with multiple rooms. The multi-robot control architecture for our algorithm is distributed, homogeneous and local. This allows the robots to operate independently without a single central control, which is a potential point of failure. The algorithm consists of five simple behavioural rules and each robot has the same rule set.

We have demonstrated the effectiveness of the algorithm on physical robots. To implement the algorithm on physical robots, we formulated the five behavioural rules into five reactive behaviours: (1) Obstacle avoidance, (2) Target detection, (3) Respond to neighbour's message, (4) Follow external commands, and (5) Wander. Obstacle avoidance together with wander is responsible for autonomous navigation in the unknown environment. Using these two behaviours, the robot is able to autonomously avoid collision and also looks for openings to move into. Target detection is used to search for the targets. Respond to neighbour's messages ensures that one target is found by only one robot and also promotes local interactions among the robots. Follow external commands allow a commander to issue control commands to the robots. These behaviours are tested to work in our simulation program before implementing on the physical robots. In the physical experiments, we deployed five robots to search for three targets located in different rooms in a 4m by 4m structured

environment. We also varied the layout, robots' starting position and targets' position. The robots found all three targets in all the physical experiments. Therefore, the algorithm is robust to changes in the environment set-up.

To analyse the system performance of our algorithm, we performed multiple simulation experiments. We varied the number of robots from four to twenty, changed the robots' starting positions and targets' positions, and double the size of the environment. A hundred simulation runs are repeated for each parameter change. Some findings are obtained from these experiments. Firstly, we found that the system performance improves with the number of robots and reaches a point where there is no significant improvement. This also showed that our algorithm is scalable in numbers. Secondly, the system performance varies with different robots' starting position. The difference decreases with increasing number of robots. Thirdly, the robots performed better in the bigger environment. This suggests that small robots work well with our proposed algorithm. From these findings, we can conclude that using a larger number of small robots in the team will reduce the effect of robots' starting position on the system performance and yet yield better performance. However, it reaches a point where no significant improvement is achieved with adding more robots.

Finally, we formulated a benefit function that takes into account cost considerations to evaluate the benefit of increasing the number of robots. Using our benefit function, we found that the optimal number is ten robots for an environment that is four times its target sensing range for the type of sensors used. However, this waits to be verified by additional work, which is outside the scope of this dissertation.

8.2 Future Directions

There are a few possible improvements on this work.

1. It is difficult for the human designer to optimise the local reactive behaviours. The performance of these behaviours depends on the various parameters involved, such as the trigger distance. The process of selecting these parameters is usually tedious and may result in sub-optimal results. One possible improvement is to use the machine learning technology, such as genetic algorithms, to improve these behaviours, and hence, improving the performance of the algorithm.
2. The system performance is affected by various system parameters, such as the number of robots, environment layout and size, and the speed of robots. Currently, there are no means to relate these factors to the system performance. Having such a relationship is useful as the user could determine the number of robots he needs to deploy for his required system performance.
3. Further studies can also be conducted to examine the effect of robot self-localization on the system performance. With self-localization capabilities, the robot can share more information with fellow robots. This can reduce the problem of having a few robots searching the same area and robots revisiting explored areas.
4. It will be interesting to do a comparison study between the proposed algorithm and other deliberate approaches. In this study, factors such as demand on the robot's capabilities, cost, flexibility of the system and system performance may be considered.

Chapter 9: References

[1] R. C. Arkin, "Cooperation without communication: Multiagent schema-based robot navigation," Journal of Robotic Systems, 1992, pp 351-364.

[2] R. C. Arkin, and T. Balch, "Cooperative multiagent robotic systems," in Artificial Intelligence and Mobile Robots, D. Kortenkamp, R. P. Bonasso, R. Murphy (Eds.), MIT/AAAI Press, 1998.

[3] S. Aron, J. L. Deneubourg, S. Goss, and J. M. Pasteels, "Functional self-organization illustrated by inter-nest traffic in the Argentine ant Iridomyrmex humilis," In Bological Motion, edited by W. Alt and G. Hoffman, Berlin: Springer-Verlag, 1990, pp 533-547.

[4] D. Apostolopoulos, L. Pedersen, B. Shamah, K. Shillcutt, M. D. Wagner, and W. R. L. Whittaker, "Robotic antarctic meteorite search: Outcomes," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2001, pp 4174-4179.

[5] T. Balch and R. C. Arkin, "Communication in reactive multiagent robotic systems," Autonomous Robots, vol 1, 1994, pp 1-25.

[6] T. Balch, and R. C. Arkin, "Behavior-based formation control for multirobot teams," in IEEE Transactions on Robotics and Automation, vol 14, 1998, pp 926-939.

[7] D. Barnes, and J. Gray, "Behaviour synthesis for cooperant mobile robot control," In International Conference on Control, pages 1135–1140, 1991.

[8] G. Beni, "The concept of cellular robotic system," in IEEE International Symposium on Intelligent Control, 1988, pp 57-62.

[9] G. Beni, and J. Wang, "Swarm intelligence in cellular robotics systems," in Proceedings of NATO Advanced Workshop on Robots and Biological System, 1989.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm intelligence : from natural to artifical systems," Oxford University Press, 1999.

[11] J. Borenstein, and Y. Koren, "The vector field histogram – fast obstacle avoidance for mobile robots," IEEE Journal of Robotics and Automation, Vol 7, No 3, June 1991, pp 278-288.

[12] R. A. Brooks, "A robust layered control system for a mobile robot," IEEE Journal of Robotics and Autonmation, vol RA-2, no. 1, 1986, pp 14-23.

[13] W. Burgard, M. Moors, and F. Schneider, "Collaborative exploration of unknown environments with teams of mobile robots," Advances in Plan-Based Control of Robotic Agents, M. Beetz et al. (Eds), 2002, pp 52-70.

[14] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions", In Autonomous Robots 4, 1997, pp. 7-27.

[15] C. K. Cheng, G. Leng, "Cooperative search algorithm for distributed autonomous robots," Paper Accepted, to appear in Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2004.

[16] J. L. Crowley, "World modeling and position estimation for a mobile robot using ultrasonic ranging," in Proceedings of the IEEE International Conference on Robotics and Automation, 1989, pp 674-680.

[17] C. DeBolt, C. Freed, T. N. Nguyen, and T. B. Nguyen, "Basic UXO gathering system (BUGS); multiple, small, inexpensive robots for autonomous UXO clearance," UXO Forum Conference Proceedings, 1998.

[18] J. L. Deneubourg, S. Goss, N. R. Franks, and J.-M. Pasteels, "The blind leading the blind: Modelling chemically mediated army ant raid patterns," J. Insect Behavior 2, 1989, pp 719-725.

[19] J. L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels, "The self-organizing exploratory pattern of the Argentine ant," J. Insect Behavior 3, 1990, pp 159-168.

[20] A. Drogoul, and J. Ferber, "From tom thumb to the dockers: Some experiments with foraging robots," in 2nd International Conference on Simulation of Adaptative Behavior, 1992, pp 451-459.

[21] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for swarm robots," in Proceedings of IEEE International Conference on Intelligent Robots and Systems, 1993, pp. 441-447.

[22] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "Collaborative multi-robot localization," in Proceedings of the 23rd German Conference on Artificial Intelligence, Springer Verlag, 1999.

[23] D. W. Gage, "Command control for many-robot systems," Proceedings of AUVS-92, the Ninthteen AUVS Technical Symposium, 1992.

[24] D. W. Gage, "Randomized search strategies with imperfect sensors," in Proceedings of SPIE Mobile Robots VIII, vol 2058, 1993, pp 270-279.

[25] D. W. Gage, "Many-robot MCM search systems," in Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium, 1995.

[26] D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige, "Many robots make short work," AI Magazine, 18(1), pp 55-64, 1997.

[27] S. Hackwood, and G. Beni, "Self-organisation of sensors for swarm intellignce," in IEEE International Conference on Robotics and Automation, 1992, pp. 819–829.

[28] D. F. Hougen, S. Benjaafar, J. C. Bonney, J. R. Budenske, M. Dvorak, M. Gini, H. French, D. G. Krantz, P. Y. Li, F. Malver, B. Nelson, N. Papanikolopoulos, P. E. Ryski, S. A. Stoeter, R. Voyles, and K. B. Yesin, "A minature robotic system for reconnaissance and surveillance," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2000.

[29] A. Howard, M. J. Mataric, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," in Autonomous Robots, Vol 13, 2002, pp 113-126.

[30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in IEEE International Conference on Robotics and Automation, 1985, pp 500-505.

[31] B. H. Krogh, and C. E. Thorpe, "Integrated path planning and dynamic steering control for autonmous vehicles," in Proceedings of the IEEE International Conference on Robotics and Automation, 1986, pp 1664-1669.

[32] C. R. Kube, and E. Bonabeau, "Cooperative transport by ants and robots," in Robotics and Autonomous Systems, vol. 30, 2000, pp 85-101.

[33] R. Kuc, and B. Barshan, "Navigating vehicles through an unstructured environment with sonar," in Proceedings of the IEEE International Conference on Robotics and Automation, 1989, pp 1422-1426.

[34] T. Y. Li, H. C. Chou, "Motion planning for a crowd of robots," in Proceedings of the IEEE International Conference on Robotics and Automation, 2003, pp 4215-4221.

[35] A. Manz, R. Liscano, and D. A. Green, "A comparison of realtime obstacle avoidance methods for mobile robots," in Experimental Robotics, June 1991.

[36] A. Martinoli, and F. Mondada, "Collective and cooperative group behaviours: Biologically inspired experiments in robotics," In Proceedings of the Fourth International Symposium on Experimental Robotics (1995).

[37] M. J. Mataric, "Minimizing complexity in controlling a mobile robot population," in Proceedings of the IEEE International Conference on Robotics and Automation, 1992, pp 830-835.

[38] M. J. Mataric, "Interaction and intelligent behavior," PhD thesis, MIT, EECS, May 1994.

[39] M. J. Mataric, "Issues and approaches in the design of collective autonmous agents," in Robotics and Autonomous Systems, vol. 16, 1995, pp. 321-331.

[40] M. J. Mataric, "Behaviour-based control: examples from nagvigation, learning, and group behaviour," in J. Expt. Theor. Artif. Intell., vol. 9, 1997, pp 323-326.

[41] W. S. Newman, and N. Hogan, "High speed robot control and obstacle avoidance using dynamic potential functions," in Proceedings of the IEEE International Conference on Robotics and Automation, 1987, pp 14-24.

[42] D. J. Pack, and B. E. Mullins, "Towards finding an universal search algorithm for swarm robots," in Proceedings of IEEE International Conference on Intelligent Robots and Systems, 2003, pp. 1945-1950.

[43] L. E. Parker, "Multi-robot team design for real-world applications," in Distributed Autonomous Robotic Systems 2, edited by H. Asama, T. Fukuda, T. Aria and I. Endo, Springer-Verlag, Tokyo, 1996, 91-102.

[44] L. E. Parker, "Current state of the art in distributed autonomous mobile robotics," in Proceedings of the 4th International Symposium on Distributed Autonomous Robotic Systems (DARS), 2000, pp. 3-12.

[45] H. V. D. Parunak, "Go to the ant: engineering principles from natural multi-agent systems," in Annals of Operations Research, vol. 75, 1997, pp 69-101.

[46] D. Payton, M. Daily, R. Estkowski, M. Howard, and C. Lee, "Pheromone robotics," in Autonomous Robots, vol. 11, 2001, pp. 319–324.

[47] D. Payton, R. Estkowski, and M. Howard, "Progress in pheromone robotics," in Intelligent Autonomous Systems, vol. 7, M. Gini et al., Eds. IOS Press, 2002, pp. 256–264.

[48] S. Premvuti and S. Yuta, "Consideration on the cooperation of multiple autonomous mobile robots," In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 59–63, 1990.

[49] C. W. Reynolds, "Flocks, herds, and schools: A distibuted behavioral model," in Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings), 1987, pp 25-34.

[50] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. P. Papanikolopoulos, "Performance of a distributed robotic sytem using shared communications channels," in IEEE Transactions on Robotics and Automation, vol 18, 2002, pp 713-727.

[51] M. Schneider-Fontan, and M. J. Mataric, "Territorial multi-robot task division," in IEEE Transcations on Robotics and Automation, vol 14, no 5, 1998, pp 815-822.

[52] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Musliner, M.Pelican, S. Thrun, "Coordinated deployment of multiple

heterogeneous robots," in Proccedings of the Conference on Intelligent Robots and Systems (IROS), 2000.

[53] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershburger, A. Stentz, and R. Zlot, "A layered architecture for coordination of mobile robots," in Multi-Robot Systems: From Swarms to Intelligent Automata, A. Schultz and L. Parker (eds.), Kluwer, 2002.

[54] R. G. Smith, "The contract net protocol: high-level communication and control in a distributed problem solver," in IEEE Transactions on Computers, C-29 (12), 1980, pp 1104-1113.

[55] K. Sugawara, and M. Sano, "Cooperative acceleration of task performance: Foraging behavior of interacting mulit-robots system," Physica D, No. 100, 1996, 343-354.

[56] G. Theraulaz, and J. L. Deneubourg, "On formal constraints in swarm dynamics," in Proceedings of the IEEE International Symposium on Intelligent Control, 1992, pp 225-233.

[57] I. Ulrich, and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in Proceedings of the IEEE International Conference on Robotics and Automation, 1998, pp 1572-1577.

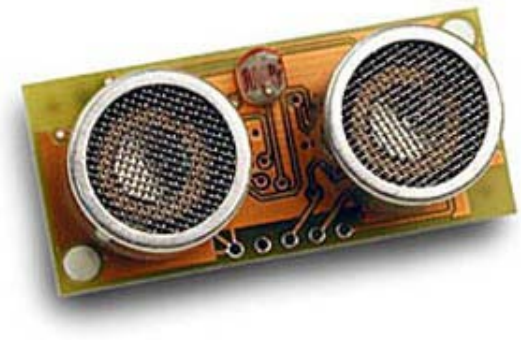
[58] I. Ulrich, and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification," in IEEE International Conference on Robotics and Automation, 2000, pp 2505-2511.

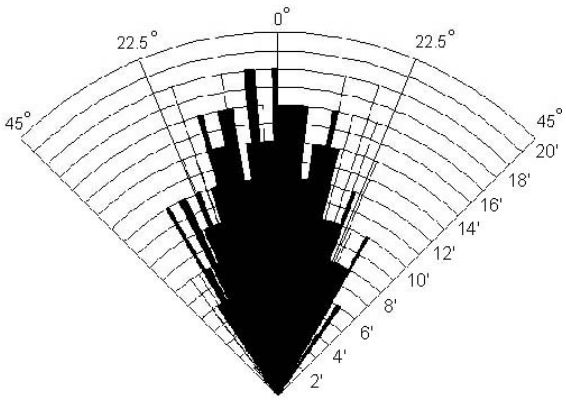
[59] I. A. Wagner, M, Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," in IEEE Transactions on Robotics and Automation, 1999, pp 918-933.

[60] C. R. Weisbin, G. de Saussure, and D. Kammer, "SELF-CONTROLLED. A real-time expert system for an autonomous mobile robot," Computers in Mechanical Engineering, 1986, pp 12-19.

[61] H. Yamaguchi, "A cooperative hunting behavior by mobile-robot troops," in The International Journal of Robotics Research, vol 18, 1999, pp 931-940.

Appendix A: Devantech SRF08 Sensor

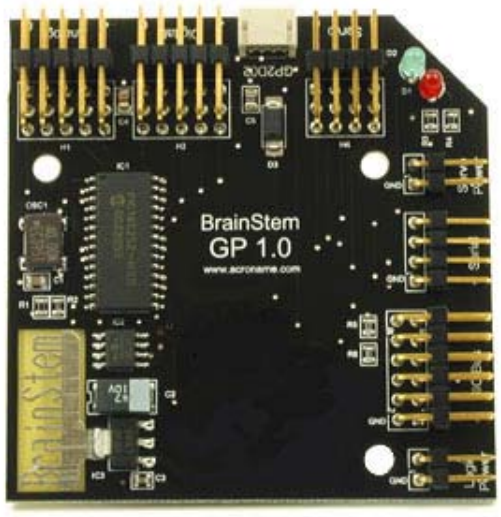




Technical Details:

Beam Pattern	See graph
Voltage	5v
Current	15mA Typ. 3mA Standby
Frequency	40KHz
Maximum Range	6 m
Minimum Range	3 cm
Max Analogue Gain	Variable to 1025 in 32 steps
Connection	Standard IIC Bus
Light Sensor	Front facing light sensor
Timing	Fully timed echo, freeing host computer of task
Echo	Multiple echo - keeps looking after first echo
Units	Range reported n uS, mm or inches
Weight	0.4 oz.
Size	43mm w x 20mm d x 17mm h

Appendix B: BrainStem GP 1.0



BrainStem GP 1.0 features:

- 40 MHz RISC processor
- 5 channel, 10 bit A/D
- 5 digital I/O lines
- GP2D02 Driver
- 1 MBit IIC port
- IIC routing
- Status LED
- Stores 11 1K TEA programs
- Runs up to 4 TEA programs concurrently
- RS-232 serial port
- Reflex architecture
- 4 high-resolution servo outputs
- Execution of 9000 instructions per second
- Access to I/O features via built-in serial command set
- Convenient power and ground connections for each I/O pin

Appendix C: SFR08 Experiments

CoSyBot 1								
Range / cm	Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 5	Sonar 6	Sonar 7	Sonar 8
3	3 – 4	4 – 5	3 – 4	3 – 4	3 – 4	4 – 5	3 – 4	4 – 5
25	23 – 25	25 – 26	25 - 27	23 – 25	25 – 27	24 – 26	25 – 27	25 – 27
50	49 – 52	50 – 52	48 - 51	50 - 53	50 – 52	51 – 53	49 – 50	49 – 51
100	99 – 103	99 – 102	98 - 101	100 – 103	99 – 102	100 – 103	100 – 102	99 – 103
125	119 – 121	121 – 123	120 – 122	121 – 123	119 – 121	120 – 123	121 – 124	120 – 122
150	119 – 121	121 – 123	120 – 122	121 – 123	119 – 121	120 – 123	121 –124	120 – 122

