Distributed Control of Robotic Swarms

FINAL REPORT

MATTHEW BOYS

SUPERVISOR: DR AHMET ŞEKERCIOĞLU

Significant Contributions

- Improved the eBug testbed hardware utilising rapid prototyping and 3D printing.
- Established a simulation infrastructure for swarm performance tests.
- Simulated various swarm scales and map sharing techniques.
- Evaluated the efficiency of map coverage with a frontier seeking strategy.

MONASH University

Department of Electrical and Computer Systems Engineering

ECE4095 Final Year Project, Semester 2, 2015 **Matthew Boys**

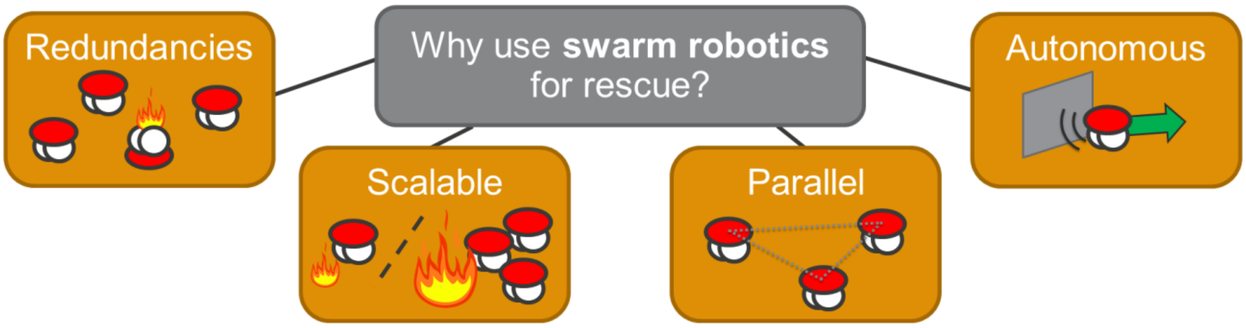
Swarm Robotics for Mapping and Rescue

Supervisor: Dr Ahmet Şekercioğlu

Project Aim: Explore the effectiveness of **swarm robotics** for mapping out disaster zones.

Chernobyl nuclear disaster *Wikimedia Creative Commons*

Why use **swarm robotics** for rescue?



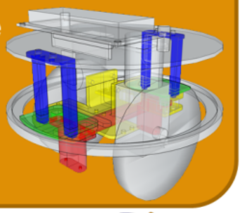
eBug Hardware

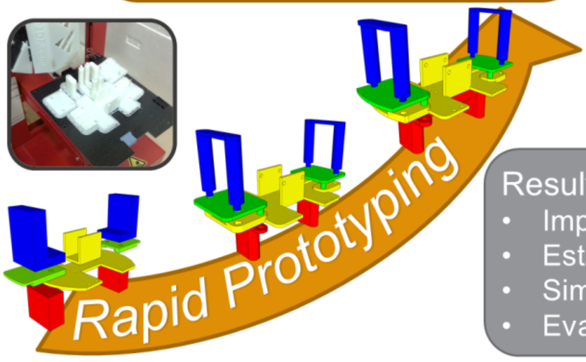
Open Source

Modular Sensors

3D Printed Chassis

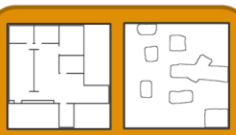
- Customisable
- Transmittable
- Reproducible



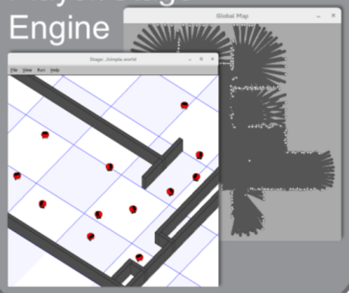


Simulations

Hardware Description


World Maps

Player/Stage Engine



Swarm Control Algorithm

Virtual Forces
- Obstacles repel
- Unexplored frontiers attract

Map Sharing

Results:
- Improved testbed hardware utilising **rapid prototyping**
- Established **simulation infrastructure** for swarm performance tests
- Simulated various **swarm scales** and **map sharing** techniques
- Evaluated efficiency of **map coverage** with **frontier seeking** strategy

Summary

Swarm robotics has the potential to dramatically assist search and rescue operations by mapping dangerous environments and ultimately saving lives. This project contributes to the field by improving an open source swarm robotics research platform using rapid prototyping and 3D printing. A novel control algorithm is then presented using frontier seeking virtual forces with several map sharing variations. These are implemented in simulation software and evaluated for map exploration effectiveness. The frontier seeking behaviour is shown to be more effective in some open-area maps than the established virtual forces control algorithm while the map sharing techniques do not give conclusive results due to the limited sample size and unexplored random factors. The observations reveal more about the emergent behaviour of specific swarm algorithms and the exploration rate improvements of frontier seeking could lead to practical search and rescue swarm robots.