CoSyBot 2								
Range / cm	Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 5	Sonar 6	Sonar 7	Sonar 8
3	3 – 4	3 – 4	4 – 5	3 – 4	3 – 4	3 – 4	4 – 5	3 – 4
25	24 – 26	25 – 26	24 – 25	23 – 25	24 – 26	24 – 26	25 – 28	26 – 28
50	48 – 50	49 – 51	51 – 53	50 – 53	50 – 52	50 – 52	49 – 52	47 – 49
100	97 – 100	100 – 102	99 – 103	100 – 103	96 – 99	99 – 103	99 – 102	97 – 100
125	121 – 123	119 – 121	122 – 123	120 – 122	121 – 123	119 – 121	123 – 124	118 – 120
150	121 – 123	119 – 121	122 – 123	120 – 122	121 – 123	119 – 121	123 – 124	118 – 120

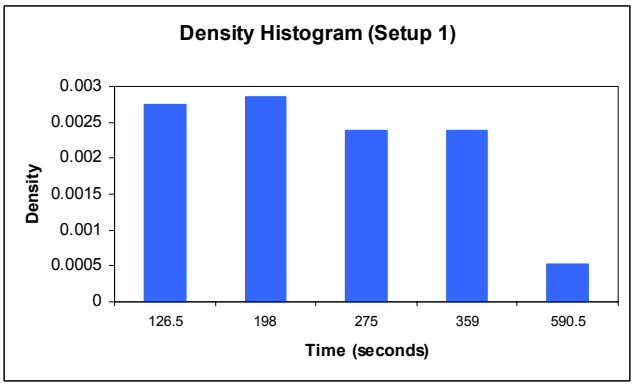
CoSyBot 3								
Range / cm	Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 5	Sonar 6	Sonar 7	Sonar 8
3	4 – 5	3 – 4	3 – 4	3 – 4	4 – 5	4 – 5	3 – 4	4 – 5
25	25 – 26	24 – 25	24 – 26	23 – 25	26 – 28	25 – 27	24 – 26	25 – 27
50	49 – 51	50 – 52	50 – 52	50 – 52	49 – 52	50 – 51	47 – 49	49 – 52
100	101 – 103	100 – 102	98 – 101	99 – 103	96 – 99	99 – 102	99 –103	99 – 103
125	120 – 122	119 – 121	118 – 120	121 – 122	122 – 124	120 – 123	119 – 120	121 – 124
150	120 – 122	119 – 121	118 – 120	121 – 122	122 – 124	120 – 123	119 – 120	121 – 124

CoSyBot 4								
Range / cm	Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 5	Sonar 6	Sonar 7	Sonar 8
3	3 – 4	4 – 5	3 – 4	4 – 5	3 – 4	3 – 4	3 – 4	4 – 5
25	23 – 25	25 – 26	24 – 26	25 – 26	23 – 25	23 – 24	23 – 25	26 – 28
50	49 – 52	50 – 53	48 – 50	50 – 53	49 – 51	47 – 49	48 – 50	50 – 52
100	97 – 101	99 – 103	98 – 101	99 – 103	96 – 100	96 – 100	98 – 102	99 – 103
125	119 – 120	121 – 123	118 – 120	121 – 123	120 – 124	119 – 121	120 – 122	122 – 124
150	119 – 120	121 – 123	118 – 120	121 – 123	120 – 124	119 – 121	120 – 122	122 – 124

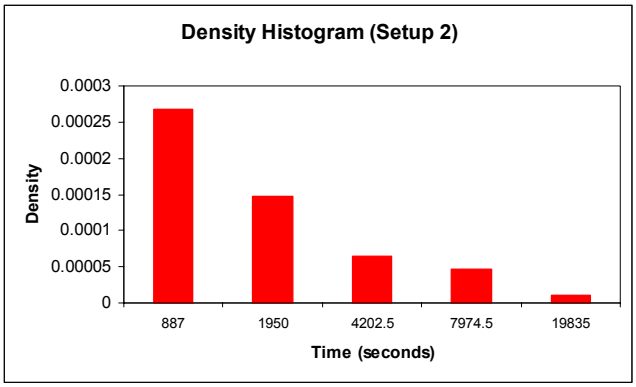
CoSyBot 5								
Range / cm	Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 5	Sonar 6	Sonar 7	Sonar 8
3	3 – 4	3 – 4	3 – 4	4 – 5	3 – 4	4 – 5	3 – 4	4 – 5
25	23 – 25	24 – 26	24 – 25	25 – 28	24 – 26	25 – 28	24 – 26	25 – 28
50	49 – 53	48 – 51	49 – 52	50 – 54	50 – 52	51 – 55	49 – 52	49 – 52
100	97 – 101	97 – 100	98 – 102	99 – 104	99 – 103	100 – 104	99 –103	98 – 103
125	119 – 121	118 – 120	121 – 123	123 – 124	121 – 122	123 – 125	121 – 123	120 – 122
150	119 – 121	118 – 120	121 – 123	123 – 124	121 – 122	123 – 125	121 – 123	120 – 122

Appendix D: Simulation Results

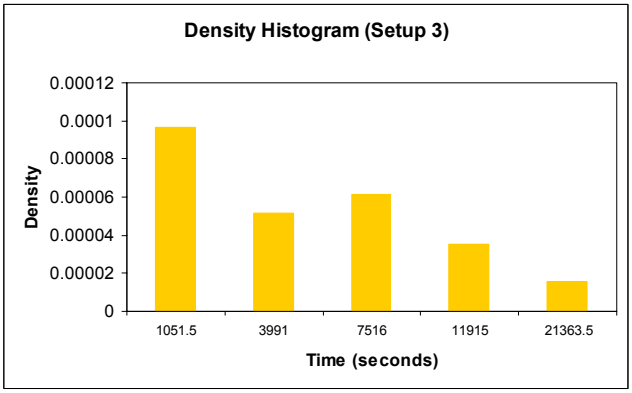
4 robots in 4m by 4m environment



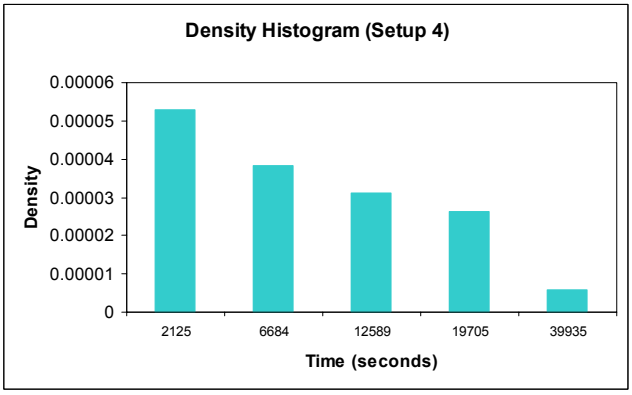
Set-up 1	
Mean / s	291
Std. dev. / s	138



Set-up 2	
Mean / s	6150
Std. dev. / s	6148

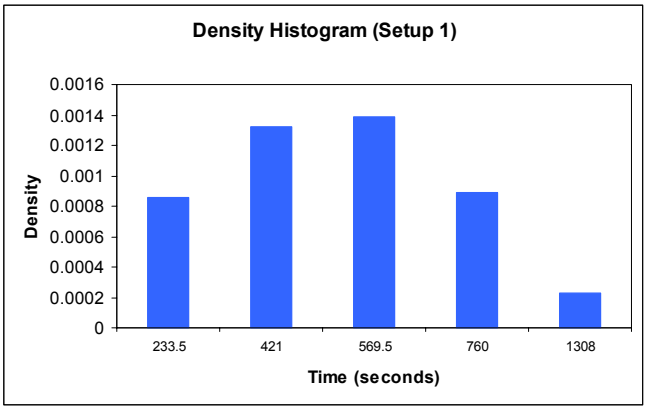


Set-up 3	
Mean / s	9157
Std. dev. / s	7110

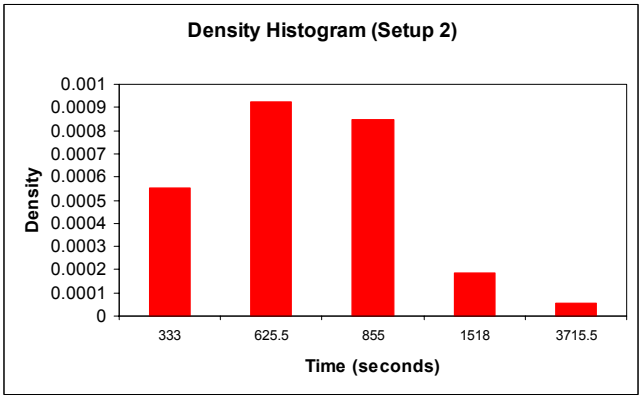


Set-up 4	
Mean / s	14718
Std. dev. / s	10484

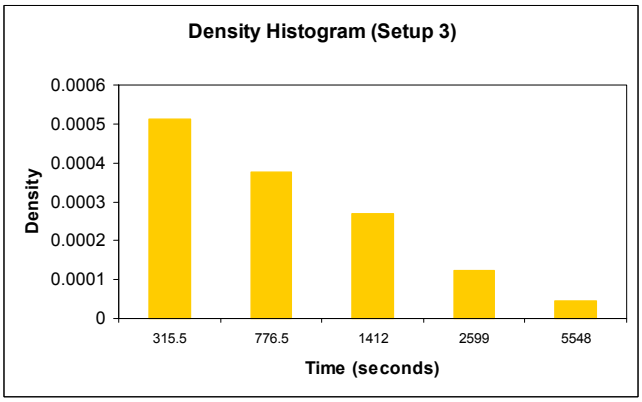
4 robots in 8m by 8m environment



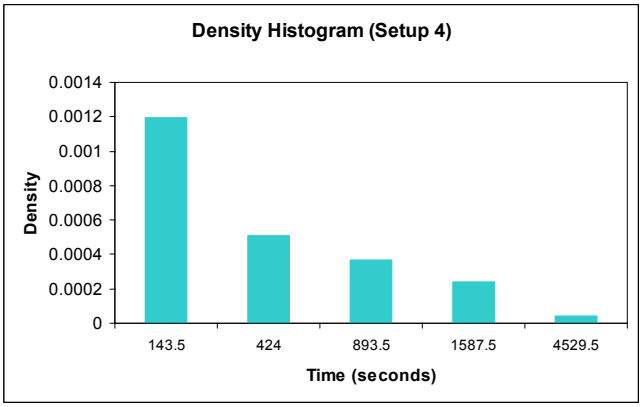
Set-up 1	
Mean / s	638
Std. dev. / s	293



Set-up 2	
Mean / s	1300
Std. dev. / s	946

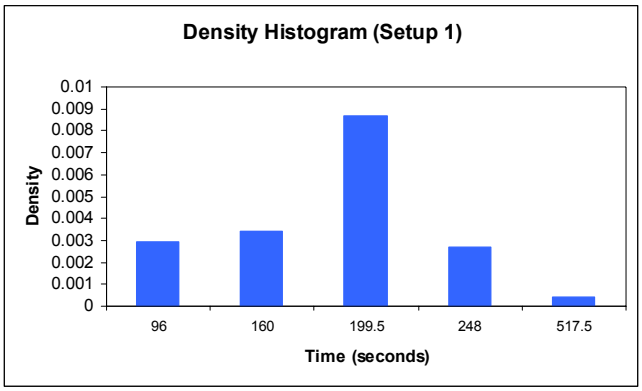


Set-up 3	
Mean / s	2061
Std. dev. / s	1957

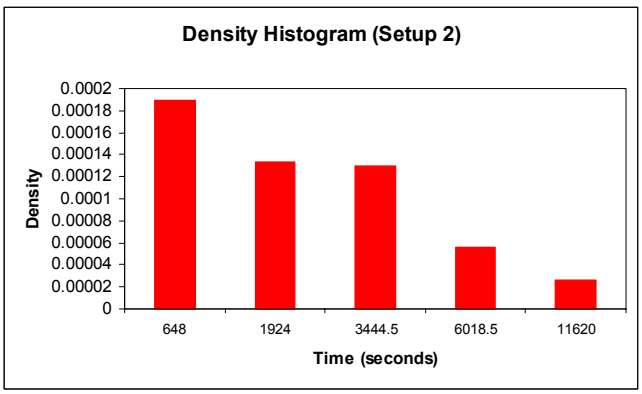


Set-up 4	
Mean / s	1243
Std. dev. / s	1072

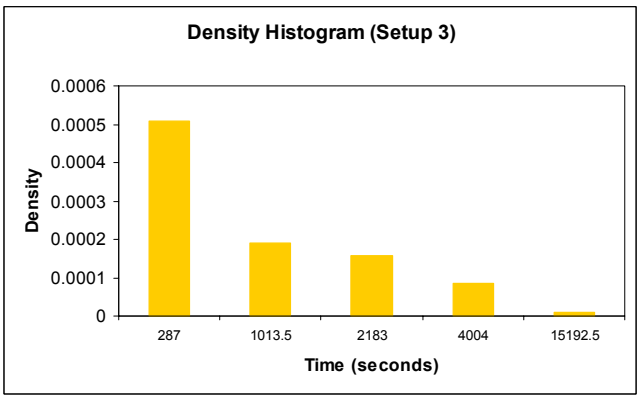
5 robots in 4m by 4m environment



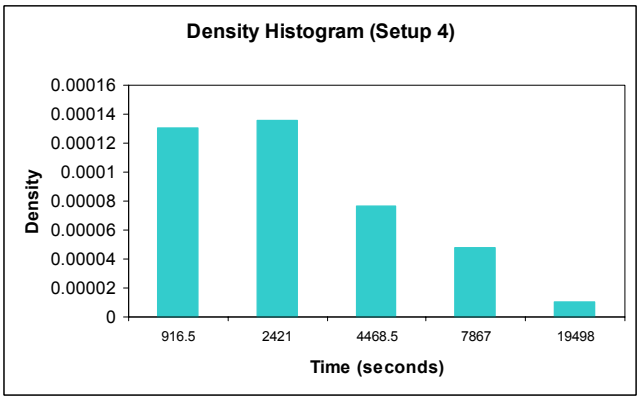
Set-up 1	
Mean / s	216
Std. dev. / s	105



Set-up 2	
Mean / s	4414
Std. dev. / s	3676

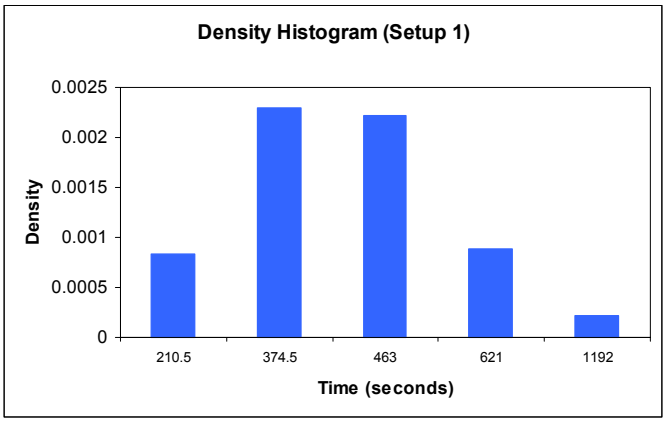


Set-up 3	
Mean / s	3390
Std. dev. / s	4091

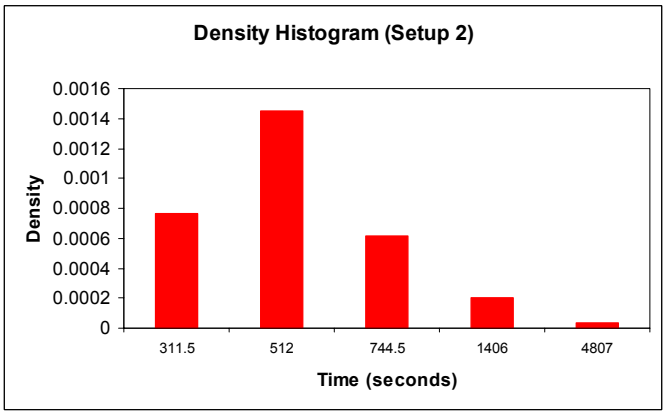


Set-up 4	
Mean / s	6286
Std. dev. / s	5941

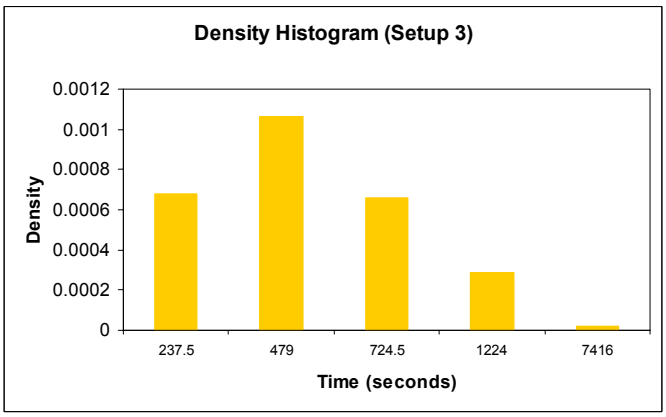
5 robots in 8m by 8m environment



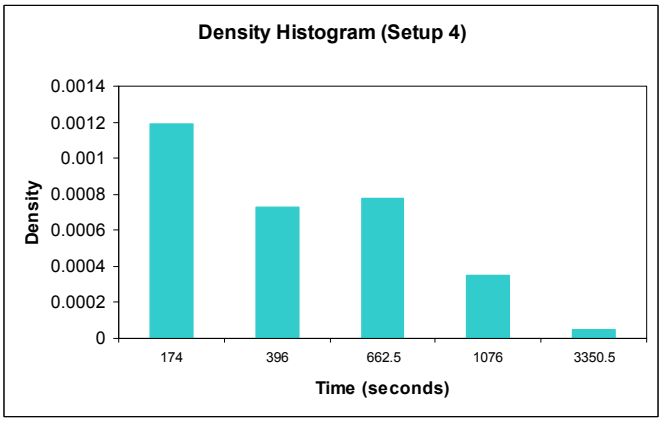
Set-up 1	
Mean / s	551
Std. dev. / s	301



Set-up 2	
Mean / s	1300
Std. dev. / s	1365

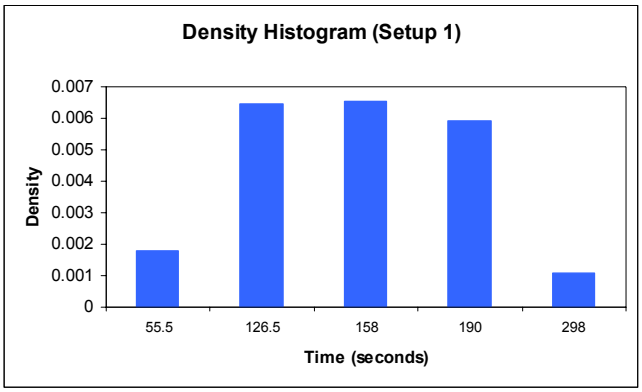


Set-up 3	
Mean / s	1203
Std. dev. / s	1567

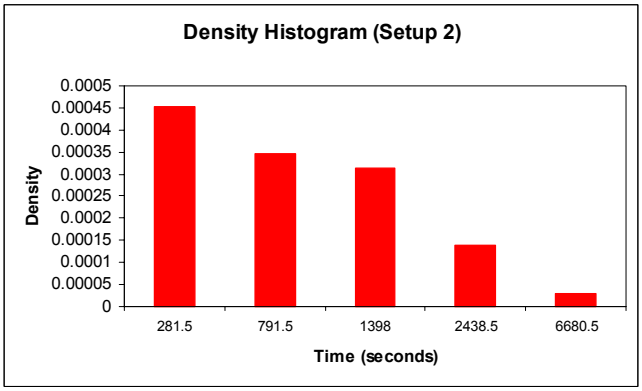


Set-up 4	
Mean / s	928
Std. dev. / s	902

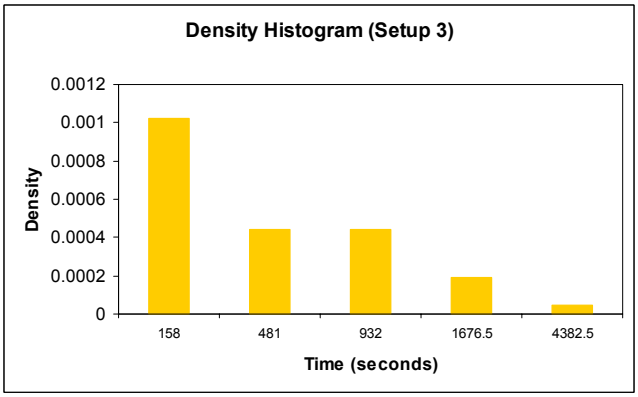
6 robots in 4m by 4m environment



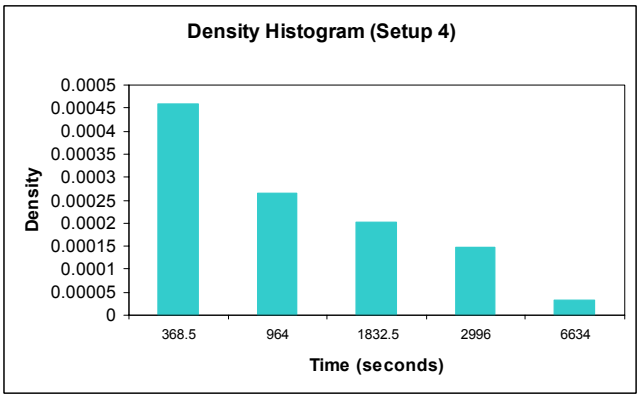
Set-up 1	
Mean / s	167
Std. dev. / s	66



Set-up 2	
Mean / s	1940
Std. dev. / s	1859

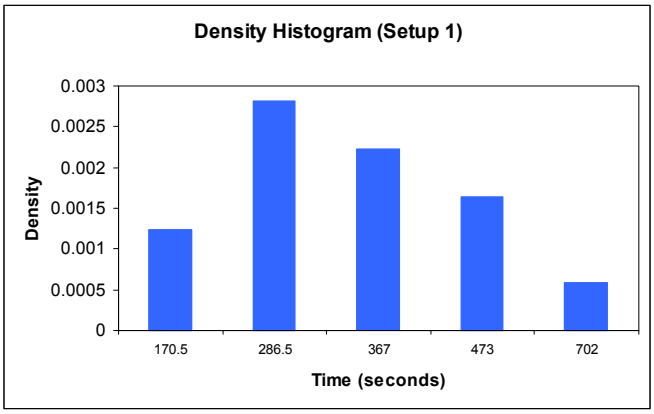


Set-up 3	
Mean / s	1430
Std. dev. / s	1471

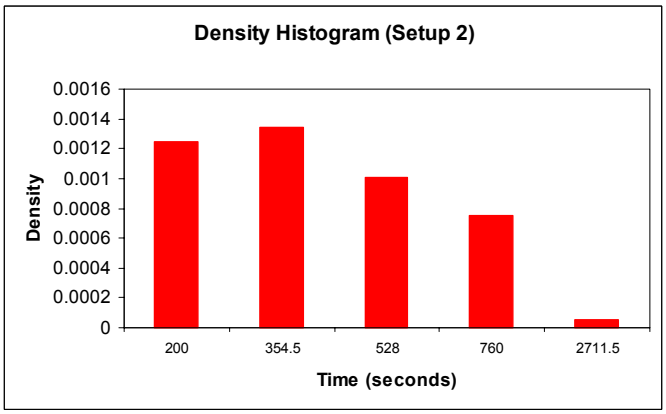


Set-up 4	
Mean / s	2389
Std. dev. / s	2235

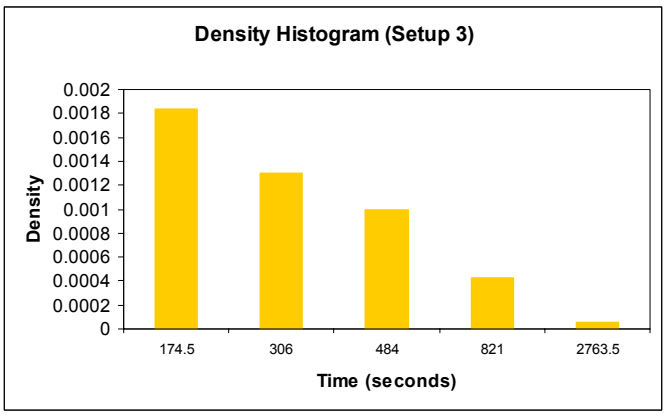
6 robots in 8m by 8m environment



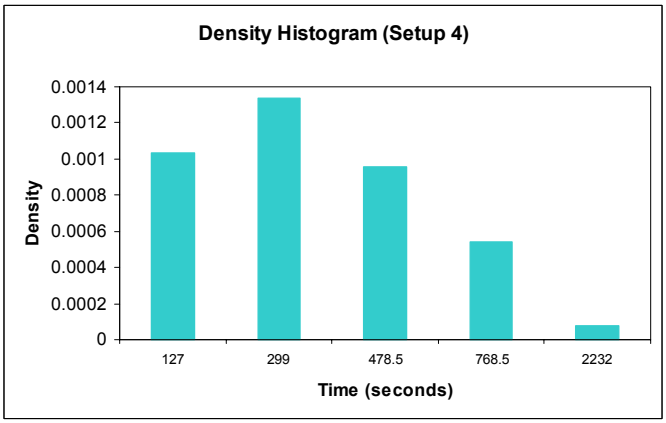
Set-up 1	
Mean / s	395
Std. dev. / s	167



Set-up 2	
Mean / s	713
Std. dev. / s	683

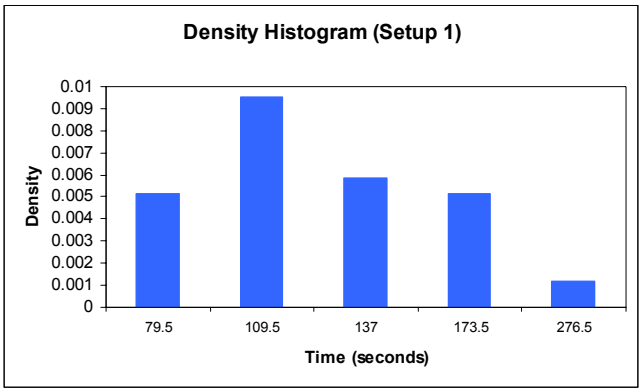


Set-up 3	
Mean / s	685
Std. dev. / s	689

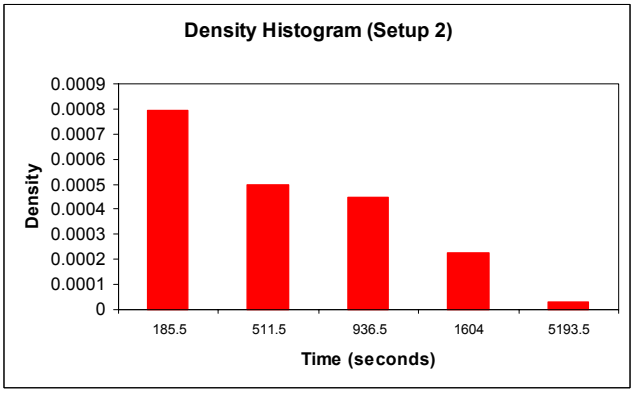


Set-up 4	
Mean / s	644
Std. dev. / s	595

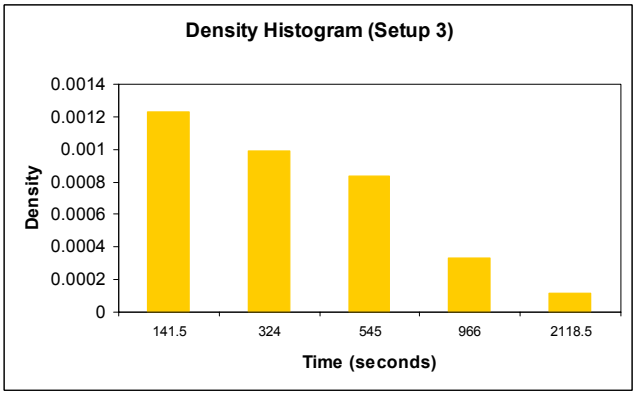
7 robots in 4m by 4m environment



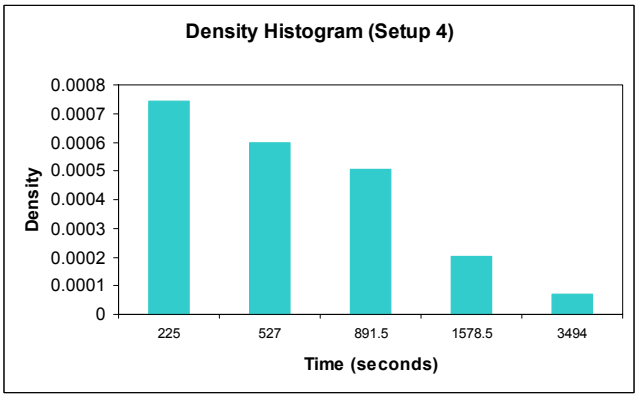
Set-up 1	
Mean / s	147
Std. dev. / s	54



Set-up 2	
Mean / s	1371
Std. dev. / s	1473

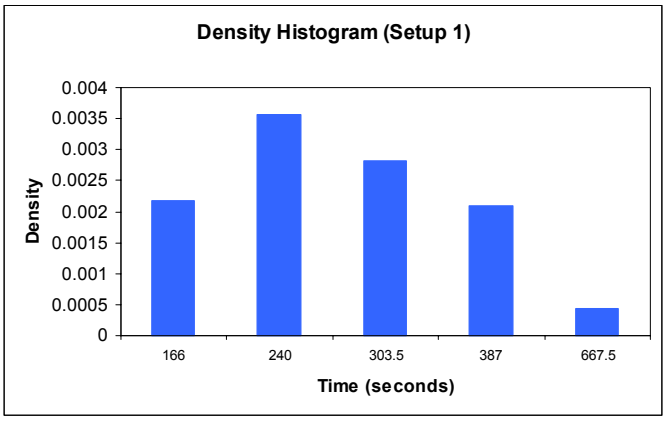


Set-up 3	
Mean / s	772
Std. dev. / s	645

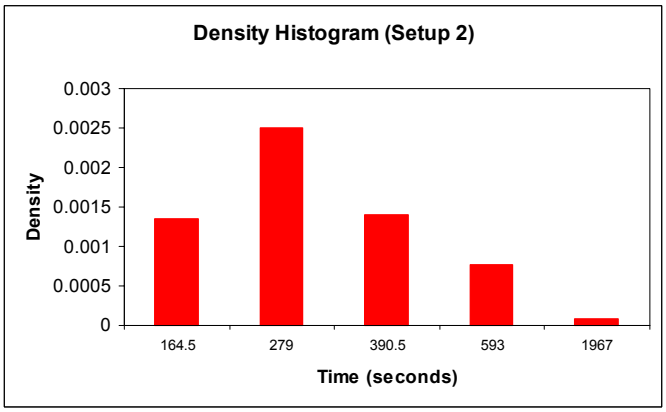


Set-up 4	
Mean / s	1153
Std. dev. / s	913

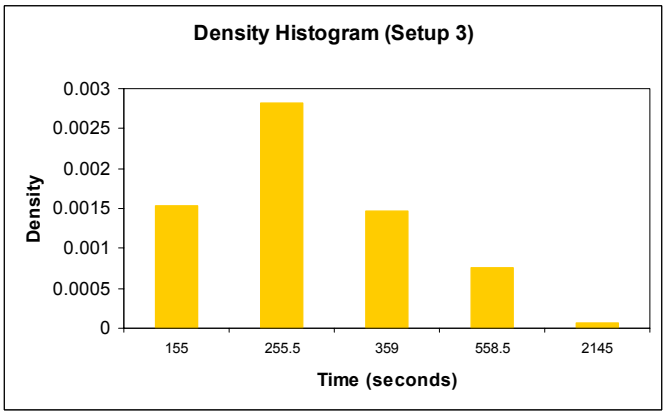
7 robots in 8m by 8m environment



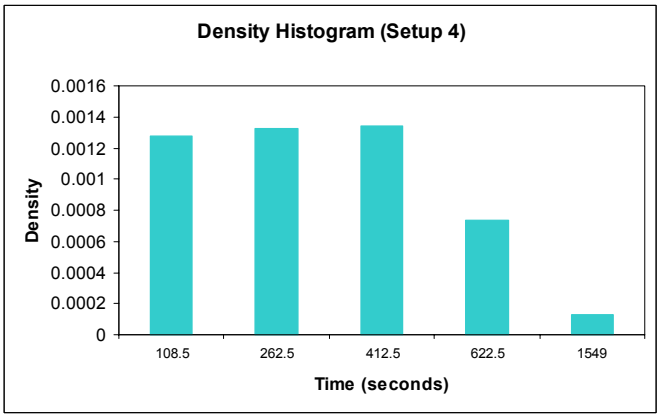
Set-up 1	
Mean / s	337
Std. dev. / s	146