Table of Contents

Introduction

Imagine if you were the first response at a nuclear power-station disaster. To coordinate the most effective rescue effort you need an accurate, up-to-date layout of the area as fast as possible. Now imagine you brought a swarm of mapping robots. These can autonomously move into the disaster zone, distributing their efforts using cleaver control algorithms and measure fire and radiation levels while plotting collapsed structures and reaching trapped humans. As the coordinator you can move straight into focusing rescue resources on identified victims while monitoring live data transmitted from inside the danger-zone from the networked robots.

This plausible scenario utilises all the strengths of a swarm robotics system, so it is no surprise that many researchers have explored swarm control algorithms under this context of urban search and rescue. However the technology is still in early stages while the occurrence of large-scale disasters continue, creating urgency for the continuation of research and the development of practical applications.

This project contributes to this field in several ways. Firstly, a thorough review of the state of swarm robotics research is presented including an assessment from the perspective of search and rescue application. Secondly, the improvements to the Wireless Sensor and Robot Networks Laboratory's eBug 2014 swarm robot are summarised showing how rapid prototyping with 3D printing was effectively used to benefit the research platform. Thirdly, a novel swarm control algorithm inspired by established methods is presented including an implementation on the Player/Stage simulation software and an analysis on exploration effectiveness.

Literature Review – Swarm Robotics Control Algorithms

Definition of swarm control

Swarm robotics involves studying how large numbers of relatively simple agents can be designed so that collective interactions and decisions result in some desired emergent behaviour. [1] In an attempt to further define swarm robotics, and to separate this field from the broader multi-robot system [2] various key characteristics have been declared. A prominent survey identified several requirements for an effective robotic swarm system. [3] To this list the term 'Redundancy' has been added as it seems a logical refinement mentioned in many other papers [4]. The refined list with short explanations is as follows:

- Autonomous: Individual agents have some sort of local sensing and actuation guided by on-board processing.
- Local Sensing and Communication: Individual robots only have access to their local sensing and communication capabilities with their local group of neighbours. While global communication is possible in a swarm it is yet to be implemented in a stable, scalable and redundant way.
- Few Homogenous Groups: The system consists of one or a few groups containing a large number of similar robots. Designing unique robots will destroy scalability and coherent group behaviour, however this does not exclude the emergence of diverse behaviour from initially identical robots with adaptive reasoning.
- Numerous and Scalable: A large number of robots working together. The efficiency or effective range of the whole system can be improved by increasing the number of robots.
- Redundancy: The stability of the swarm and its ability to achieve the set goal is not effected by the failure of a single worker.
- Incapable Individuals: The design of the individuals is not for their solitary capabilities but for the emergent behaviour from the interactions and cooperation of the group, achieving far more than what is possible with a single robot.

When designing swarm robotics control systems all these should be taken as foundational components. The first three terms restrict the algorithm design. Autonomy is a general robotics algorithm paradigm. Local interactions and homogeneity are guidelines that focus the algorithm design on leveraging the advantages of swarms over individual robots. These advantages are usually the properties of scalability and redundancy.

The final three points act as measures showing the efficiency and limits of the swarm behaviour. Comparing two algorithms using scalability and redundancy as measures, is useful for identifying optimal swarm techniques. However, there is no universal way to measure these properties and comparisons usually depend on the context of specific algorithm applications. The attribute of incapable individuals must also be defined in the context of the algorithm goals, possibly expressed in terms of hardware cost or goal completion time. These metrics can also be used to compare a swarm solution to alternative single-robot systems or specialised non-homogeneous multi-agent systems.

The need for swarm robotics in mapping danger zones

In comparison to single-agent robotics or multi-agent robotics without swarm behaviour it is expected that swarm robotics will present several advantages [1]:

- Robustness: The system has the ability to cope with the loss of individuals.

- Scalability and Parallelism: The system can be improved in efficiency or scope by adding more agents.
- Flexibility: The emergent behaviour can perform well in a spectrum of environments, scenarios or tasks.

With the continued reduction of hardware costs and the above advantages, a swarm can often be a more efficient and cost effective alternative than a single complex robot, especially in highly parallel tasks. However using a swarm of robots introduces complexity in algorithm design to achieve controllable outcomes. [5] Currently, most swarm algorithms rely on the intuition of the designer as no formal swarm design methods have been identified to achieve specific emergent behaviour. [1] Continued expansion of the swarm robotics field is of much pertinent value including the evaluation of innovative ideas and the recording of emergent behaviours from simple rule-sets.