Set-up 2	
Mean / s	562
Std. dev. / s	492

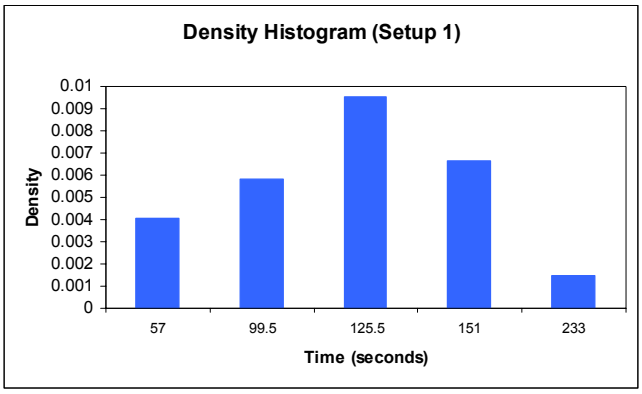


Set-up 3	
Mean / s	499
Std. dev. / s	517

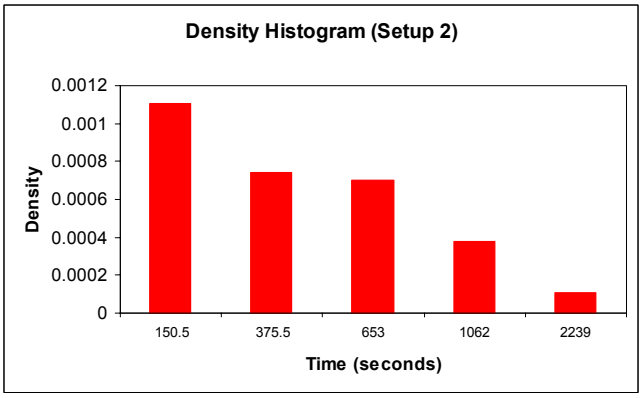


Set-up 4	
Mean / s	509
Std. dev. / s	413

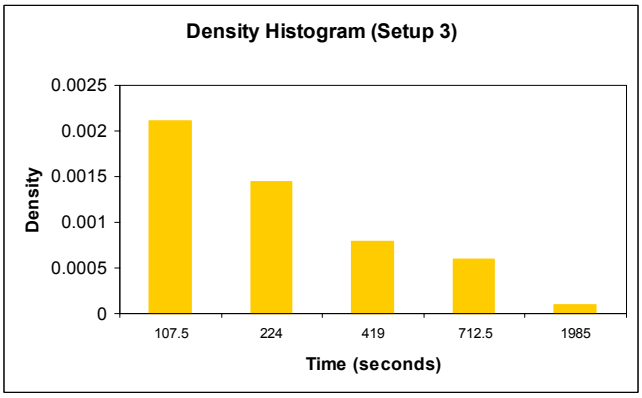
8 robots in 4m by 4m environment



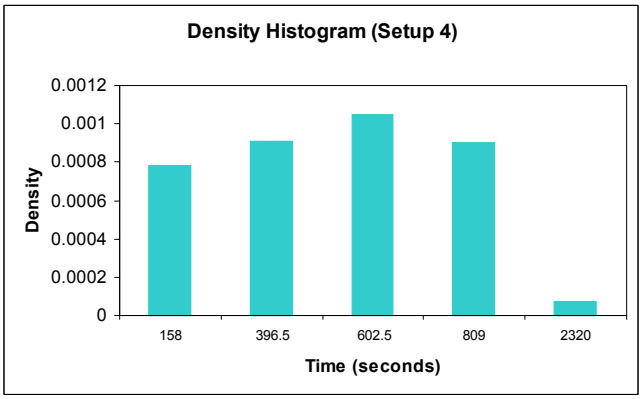
Set-up 1	
Mean / s	131
Std. dev. / s	51



Set-up 2	
Mean / s	802
Std. dev. / s	638

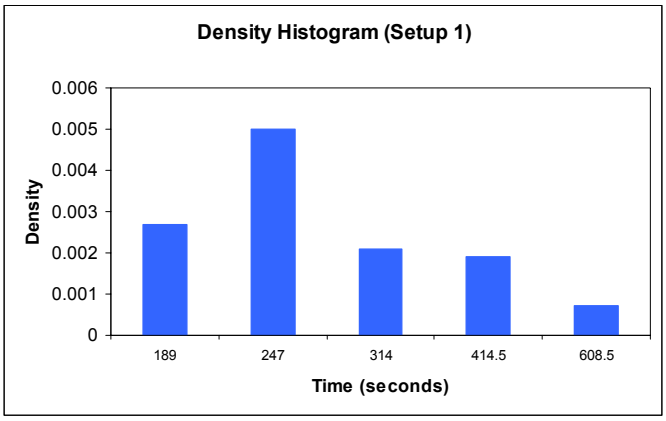


Set-up 3	
Mean / s	597
Std. dev. / s	607

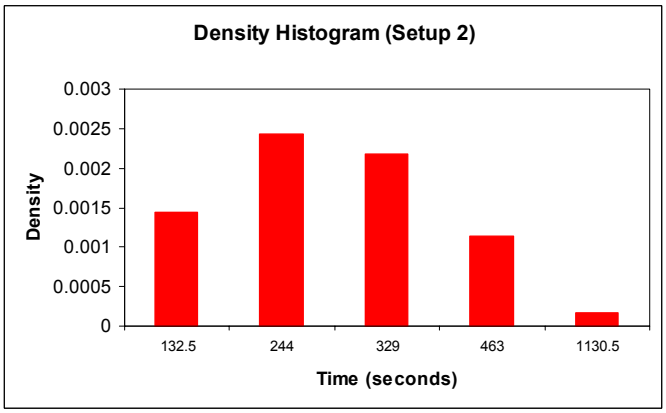


Set-up 4	
Mean / s	687
Std. dev. / s	536

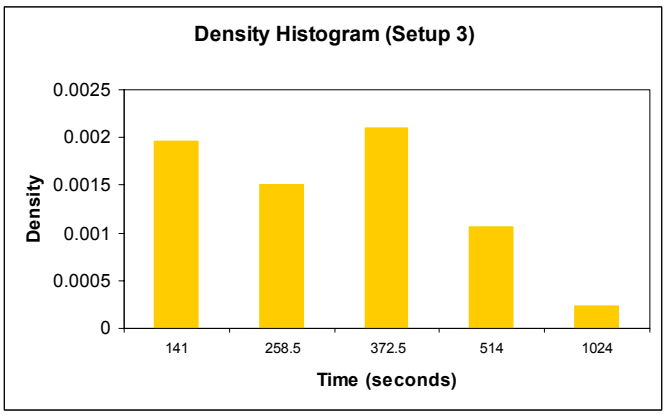
8 robots in 8m by 8m environment



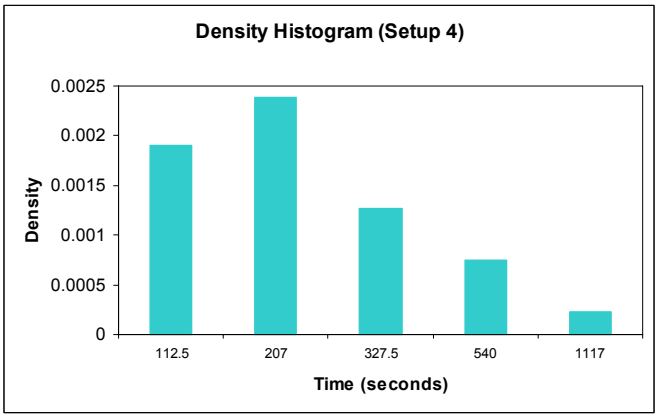
Set-up 1	
Mean / s	346
Std. dev. / s	138



Set-up 2	
Mean / s	430
Std. dev. / s	332

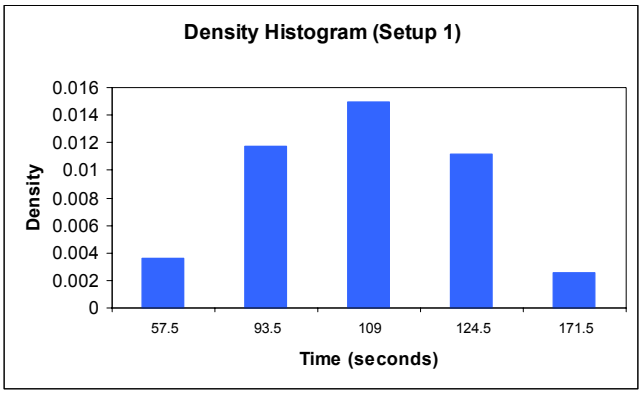


Set-up 3	
Mean / s	424
Std. dev. / s	267

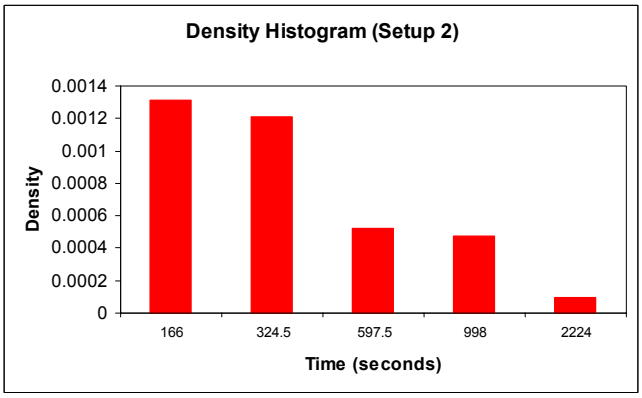


Set-up 4	
Mean / s	435
Std. dev. / s	342

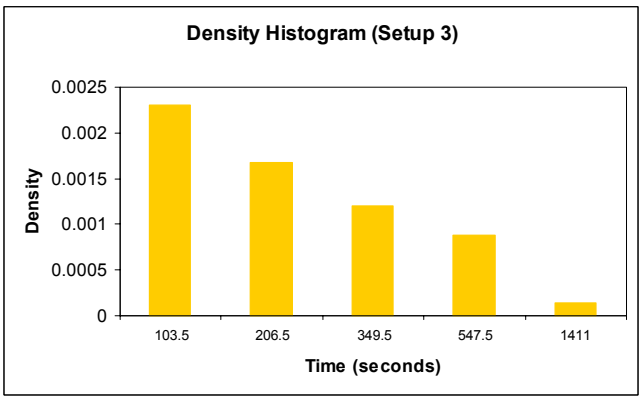
9 robots in 4m by 4m environment



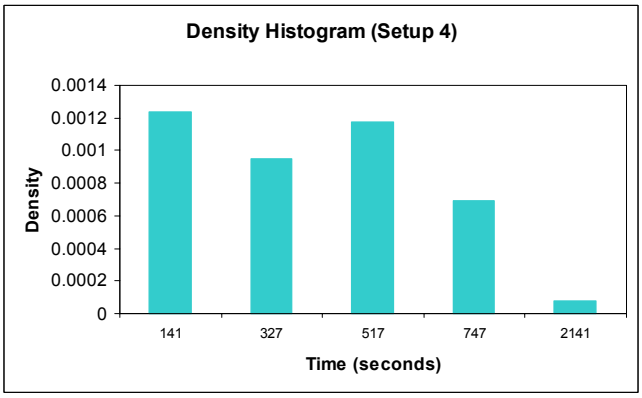
Set-up 1	
Mean / s	110
Std. dev. / s	32



Set-up 2	
Mean / s	741
Std. dev. / s	606

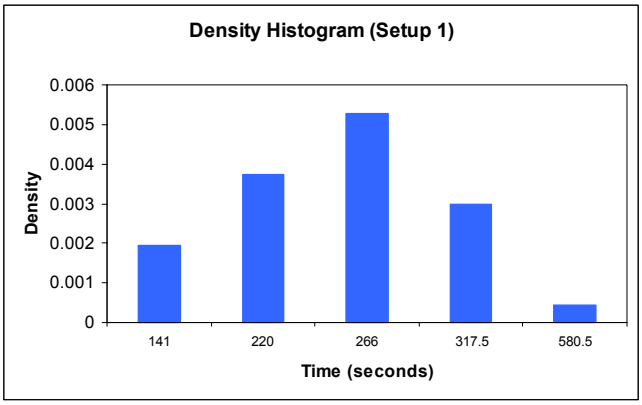


Set-up 3	
Mean / s	455
Std. dev. / s	382

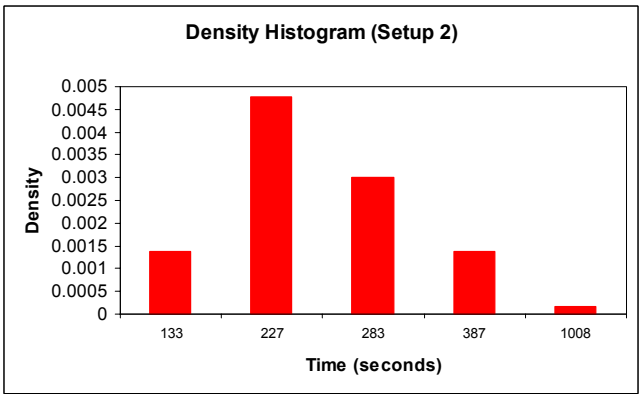


Set-up 4	
Mean / s	655
Std. dev. / s	594

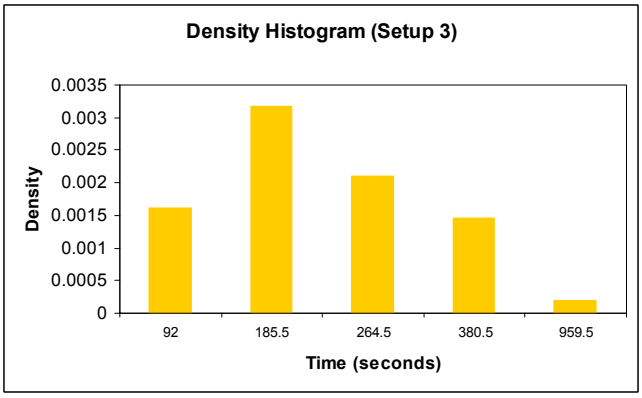
9 robots in 8m by 8m environment



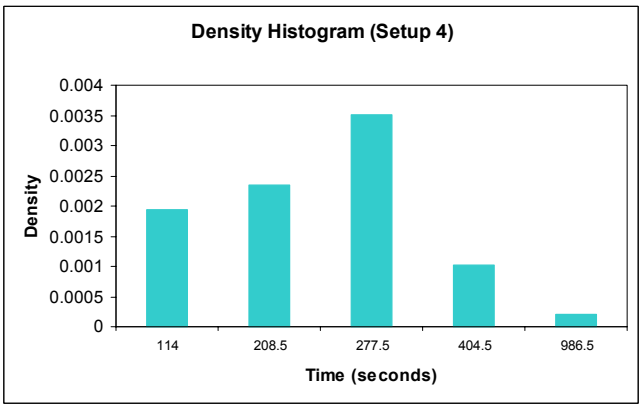
Set-up 1	
Mean / s	284
Std. dev. / s	114



Set-up 2	
Mean / s	363
Std. dev. / s	267

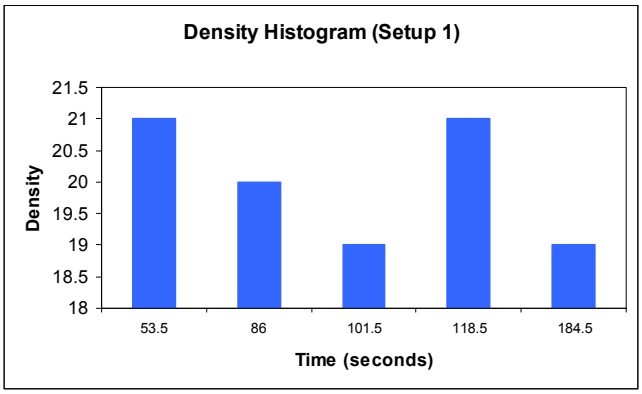


Set-up 3	
Mean / s	324
Std. dev. / s	236

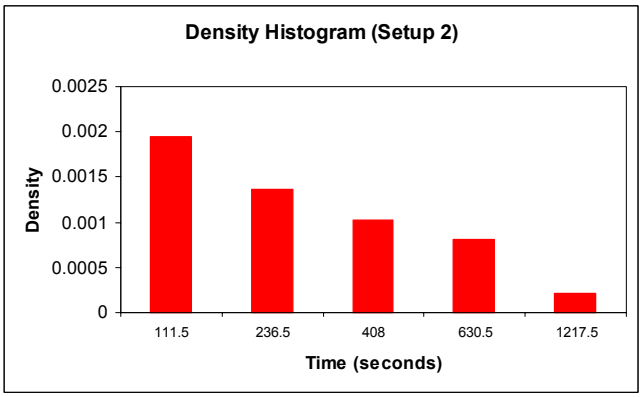


Set-up 4	
Mean / s	341
Std. dev. / s	243

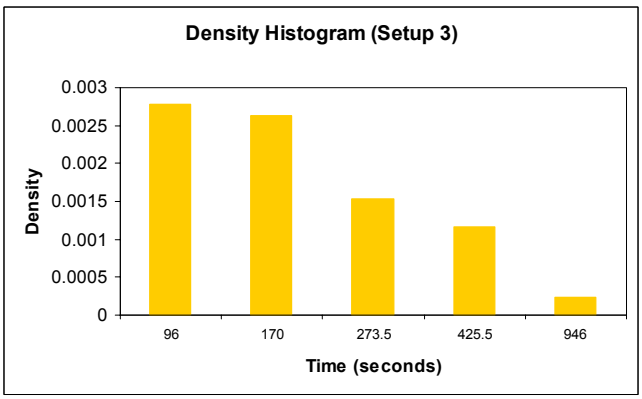
10 robots in 4m by 4m environment



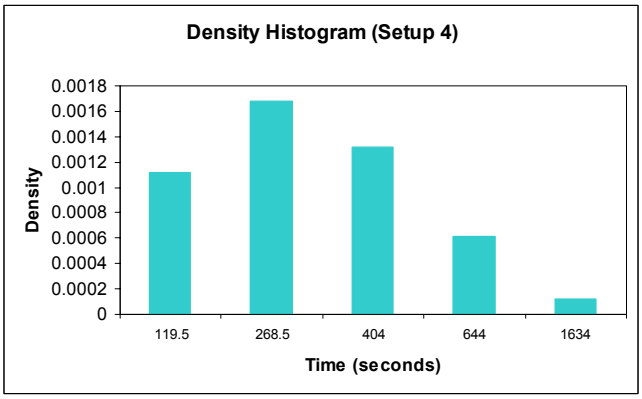
Set-up 1	
Mean / s	106
Std. dev. / s	34



Set-up 2	
Mean / s	488
Std. dev. / s	364

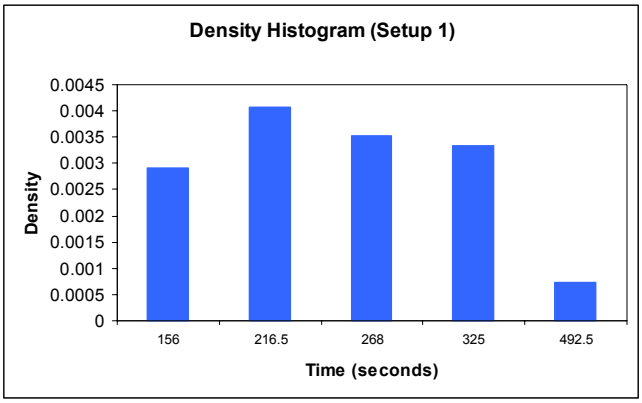


Set-up 3	
Mean / s	332
Std. dev. / s	246

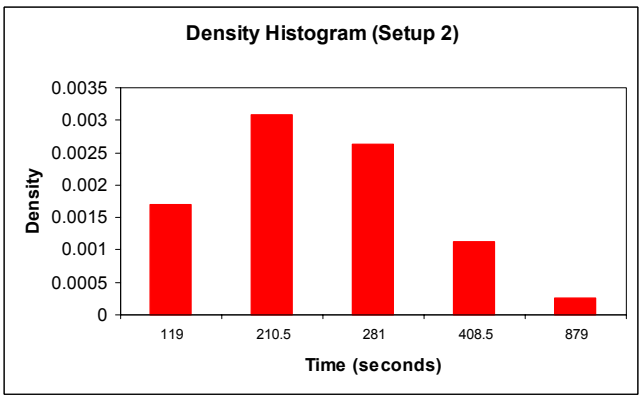


Set-up 4	
Mean / s	524
Std. dev. / s	413

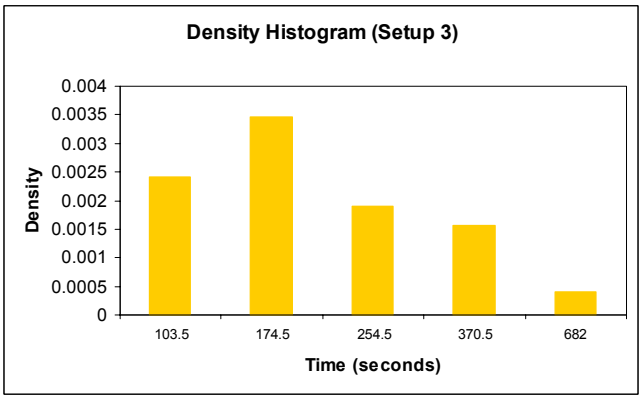
10 robots in 8m by 8m environment



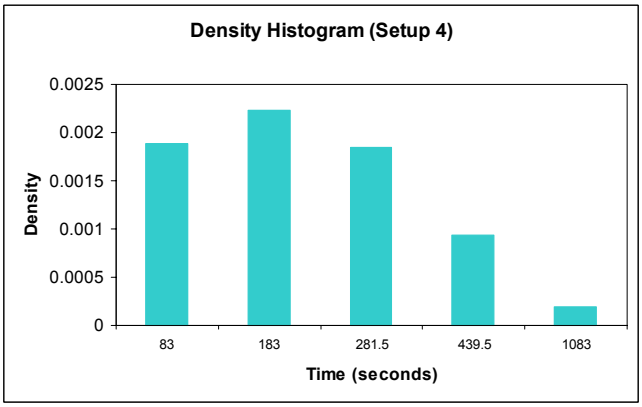
Set-up 1	
Mean / s	283
Std. dev. / s	104



Set-up 2	
Mean / s	367
Std. dev. / s	261

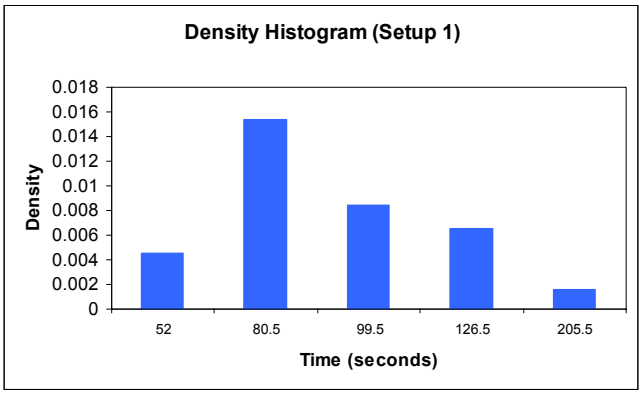


Set-up 3	
Mean / s	294
Std. dev. / s	179

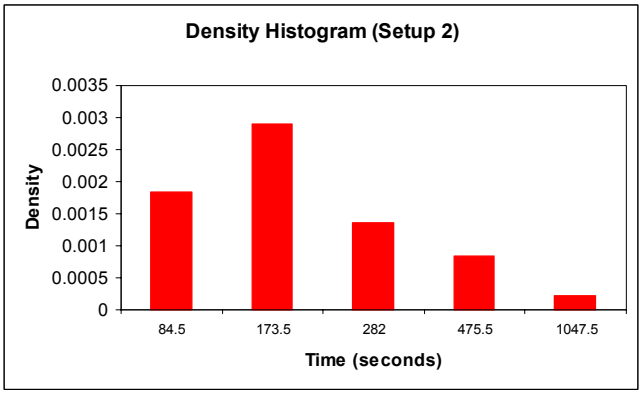


Set-up 4	
Mean / s	365
Std. dev. / s	288

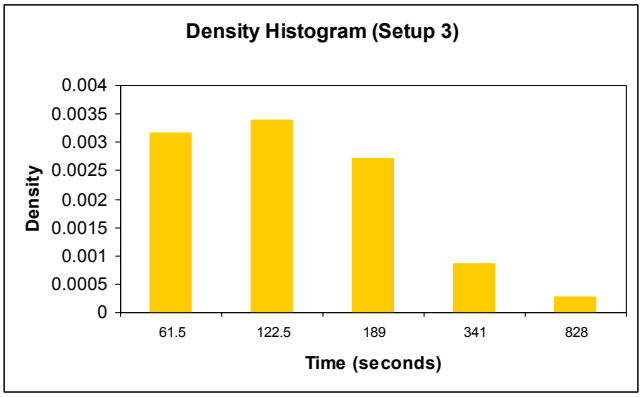
11 robots in 4m by 4m environment



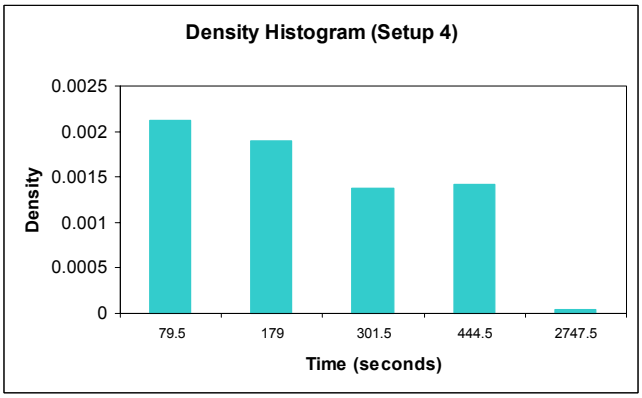
Set-up 1	
Mean / s	106
Std. dev. / s	38



Set-up 2	
Mean / s	372
Std. dev. / s	296

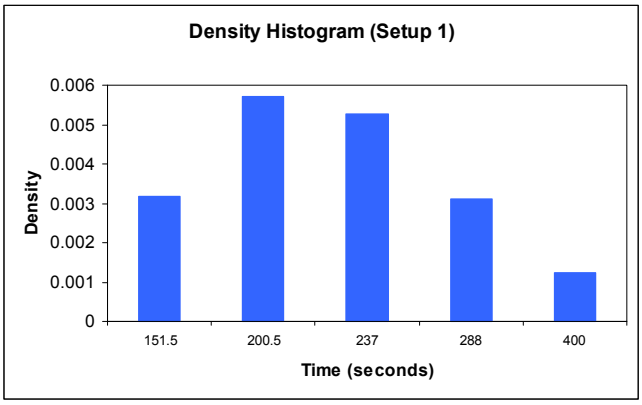


Set-up 3	
Mean / s	281
Std. dev. / s	245

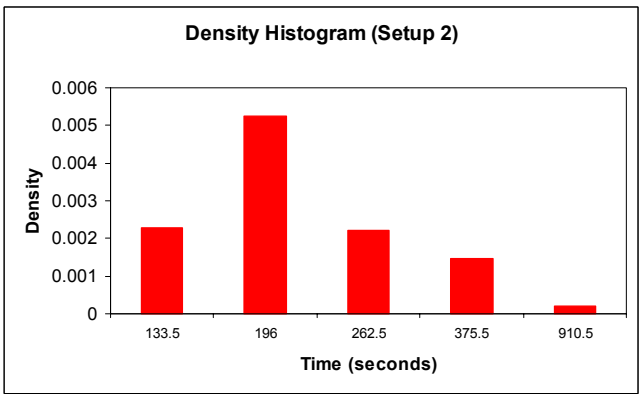


Set-up 4	
Mean / s	403
Std. dev. / s	537

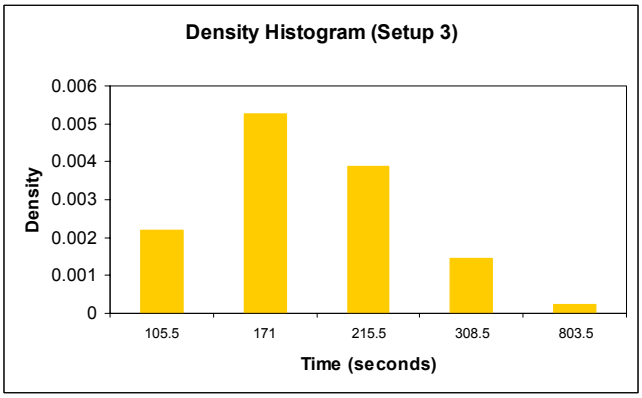
11 robots in 8m by 8m environment



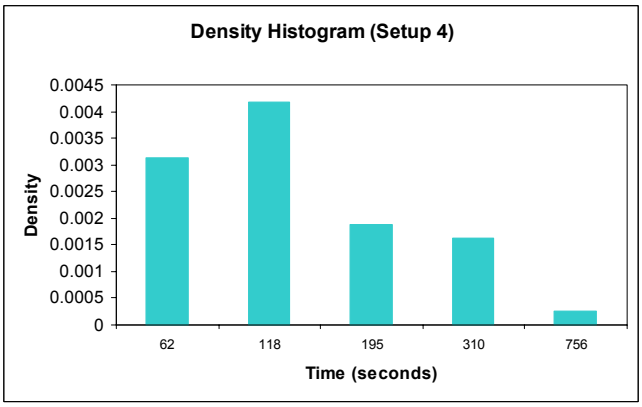
Set-up 1	
Mean / s	252
Std. dev. / s	76



Set-up 2	
Mean / s	334
Std. dev. / s	236

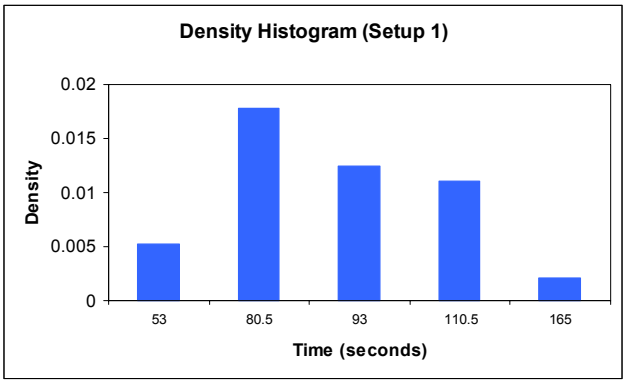


Set-up 3	
Mean / s	271
Std. dev. / s	173

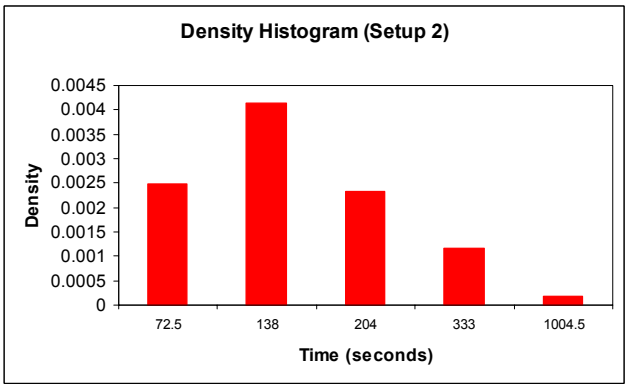


Set-up 4	
Mean / s	250
Std. dev. / s	199

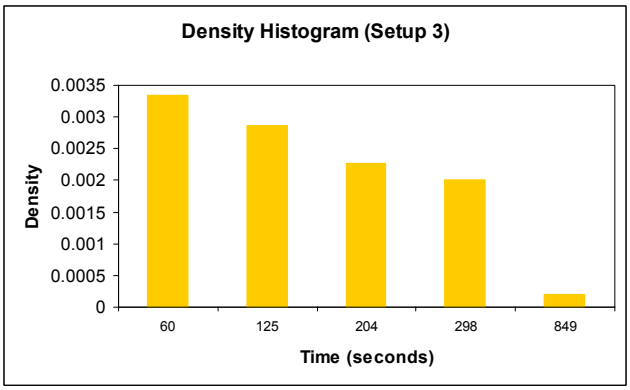
12 robots in 4m by 4m environment



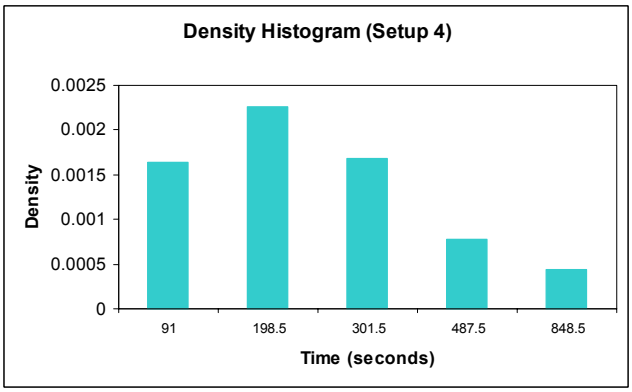
Set-up 1	
Mean / s	97
Std. dev. / s	29



Set-up 2	
Mean / s	277
Std. dev. / s	229

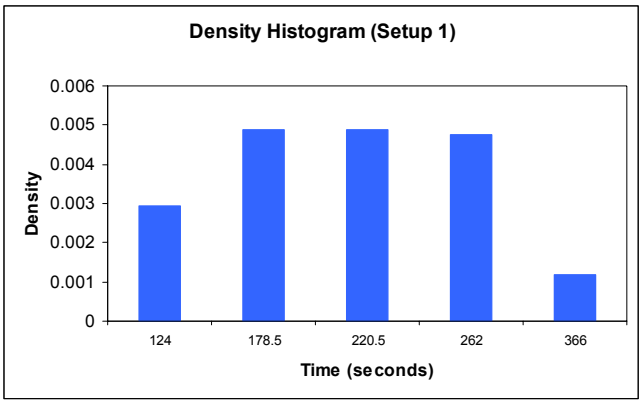


Set-up 3	
Mean / s	247
Std. dev. / s	207

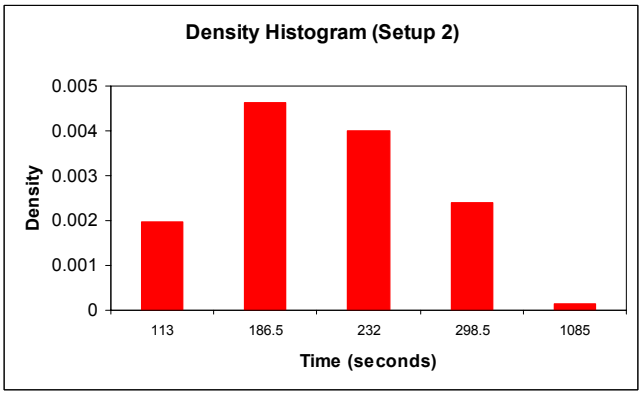


Set-up 4	
Mean / s	380
Std. dev. / s	271

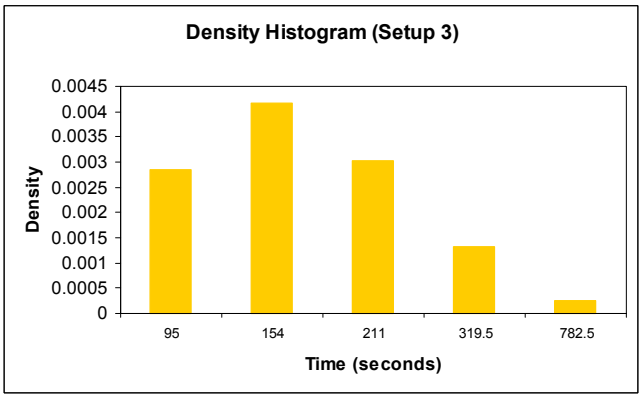
12 robots in 8m by 8m environment



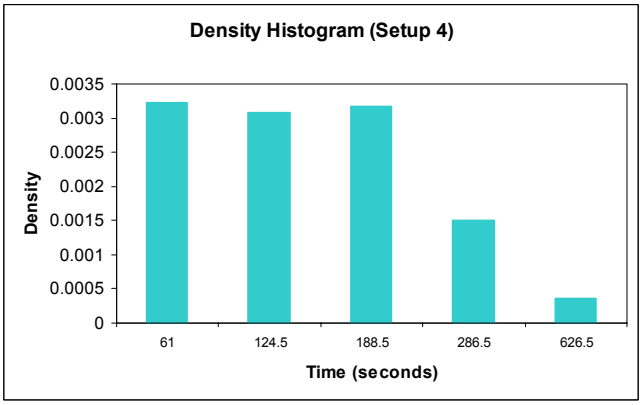
Set-up 1	
Mean / s	225
Std. dev. / s	72



Set-up 2	
Mean / s	284
Std. dev. / s	217

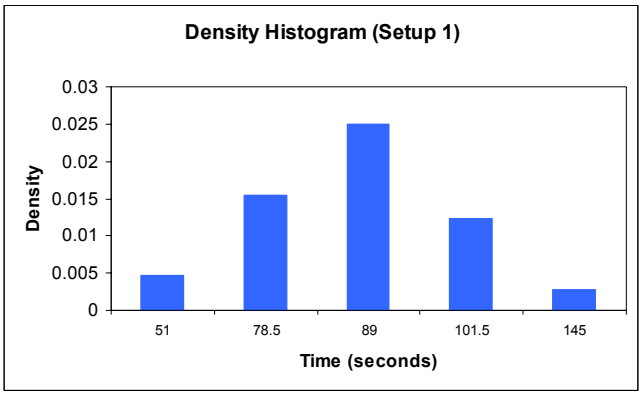


Set-up 3	
Mean / s	278
Std. dev. / s	203

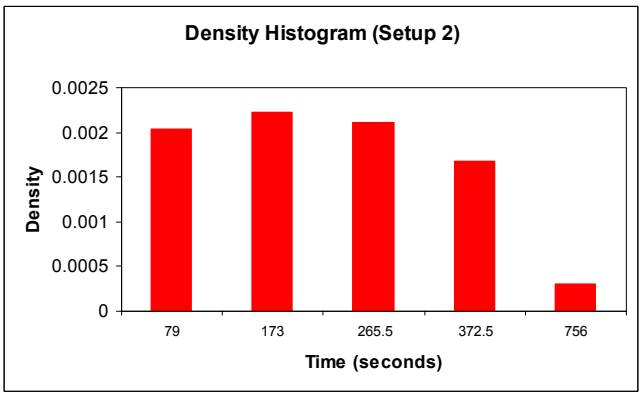


Set-up 4	
Mean / s	237
Std. dev. / s	177

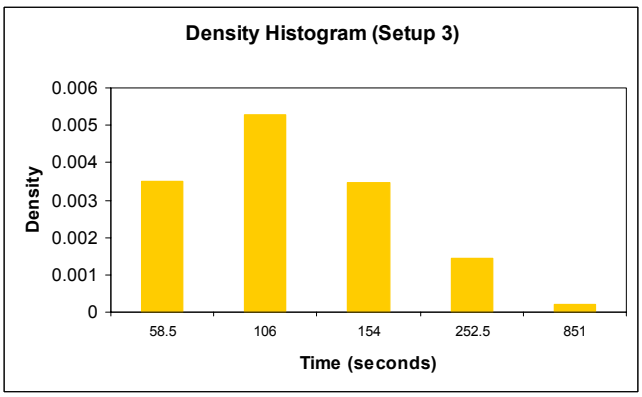
13 robots in 4m by 4m environment



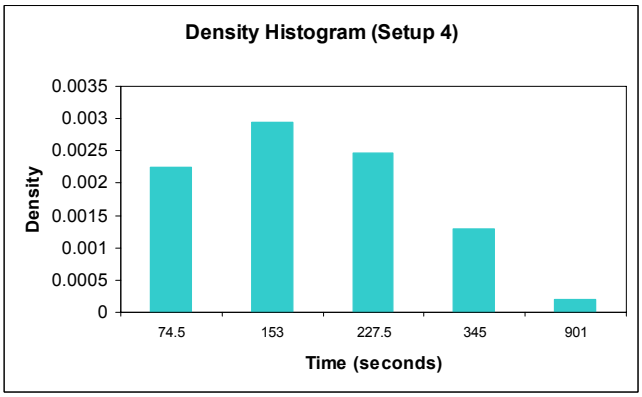
Set-up 1	
Mean / s	91
Std. dev. / s	25



Set-up 2	
Mean / s	305