From an application perspective, search and rescue operations in dangerous environments can benefit greatly from swarm robotics. Solutions also have the potential to achieve the ultimate goal of saving human lives. The initial stages of rescue coordination need fast, accurate and up-to-date mapping data in order to focus resources on relevant coordinates. A swarm robotics system equipped with mapping sensors can collect locational data providing dynamic maps of a disaster zone, or even some basic first-response aid to victims. Swarm robotics is ideal for this situation as it leverages all the strengths of an ideal swarm system:

- The task is completed in parallel by each agent giving faster results than a single system.
- Individual robots can operate autonomously resulting in a lighter load on human operators.
- There is no single point of failure resulting in a stable system in an environment where individuals are likely to fail.
- The swarm can be easily scaled up or down to match the dynamic nature of a spreading disaster.
- The emergent behaviour can adapt to fit a non-specific range of unpredictable scenarios.

Additionally, a swarm robotics rescue system could be easily adjusted to provide a dynamic distributed sensor array feeding data through a peer-to-peer network back to human rescue coordinators. This could include data on fire or radiation spread or the vital signs of located victims. Any progress in swarm robotics with potential application in search and rescue is thus of significant value to society.

Summary of related research

The origin of swarm research comes from the observation of complex group behaviour emerging from packs of animals following simple individual rules. Beginning with the simulation of a flock of birds it was shown that independent agents reacting to simple rules in a dynamic swarm are the sole cause of complex aggregate flocking behaviour. [6] Further work in swarm intelligence took inspiration from many naturally occurring swarm systems including fish, ants, primates, bacteria and atomic-particles. [7] Many varied swarm control algorithms have been developed from these observations of natural swarm systems. More recently others have used iterative simulate-and-improve processes to design algorithms and select desired behaviours. [1] A sample of recent, notable work will be discussed to provide an overview of the field, and a context for presenting a contribution. The examples are arranged by similar goals with a focus on experimentation methods and significant results. It has been refined to research published within the last five years related to search and rescue applications.

Mapping and terrain coverage

With the aim of developing a swarm exploration algorithm that covers each map region at a similar rate, an algorithm was developed and compared to several existing methods. The algorithm involved agents seeking unexplored cells on the frontiers and using a global optimisation strategy based on K-means clustering to select movements. This focus on balanced region exploration may have particular relevance in some search and rescue situations where optimised wide coverage is more pertinent than narrow detailed sensing. [8] An alternate frontier seeking strategy is examined in the project presented here.

An extension of the spanning tree terrain coverage method was proposed and tested using the Player/Stage simulation software. This experiment aimed to expand the method for a distributed swarm of robots while overcoming the inefficiencies present in environments with narrow doors. This demonstrated an effective use of simulated swarms to compare an existing algorithm with the proposed variations in a cellular map. [9] This study does not include real-world testing which may reveal limitations in the cellular mapping approach as physical environments cannot be easily reduced to an even grid of cells.

Taking inspiration from natural ant colony swarms, two foraging algorithms, for robots without GPS or odometry sensors, were developed. These enabled efficient group behaviour to find and collect “food”. Modelled after the ant’s ability to mark the environment with pheromone chemicals, while avoiding extra marker hardware, these robots could autonomously decide to take on a beacon role acting as a stationary marker along important paths. The experiment simulated two variations on this idea, namely the virtual pheromone model, where beacons hold a decaying marker number, and the cardinality model where beacon markers represent the distance to food. Their results showed that even with the extreme hardware limitations providing no global location data, the ant-inspired algorithms could achieve the goals with effectiveness that outperformed random walking and scaled directly with robot number until congestion impacted effectiveness. [5] Similar techniques are used in the project presented here to evaluate algorithm effectiveness under various hardware limitations.

In order to better understand the searching methods of unicellular organisms with the hope of furthering swarm search and rescue efforts it was shown that even an extremely simple communication process of purely repulsive forces can result in an effective search method. This proves the concept of simple individual rules resulting in self-optimising emergent behaviour from a biological perspective. From an engineering point-of-view the method is not advanced enough for large-scale applications as it was studied with only two agents in a one-dimensional environment. [10]

With the vision of being used as a first-response in a rescue operation a mapping algorithm was developed which relies on logging the location of collisions in a swarm of stochastically moving physical robots. The study showed that completely random behaviour on minimal hardware can produce accurate topological data, both in theory, simulations and physical testing. The procedure also demonstrates the probability approach to algorithm design and its validity for producing desired behaviour. [11]

Noticing the reliance of centralised command centres in current search and rescue robotics applications, a study using simulations measured the communication limitations of these algorithms finding a high risk of developing bottlenecks and single-failure points. This study presented the limitations and trade-offs of a centralised mapping algorithm using exploration ratios and time as benchmarks. [12]

Flocking and behaviour under virtual forces

Flocking algorithms aim to create the behaviour of coherent movements of a swarm in a common direction. This was implemented using virtual forces calculated by individual agents based on the relative positions of other robots, obstacles and goals. These virtual forces defined the individual robot's movements and resulted in emergent flocking behaviour that could be controlled by adjusting system parameters such as the optimal robot proximity to other robots. The algorithm was compared to variations that included magnitudes of these virtual forces, or goal-aware informed robots, resulting in improved goal-seeking with no additional sensor hardware. [13] This virtual forces control scheme is implemented in an alternate way in the project presented here as it produces interesting emergent behaviour.

Combining locally calculated virtual forces with controllable factors a method was developed to control large swarms to form complex shapes. Simulations were used to evaluate the performance of this method and to demonstrate how time-varying control matrices could be used to form rotating or expanding shapes that were difficult to form with previously existing control methods. [14] While the resulting behaviour is not directly applicable to autonomous search and rescue the technique of virtual forces does has applications in search contexts.

The Morse potential function emulates the forces governing atoms and can be observed in fish and bird swarms. An application to swarm robotics revealed that it can be used to disperse a swarm into a uniformly moving flock. The study examined the effect on individual processing loads as the swarm size increased and showed that the method is scalable to any size. [15] This evaluation technique is useful for algorithms that will be implemented on limited physical hardware where scaling will affect the individual's processing load. This is relevant for the algorithm presented in this project although an in-depth analysis is outside the current scope.

With the aim of maintaining continuous connectivity in very simple agents several algorithms were examined involving a swarm being pulled apart by external forces. It was shown that the optimal methods to maintain local thickness (and complete connectivity) and flexibility under external forces was to form a Euclidean Steiner tree. [16] These results, while beyond this current project, will be vital for its application as there is a need to maintain complete network connectivity in the swarm for communication and coordination.

Using a novel sensing system that enabled robots in a swarm to wirelessly determine the heading of neighbouring robots, a self-organized flocking algorithm enabled a robotic swarm to move as a coherent "super-organism" and avoid obstacles without the need of a designated leader or global homing direction. Simulations and physical robot tests showed that the sensor's wireless communication range had a greater impact on scale and stability limitations of the swarm than sensor noise or the number of detected neighbours. [17]

Swarm search and rescue surveys

A study of robotic urban search and rescue (USAR) control schemes was completed to evaluate how robotic autonomy has reduced operator workload during time-critical disaster scenes. It summarised progress in the areas of automated terrain traversal, simultaneous localization and mapping (SLAM), task sharing and high-level control schemes. The survey mentioned the various directions in which research teams are going and highlighted the unsolved challenges still to be addressed. While showing the relevance of this type of research it did not describe algorithm techniques in much depth focusing more on robotic team structures and levels of robot-human cooperation. This gives little indication on which design techniques are most effective. [18]

A survey and benchmarking of several distributed search techniques for robot swarms was carried out after the observation that this had “not yet received the proper attention”. Using the multi-robot simulator (MRSim) the algorithms were compared using the ratio of explored to unexplored map after various iterations. Some of these algorithms were also ported to physical e-puck hardware for verification. [19] This demonstrated the use of exploration ratios as a metric for search algorithm comparison that this project can duplicate.

Other novel research techniques

An alternate approach to search and rescue robotics came in the form of an air-ground robotic team. Using a team of ground vehicles for terrain mapping and a micro aerial vehicle for simultaneous localisation the experiment compared the arrangement to a single robot showing the small heterogeneous team to be more effective. While not studying all the classical swarm attributes, such as scalability and robustness, this study was useful to practically demonstrate an alternative swarm method utilising separate agents for specific functions. [20]

An innovative test of robustness was performed by studying the collision of two swarm systems, both following a honeybee inspired algorithm. With the two swarms pursuing different goals with identical methods the experiment showed the robustness of the algorithm for large swarms and that small swarm populations actually benefited from the presence of a second swarm. This shows that emergent behaviour can be quite flexible in swarm contexts, even heterogeneous swarms, with seemingly counter-intuitive results in some contexts. [21]