Std. dev. / s	199

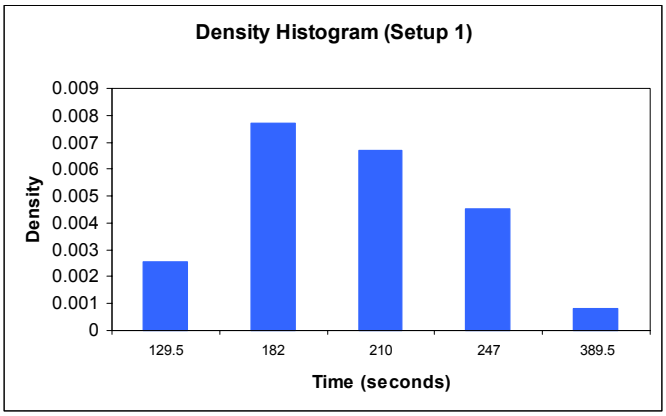


Set-up 3	
Mean / s	219
Std. dev. / s	217

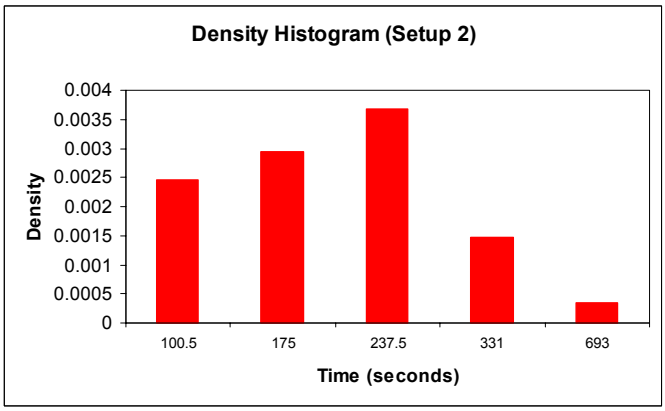


Set-up 4	
Mean / s	285
Std. dev. / s	232

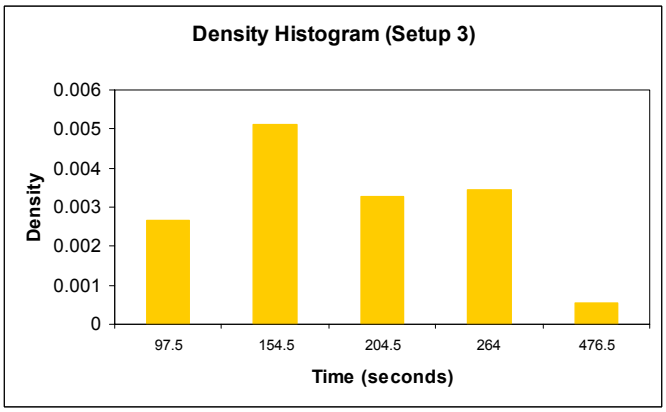
13 robots in 8m by 8m environment



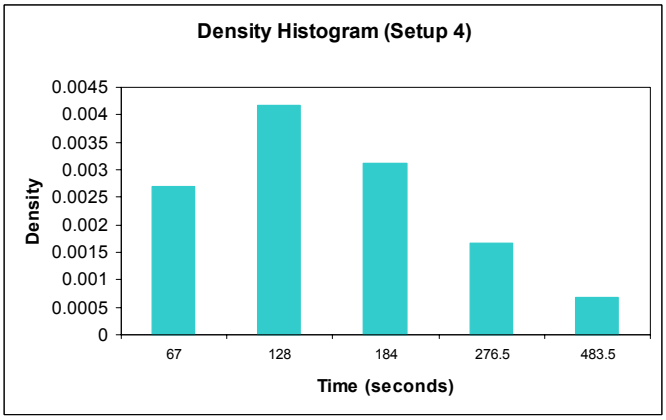
Set-up 1	
Mean / s	222
Std. dev. / s	68



Set-up 2	
Mean / s	283
Std. dev. / s	175

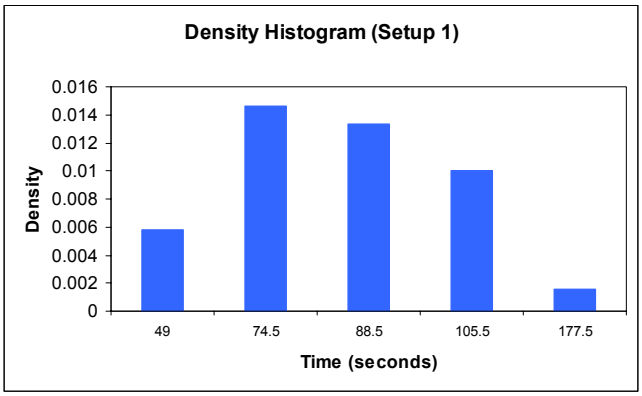


Set-up 3	
Mean / s	225
Std. dev. / s	112

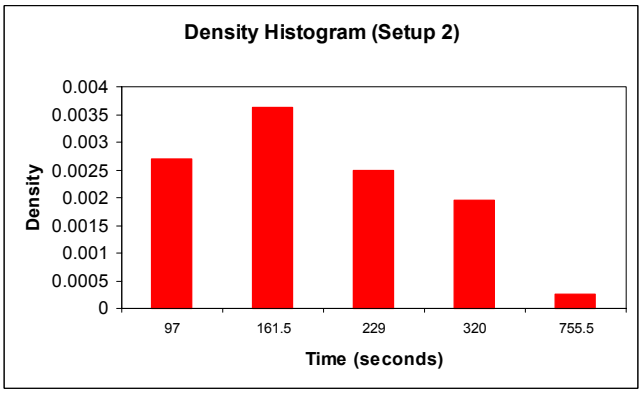


Set-up 4	
Mean / s	218
Std. dev. / s	136

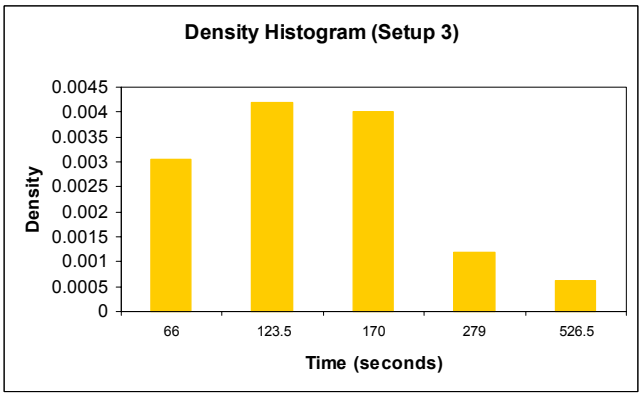
14 robots in 4m by 4m environment



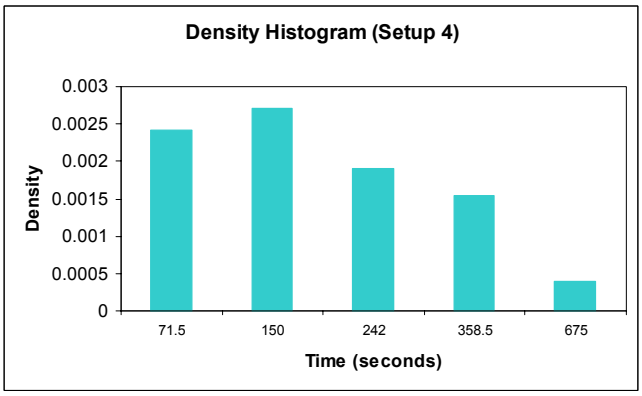
Set-up 1	
Mean / s	93
Std. dev. / s	31



Set-up 2	
Mean / s	283
Std. dev. / s	209

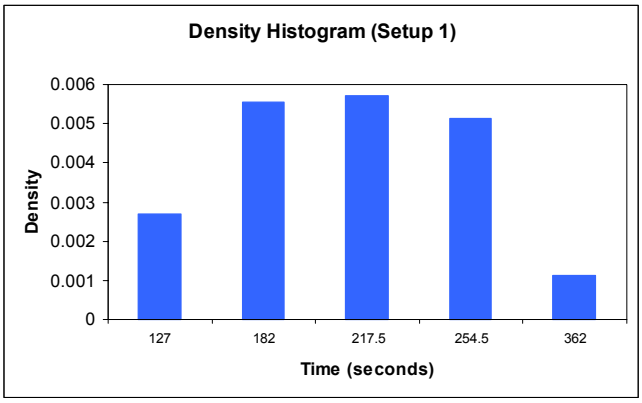


Set-up 3	
Mean / s	222
Std. dev. / s	151

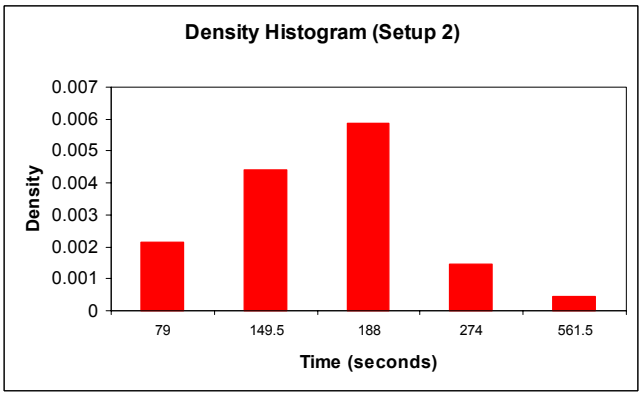


Set-up 4	
Mean / s	279
Std. dev. / s	184

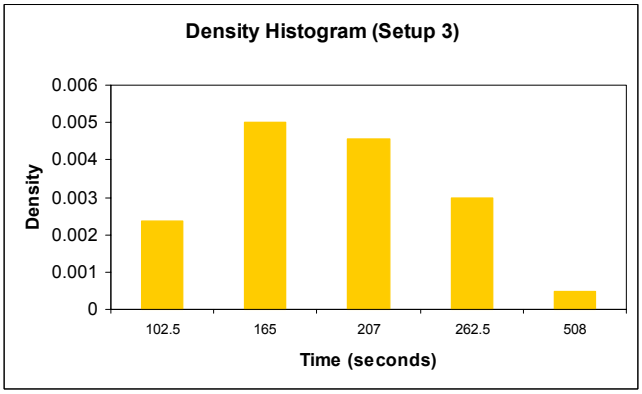
14 robots in 8m by 8m environment



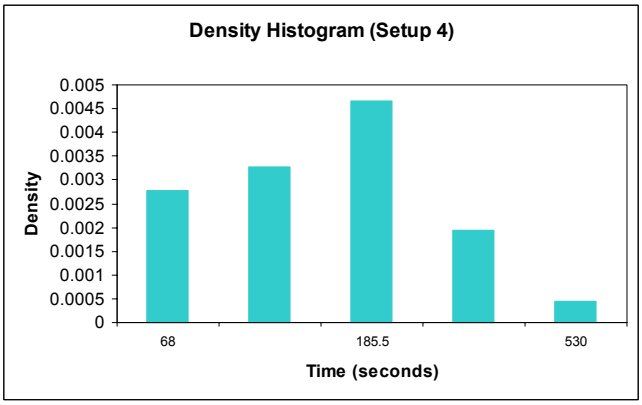
Set-up 1	
Mean / s	226
Std. dev. / s	74



Set-up 2	
Mean / s	231
Std. dev. / s	130

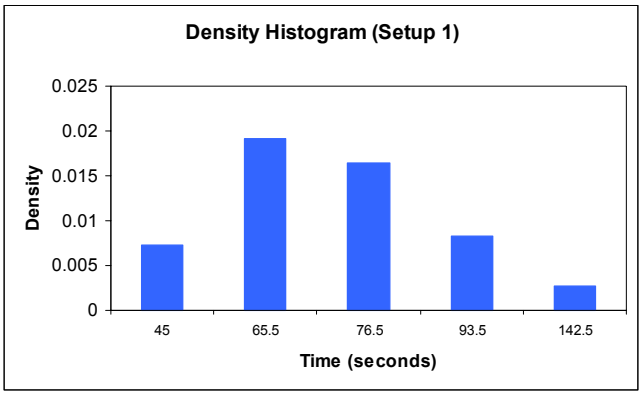


Set-up 3	
Mean / s	233
Std. dev. / s	120

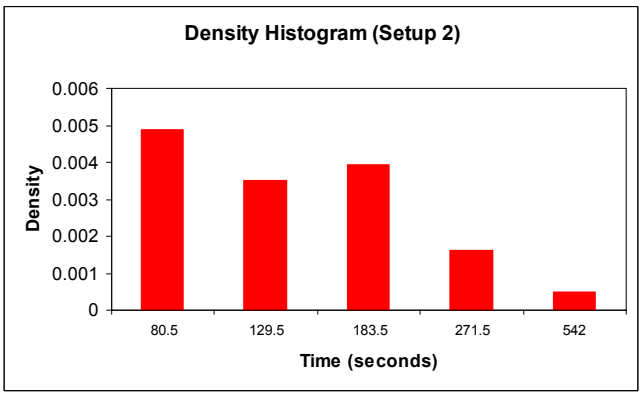


Set-up 4	
Mean / s	218
Std. dev. / s	135

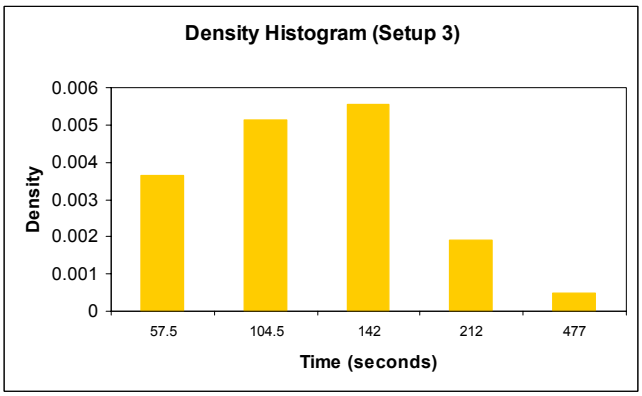
15 robots in 4m by 4m environment



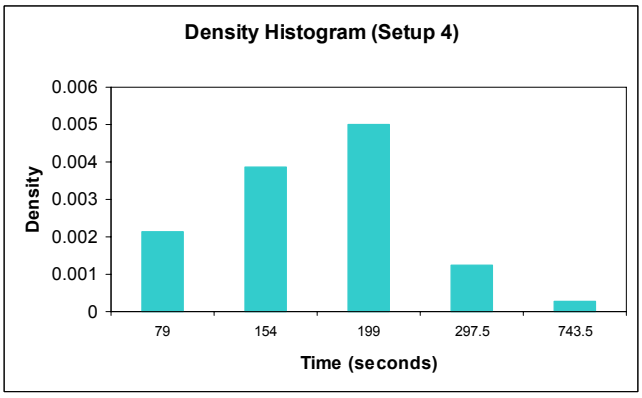
Set-up 1	
Mean / s	81
Std. dev. / s	26



Set-up 2	
Mean / s	214
Std. dev. / s	129

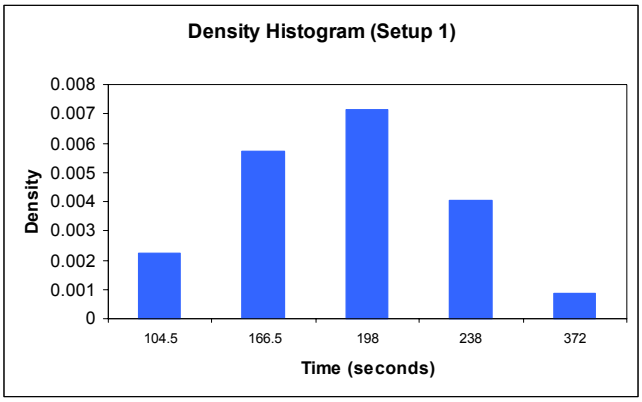


Set-up 3	
Mean / s	185
Std. dev. / s	127

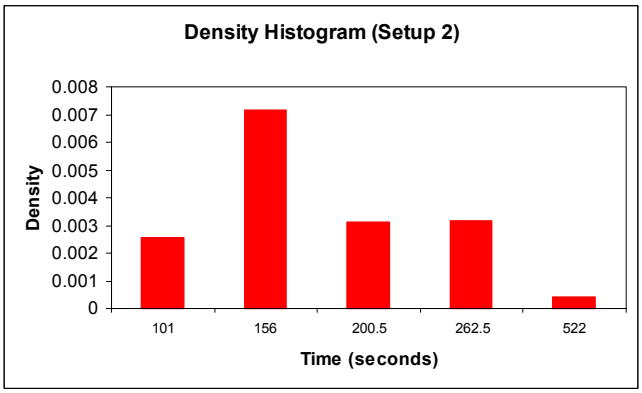


Set-up 4	
Mean / s	255
Std. dev. / s	171

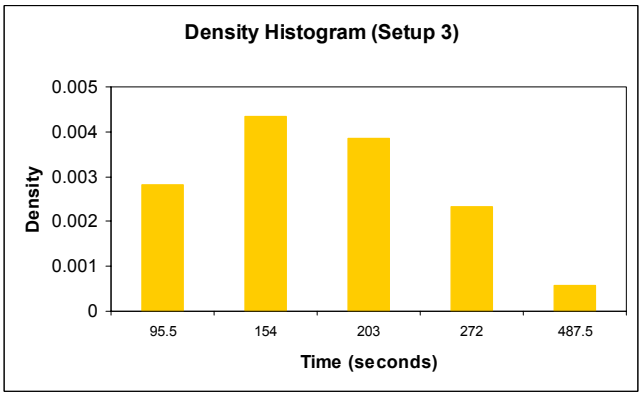
15 robots in 8m by 8m environment



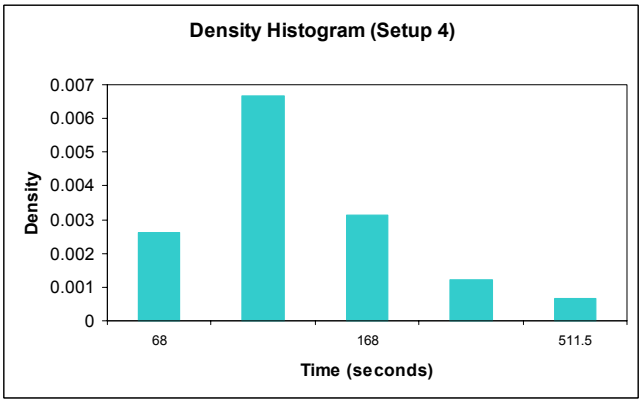
Set-up 1	
Mean / s	210
Std. dev. / s	76



Set-up 2	
Mean / s	229
Std. dev. / s	126