Using a physical exhibit an advanced collision avoidance algorithm was extended to allow robot-human interaction in an entertaining way as up to 50 robots followed visitors moving in optimised yet seemingly natural ways. This included an extension on the Optimal Reciprocal Collision Avoidance (ORCA) algorithm to allow the swarm to efficiently move through small openings with trajectories that “look natural, smooth, and goal-oriented”. This study practically demonstrated not only the application of a theoretically designed algorithm onto physical hardware, but some of the aspects of a working robot-human interface. [22]

An automatically generated swarm algorithm was developed and named AutoMoDe which utilised machine learning with bias-variance. The software was shown to develop algorithms that yield good results, both in simulations and real hardware, and is human readable. This provides an innovative evolutionary method of overcoming some of the difficulties in designing and fine-tuning swarm algorithms that can be adapted to a simulation-based swarm testing system like that developed in this project. [23]

Using a hardware setup similar to the WSRNLab’s eBug cloud architecture, a “virtual sensing” testbed was created using e-puck robots and ARGoS software to introduce virtual sensor feedback into a real-world swarm. [24] This inspires the continuation of the project presented here showing how a swarm simulation system can be integrated with a hardware testbed to verify simulation results and simulate specific sensor input data. While this project’s scope does not extend beyond the simulation testing an emulation of this system in the WSRNLab by coupling the developed simulation resources with the lab’s existing cloud architecture testbed will be a fruitful next endeavour.

The innovative Swarmanoid system displayed how a simulator and a heterogeneous swarm can be developed to perform a complex search and rescue scenario in a lab. The ARGoS software was developed by this team to overcome limitations in existing software like Stage with the flexibility required by their specific hardware system. The hardware consists of three different groups of robot for motion, climbing and flying that couple together to achieve coordinated actions. This is one of the most advanced physical heterogeneous swarm research systems. [25] While the project presented here only involves the homogeneous eBugs the Swarmaniod system may inspire further work into heterogeneous swarms such as WSRNLab's Crazyflie program which will benefit from this project.

Conclusions of literature review

The area of swarm robotics is growing rapidly with many laboratories progressing in different directions using a range of innovative techniques. This generates many creative algorithms and systems and pulls in inspiration from biology, physics and computer science.

The downside to this state is the difficulty in comparisons and evaluations. Many research teams must completely recreate existing work in their custom set-ups before they can compare it to their own results. Some standard configurations are slowly emerging, usually in the form of open-source simulation software like ARGoS and Player/Stage or accessible research hardware like the e-puck robots. These improve research effectiveness in some situations but are still limited in application.

Adding to the lack of hardware standards is the lack of defined metrics for measuring swarm algorithm performance. This could be solved by more thorough definitions on how scalability or robustness should be evaluated but is likely just a product of the varied applications that research groups have envisioned. In time these applications will become narrower as swarm robotics starts to be used in real-world situations and researchers gain a better idea on which specific behaviours are desired by society. Hopefully this will create comparable standards leading to applicable improvements while also still allowing creative theoretical work for unforeseen application.

With a specific focus on research related to search and rescue there is still much variety. While most research looks at either mapping or flocking behaviour there are many approaches to generating algorithms and evaluating results. The dominant measure is calculating exploration coverage over time as this relates closely to a swarm being used as a tool for first-response location data in a disaster zone. However there are no numerical values or standard scenarios so existing methods must still be recreated before comparisons can be made. This is likely just a reflection of how varied and unpredictable search and rescue scenarios are in real life.

Because the field of swarm robotics is still rather new there many opportunities for new research. Also, because no swarm robotics systems are in wide use in actual disaster zones, the evaluation techniques are still dependant on specific research goals. With this in mind the project here aims to contribute by innovating on an established algorithm model (the virtual forces model) and evaluating the changes by measuring combined exploration coverage over time in a widely used simulation software. It is assumed that the results will provide a record on the emergent behaviour of a new technique and the applicability of this in swarm search and rescue while producing open-source code that can be tested on eBug hardware and adapted for similar simulation tests or future comparisons.

Improvements to eBug hardware

Objectives of the WSRNLab

Monash University's Wireless Sensor and Robot Networks Laboratory (WSRNLab) has been advancing a wide range of swarm robotics technologies over the last several years with the aim of creating open-source platforms for education and research. Currently this team of students and researchers have produced a communications system, a network testbed, a low cost robotics platform called eBug, a 3D sensor network and various other localization, sensing and communication systems for a distributed swarm robotics application.

Any additions or improvements to the WSRNLab's work must hold to their vision of accessibility for education and research. The result of this is that work done in the laboratory utilises free, open-source and cost-effective resources to produce documented, extendable and accessible projects. Examples of this behaviour include the movement of software source-code and digital circuit designs into an online Git repository that can be downloaded by anyone with an internet connection, and the use of an online Wiki for thorough documentation.

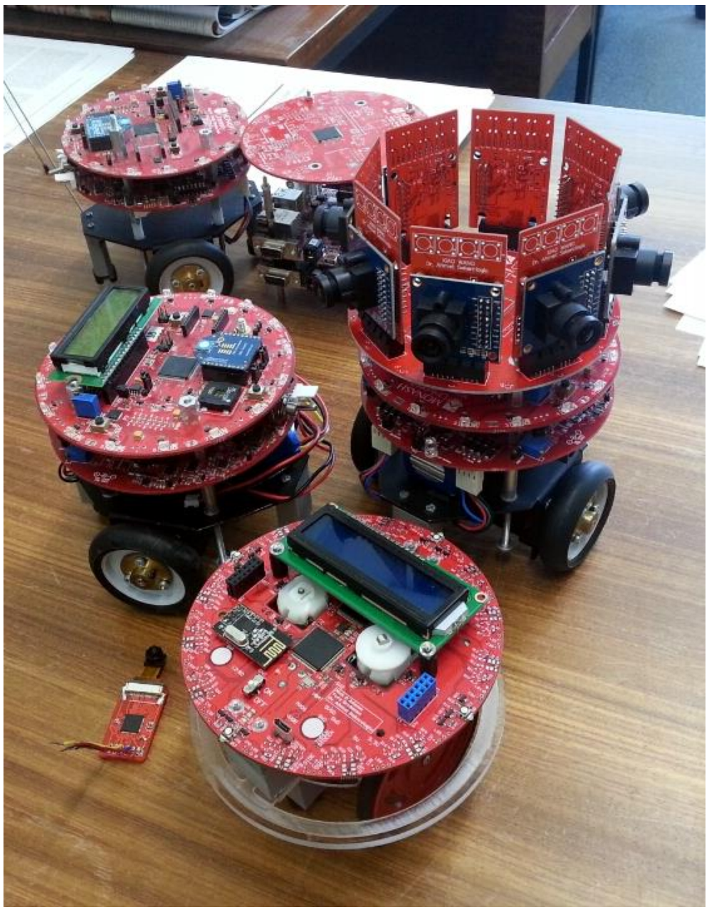


Figure 1- Several models of the WSRNLab's eBug platform

eBug 2014 hardware

One ongoing project of the WSRNLab is the eBug robot platform. The latest version of this robot, the eBug 2014 (Figure 2), is currently leaving the prototype stage and logistics are underway for a small-scale swarm to be produced. The eBug designs are available as part of the WSRNLab's open-source philosophy and older models are in use around the university. The robot is equipped with a directional bumper, differential drive system, on-board processor, wireless networking chip, LED ring and ports for attaching modular sensors.