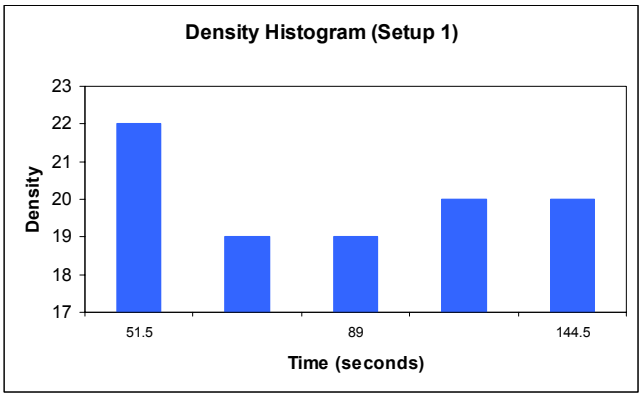


Set-up 3	
Mean / s	226
Std. dev. / s	112

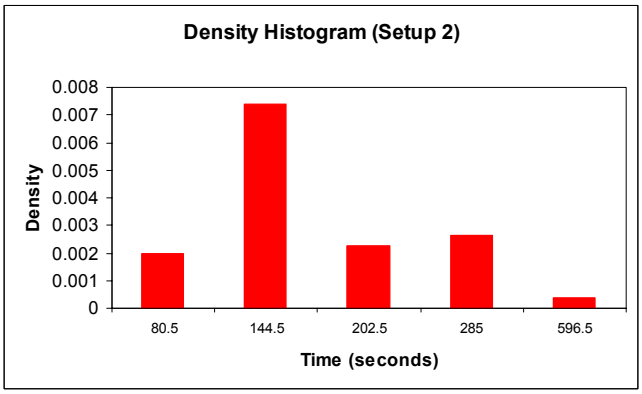


Set-up 4	
Mean / s	219
Std. dev. / s	148

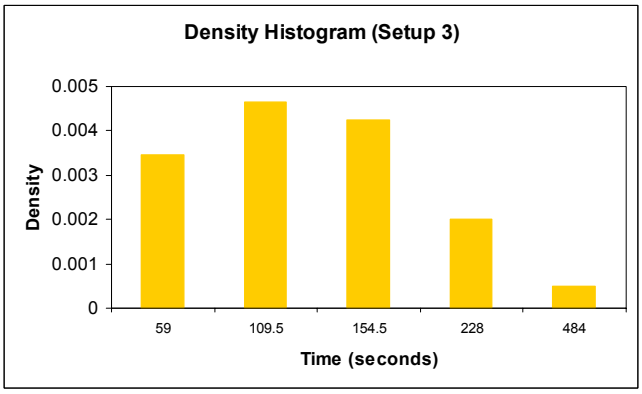
16 robots in 4m by 4m environment



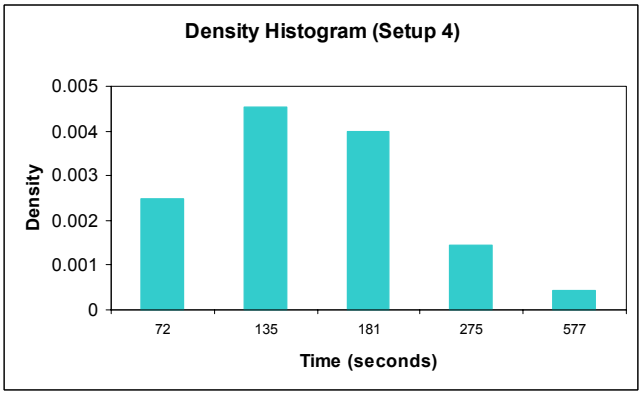
Set-up 1	
Mean / s	91
Std. dev. / s	23



Set-up 2	
Mean / s	228
Std. dev. / s	129

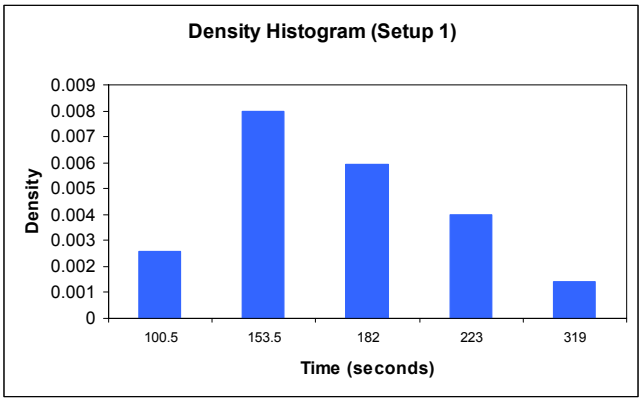


Set-up 3	
Mean / s	185
Std. dev. / s	120

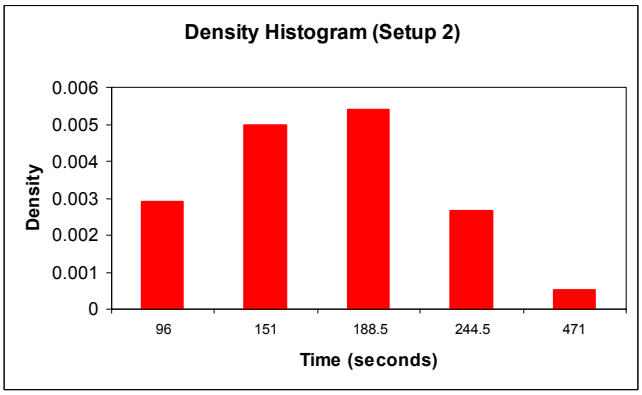


Set-up 4	
Mean / s	230
Std. dev. / s	157

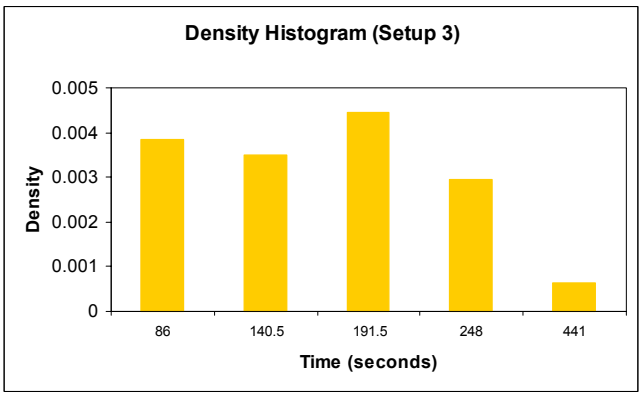
16 robots in 8m by 8m environment



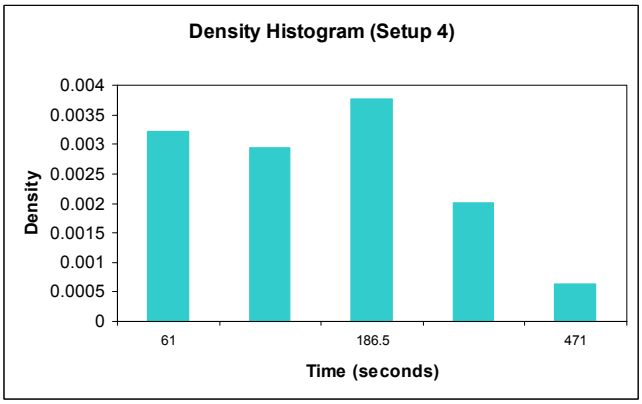
Set-up 1	
Mean / s	194
Std. dev. / s	62



Set-up 2	
Mean / s	210
Std. dev. / s	99

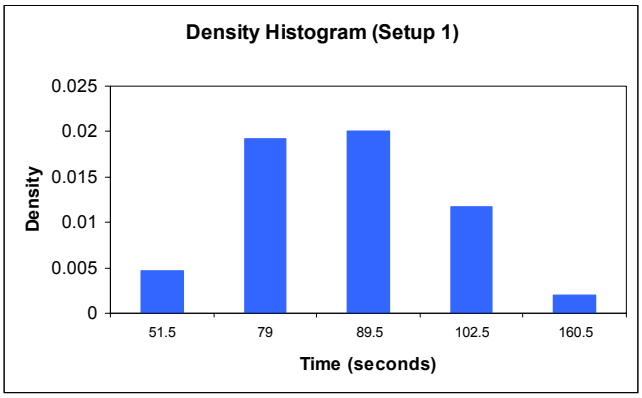


Set-up 3	
Mean / s	207
Std. dev. / s	103

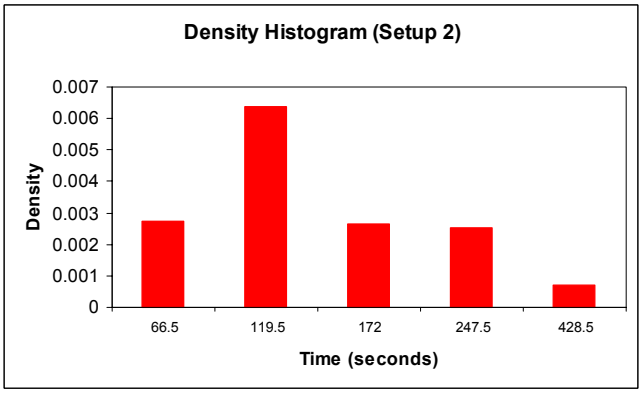


Set-up 4	
Mean / s	214
Std. dev. / s	133

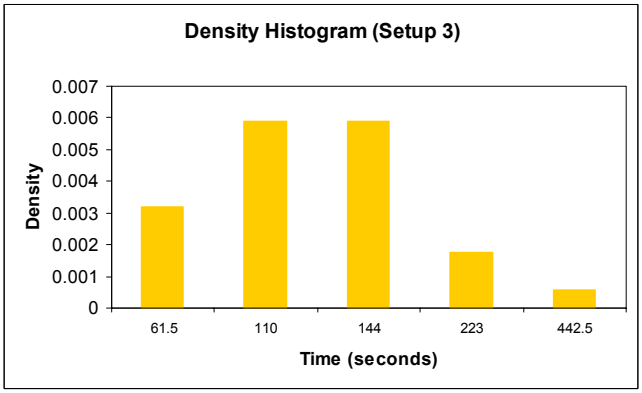
17 robots in 4m by 4m environment



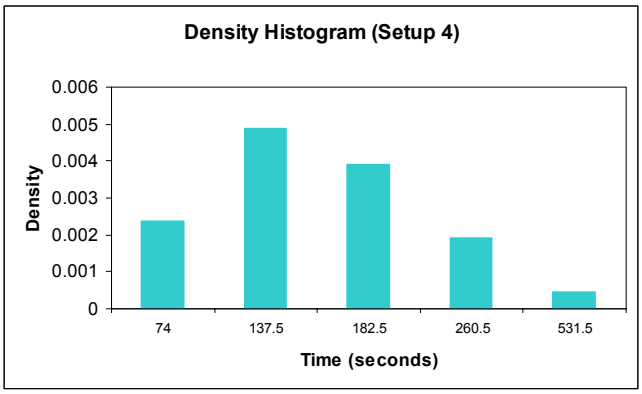
Set-up 1	
Mean / s	93
Std. dev. / s	26



Set-up 2	
Mean / s	200
Std. dev. / s	116

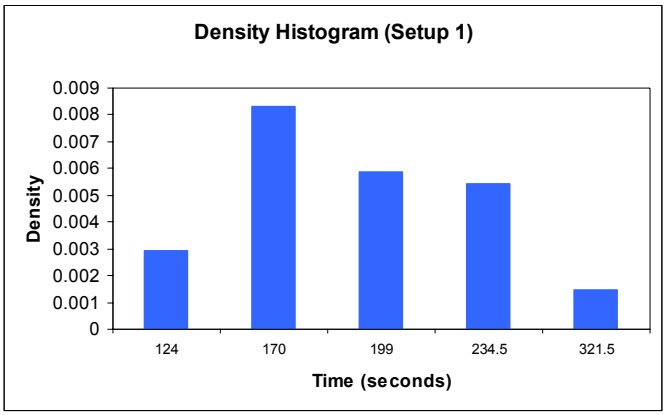


Set-up 3	
Mean / s	182
Std. dev. / s	112

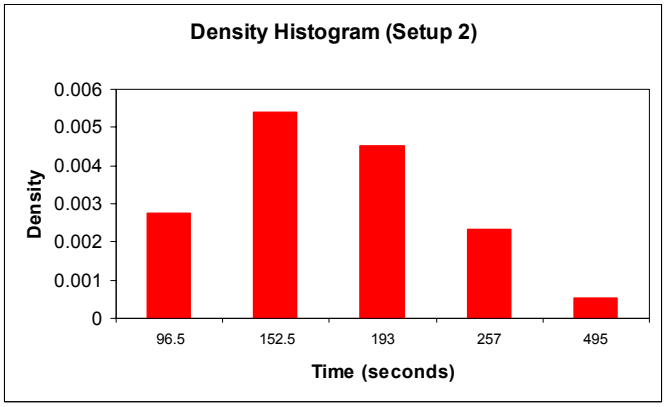


Set-up 4	
Mean / s	224
Std. dev. / s	135

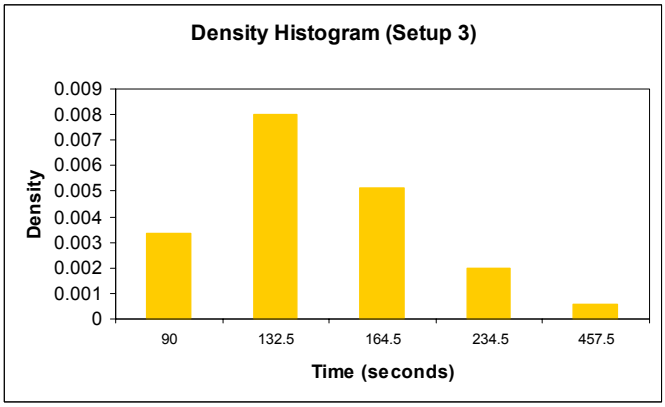
17 robots in 8m by 8m environment



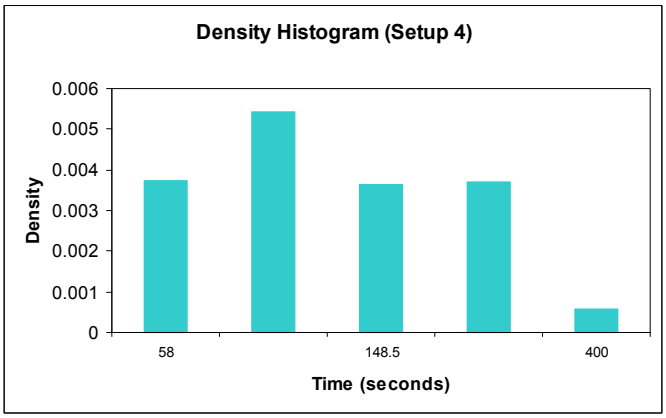
Set-up 1	
Mean / s	204
Std. dev. / s	56



Set-up 2	
Mean / s	226
Std. dev. / s	124

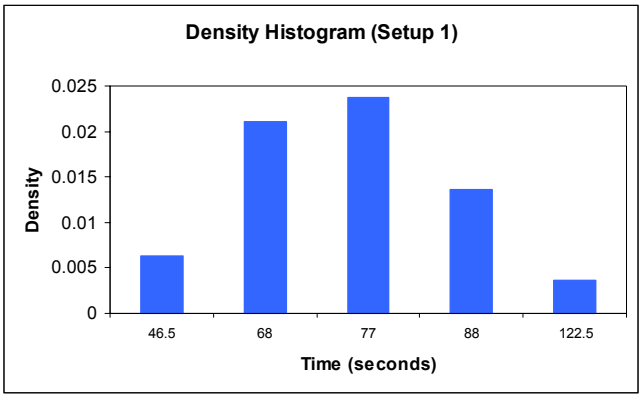


Set-up 3	
Mean / s	200
Std. dev. / s	106

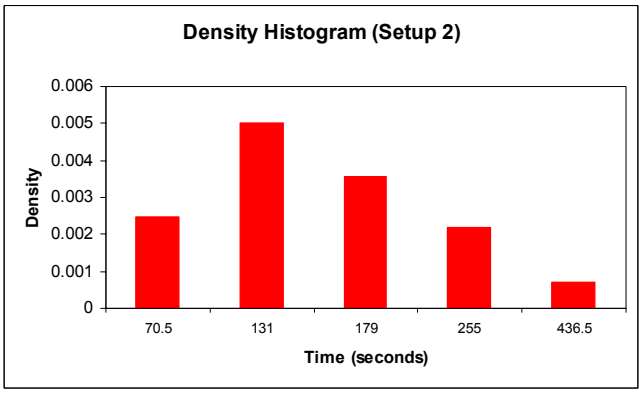


Set-up 4	
Mean / s	169
Std. dev. / s	99

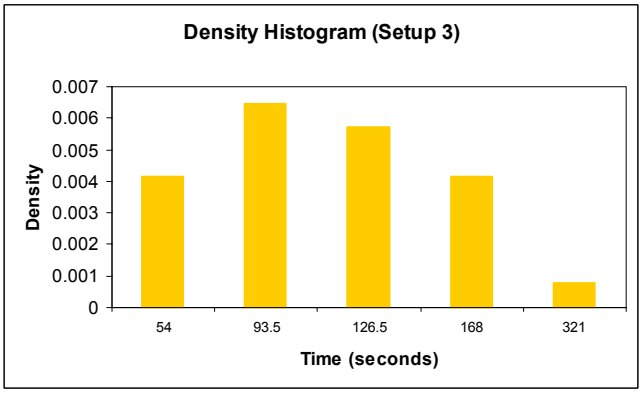
18 robots in 4m by 4m environment



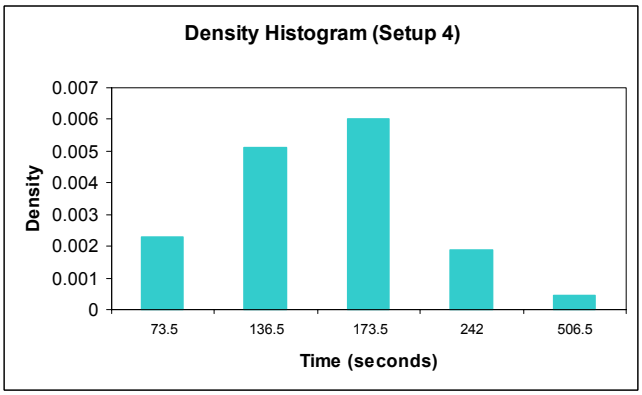
Set-up 1	
Mean / s	80
Std. dev. / s	22



Set-up 2	
Mean / s	204
Std. dev. / s	106

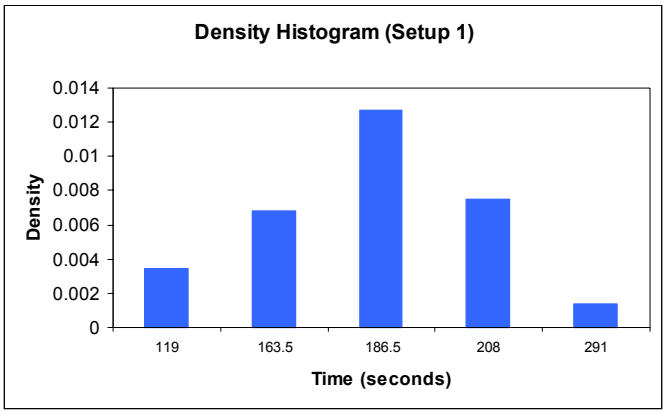


Set-up 3	