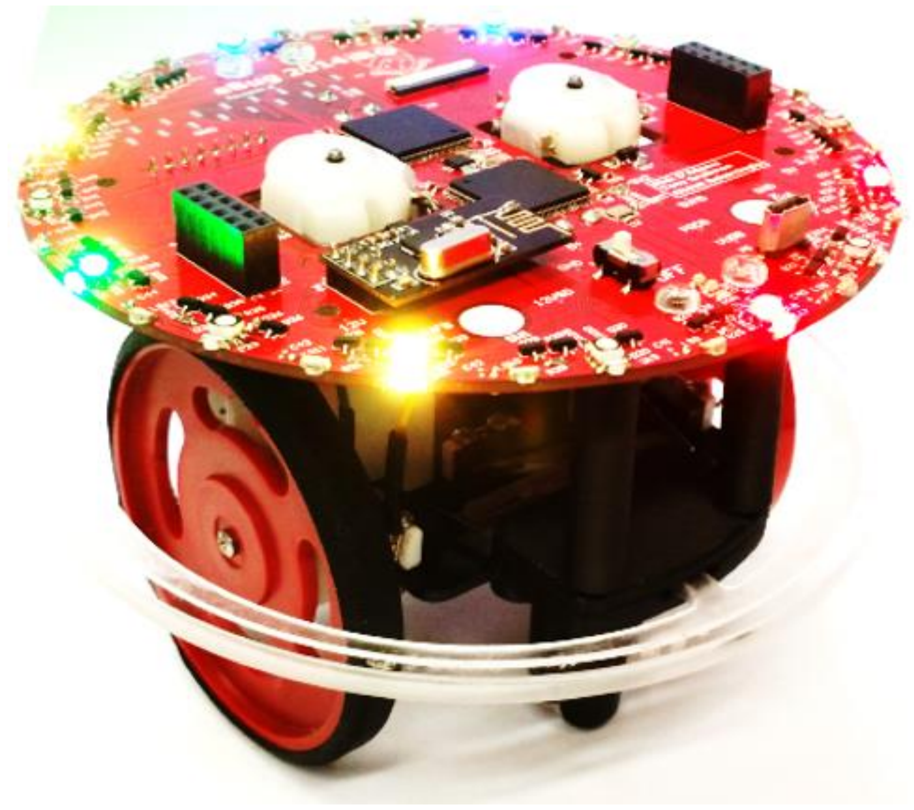


Figure 2 - eBug 2014 Swarm Robotics Platform

3D printed chassis

A required improvement to the eBug system was to reconstruct the robot's chassis in a way that can be accurately reproduced while conforming to the lab's open-source principles. It was decided that this would involve creating a 3D printable model to replace the existing prototype structure, maintaining all the current functionalities and introducing several additional features. The method of 3D printing and rapid prototyping was chosen for several reasons:

- Added complexity does not significantly increase production costs encouraging customisation and flexibility for others who wish to use the platform.
- Designs can be transmitted online and reproduced by anyone with access to a 3D printer. This fits the WRSNLab's desire for creating an open and accessible platform.
- Physical models can be quickly and cheaply produced for real-world testing and debugging.
- The process of rapid prototyping and 3D printing can be evaluated for its potential in future projects.

The combined goals for this section of the project were:

- Access to the battery so that it can be replaced without disassembling the robot.
- Open-source design so plans can be accessed and customised.
- Maintain the dimensions of the prototype with respect to supporting the other robot components.
- Maintain full functionality of the circular bump sensor.
- Structurally sound and durable.

- Evaluate the process of rapid prototyping and its potential for future projects.
- Print a physical model and apply it to the latest eBug prototype.

Methodology

This process began with the installation of the free SketchUp Make 3D modelling software.[1] This program was chosen for its free licence for education and non-profit use and for its widespread use and familiarity among the Engineering staff at Monash. While this program is designed for architectural work it is suitable for basic 3D modelling and easy to learn. It also has a plug-in system enabling the addition of features developed by the SketchUp community.

Once several online tutorials were completed on the software the existing 3D designs for the bumper and guide were imported and a rough representation of each of the major eBug hardware pieces were built (Figure 3). This was then used to design a new chassis which was printed in collaboration with Michael Zenere. The physical model was assessed and the design modified and re-printed. This process continued several times, with each revision being documented and made available on the WSRNLab's online Wiki[2].

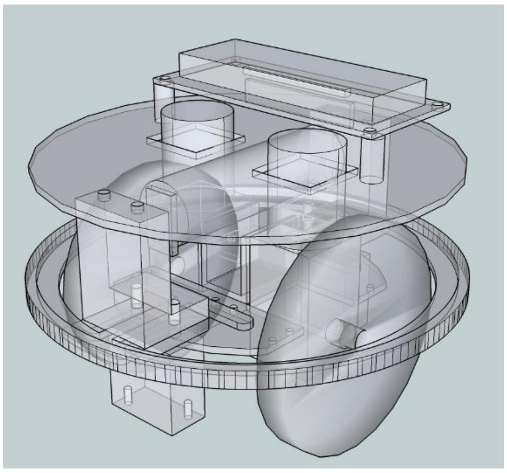


Figure 3 – 3D model of prototype eBug hardware

The design involved several pieces that clipped together with close-fitting pins and holes. This reduced the number of screws required and the construction complexity, while enabling specifically desired shapes that could not be printed as one piece. Modifications on the original design included adjustments to screw-hole sizes for easier construction, movement of bump sensors for better bumper movement, structural changes to add strength at identified weak points and changes of pin shapes to eliminate the possibility of incorrect construction. A full change log can be found on the WSRNLab's online Wiki including all major versions of the 3D models.[3] The change log is also printed here in Appendix A – 3D Printed Chassis Design Change Log. Images of several major iterations can be seen in Figure 4.