Mean / s	143
Std. dev. / s	79

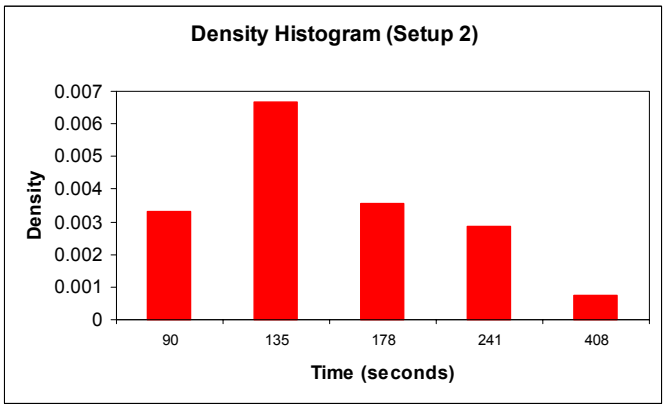


Set-up 4	
Mean / s	210
Std. dev. / s	129

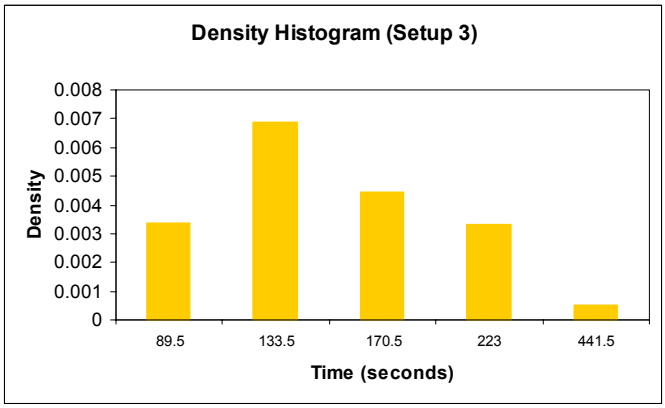
18 robots in 8m by 8m environment



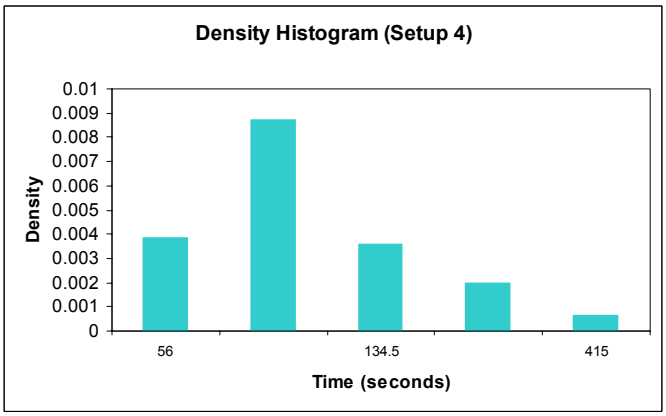
Set-up 1	
Mean / s	192
Std. dev. / s	52



Set-up 2	
Mean / s	199
Std. dev. / s	102

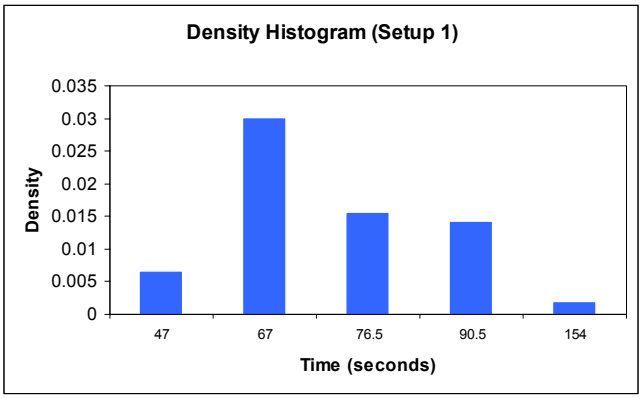


Set-up 3	
Mean / s	197
Std. dev. / s	103

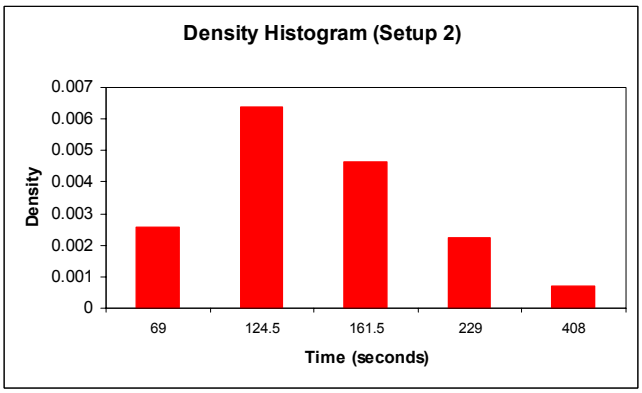


Set-up 4	
Mean / s	168
Std. dev. / s	112

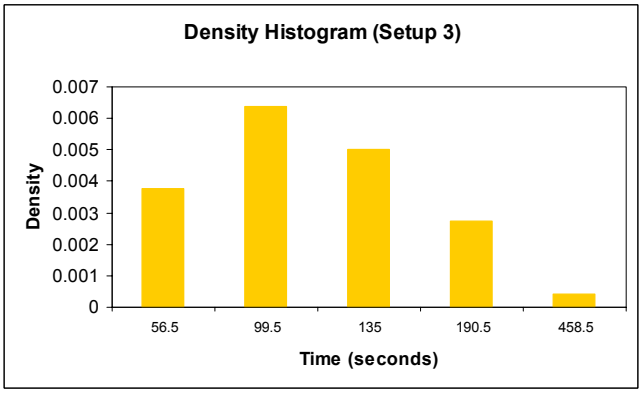
19 robots in 4m by 4m environment



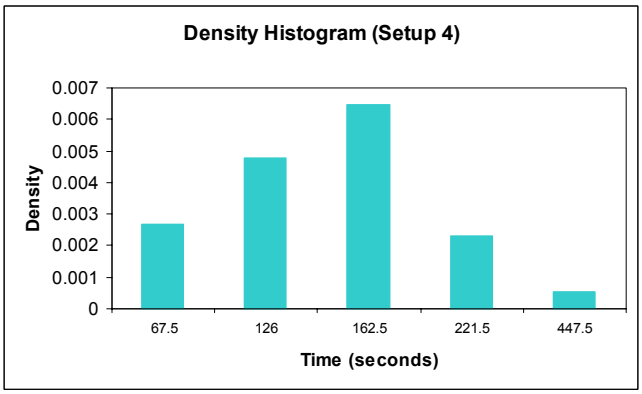
Set-up 1	
Mean / s	82
Std. dev. / s	24



Set-up 2	
Mean / s	184
Std. dev. / s	100

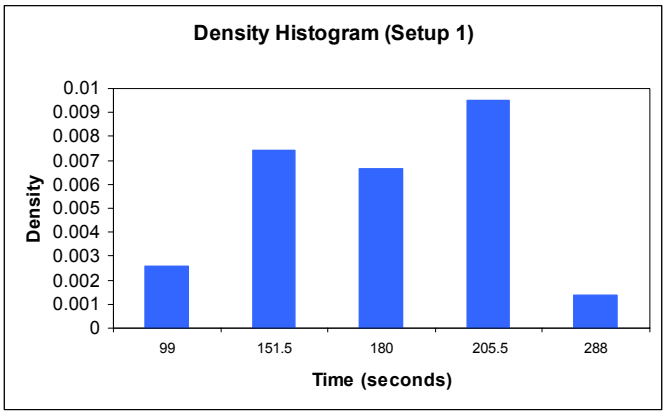


Set-up 3	
Mean / s	162
Std. dev. / s	106

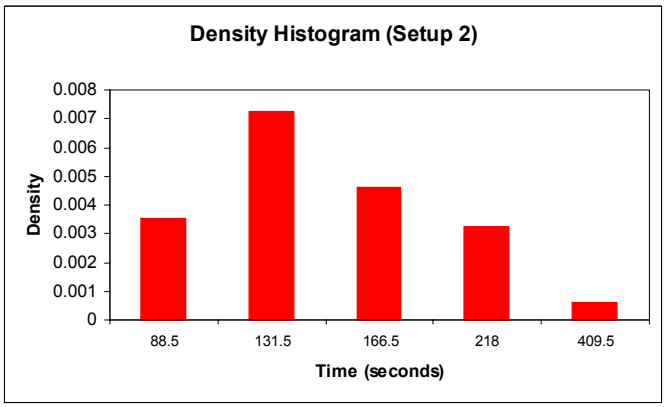


Set-up 4	
Mean / s	191
Std. dev. / s	105

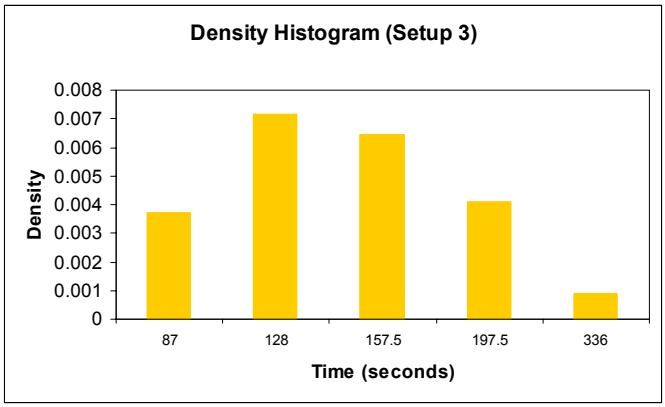
19 robots in 8m by 8m environment



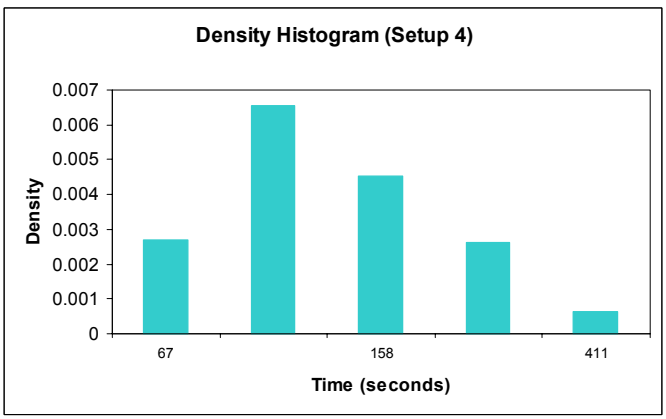
Set-up 1	
Mean / s	180
Std. dev. / s	45



Set-up 2	
Mean / s	199
Std. dev. / s	114

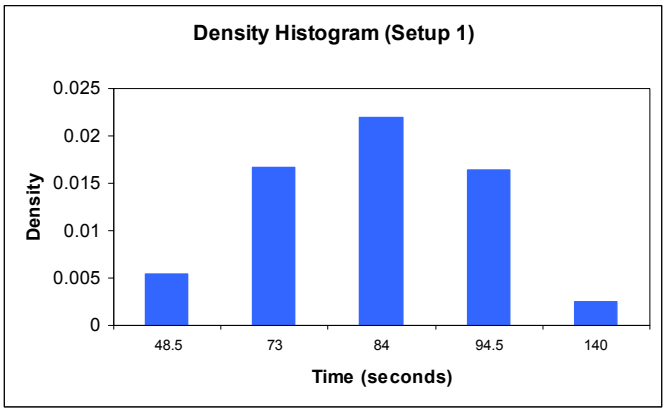


Set-up 3	
Mean / s	177
Std. dev. / s	79

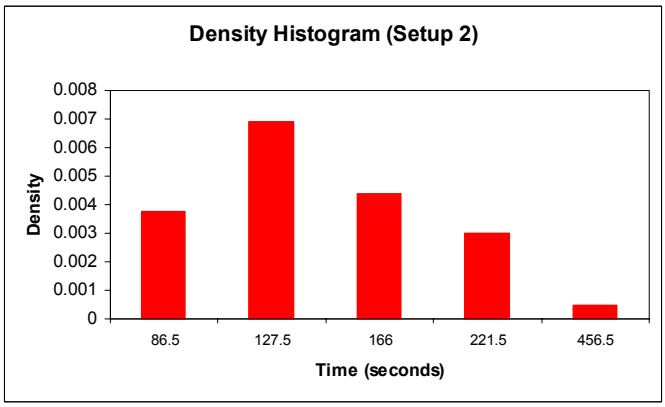


Set-up 4	
Mean / s	183
Std. dev. / s	109

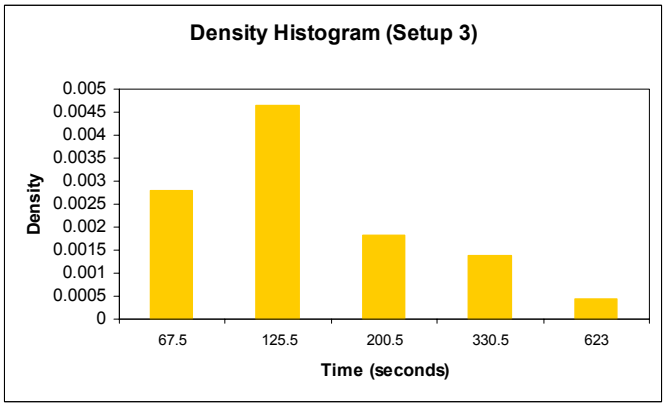
20 robots in 4m by 4m environment



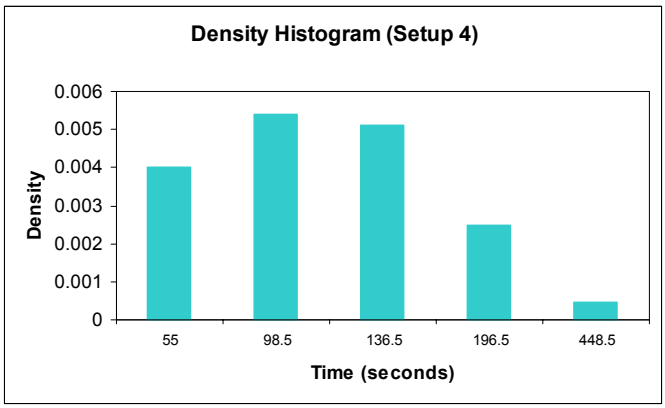
Set-up 1	
Mean / s	85
Std. dev. / s	21



Set-up 2	
Mean / s	192
Std. dev. / s	101

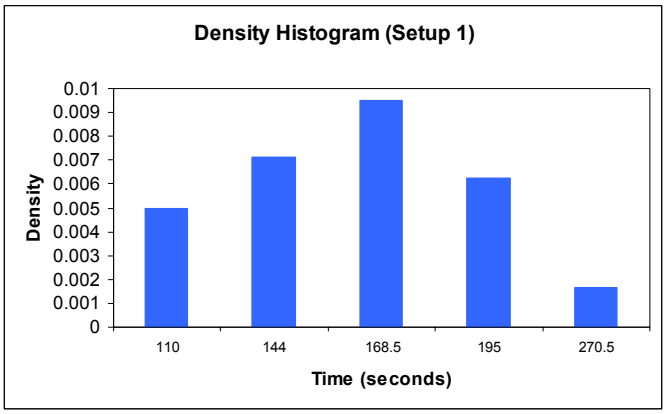


Set-up 3	
Mean / s	252
Std. dev. / s	178

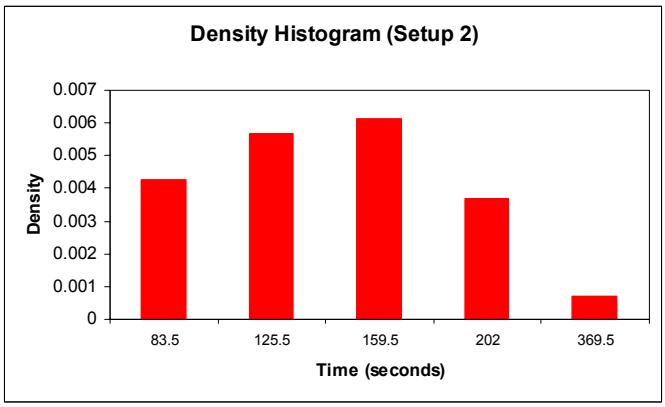


Set-up 4	
Mean / s	161
Std. dev. / s	97

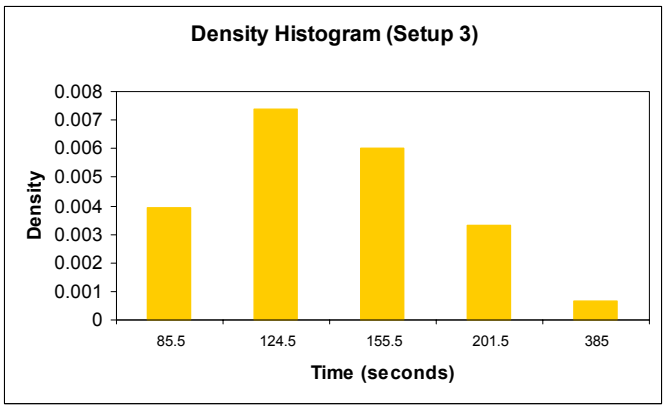
20 robots in 8m by 8m environment



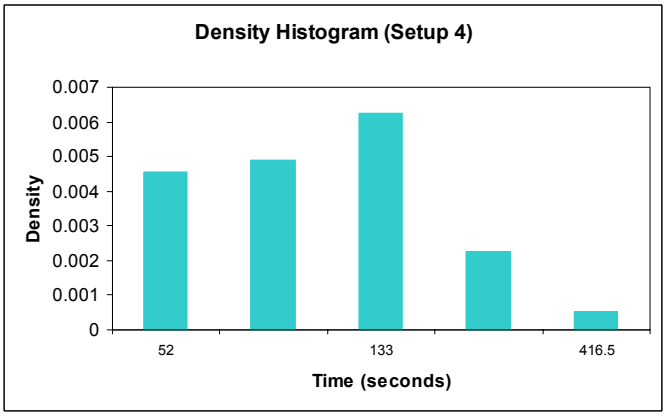
Set-up 1	
Mean / s	175
Std. dev. / s	46



Set-up 2	
Mean / s	178
Std. dev. / s	88



Set-up 3	
Mean / s	175
Std. dev. / s	84



Set-up 4	
Mean / s	168
Std. dev. / s	127