[1] SketchUp Make is freely available to download from: http://www.sketchup.com/products/sketchup-make
[2] WSRNLab's Wiki: http://www.ecse.monash.edu.au/twiki/bin/view/WSRNLab/WebHome
[3] Change log and 3D models on WSRNLab's Wiki: http://www.ecse.monash.edu.au/twiki/bin/view/MBotSwarmR/MechanicalDesign

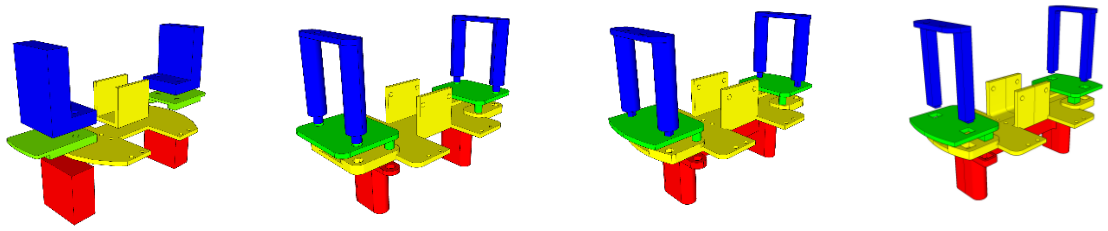


Figure 4- Several major iterations of 3D modelled eBug chassis

Several methods were trialled for 3D printing including a plastic additive printer and a selective laser sintering machine. The sintering model was quite brittle and broke during testing. Two different colours of polymer were tested on the additive printer, white and black, with the black producing slightly stronger models, especially between printing layers.

Evaluation

All the objectives for this section of the project were met with a working design being printed and attached to the eBug hardware (see Figure 5). The bumper worked well, functioning better than the prototype as no tension springs were needed, and the battery was accessible without any disassembly as was desired. All the files were consistently uploaded to the WSRNLab's Wiki website and kept up-to-date where they were accessed by other users for printing. While initial iterations of the printing often broke between printing layers, improvements were made by increasing wall thicknesses and orienting parts to print horizontally. It was also noted that different printing techniques and even different colours of materials produced different product strengths. The durability of the final model, when printed with the black polymer, was at a satisfying level.

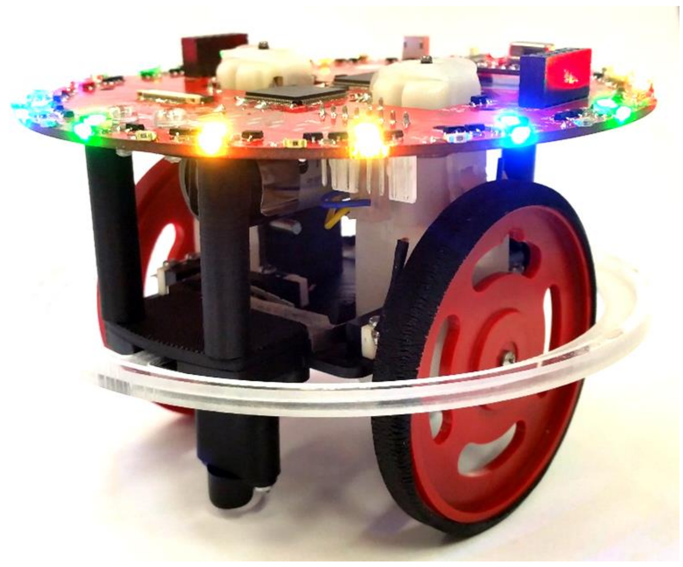


Figure 5- Current eBug model with 3D printed chassis

The process of rapid prototyping using 3D printing produced some interesting results which will benefit future projects that consider this method. Firstly, the process produced a satisfying final product, marking the method as a successful way to design and produce new hardware. The final model could be easily uploaded to the WSRNLab's file repositories continuing the open-source paradigm of the lab and providing easy transmission of updates and designs to the people who printed the parts.

While printing often took several hours to complete and a few days to organise around the schedules of collaborators it was quicker and simpler than other manufacturing techniques used in the past and resulted in exact copies of the final product that could be completely and physically evaluated. This enabled a high degree of fine-tuning of the design including adjusting screw-hole diameters by very small amounts to get the optimal result. It also meant that very large changes could be made quickly in the SketchUp software, such as changing the position of the bump sensors, with no added manufacturing complexity.

The limitations of the SketchUp modelling software did restrict complexity of the design although this did not stop any of the desired features from being implemented. Some adjustments were more difficult to make in SketchUp than alternate engineering-focused 3D modelling software like SolidWorks or NX. This increased the time to make modifications to the models. However, the simplicity of SketchUp meant there was little time needed initially to learn how to use the software, far less in comparison to SolidWorks or NX. Since the WSRNLab aims to provide accessible projects, the use of free and simple modelling software is a justified requirement, even if adjustments are more time consuming, as anyone should be able to download and modify the files with no added cost for software and comparatively little learning time.

In conclusion the requirements were satisfied and the process met expectations suggesting that this method, of rapid prototyping using 3D printing, is an effective way to produce structural hardware. The process was well suited to the philosophy of producing open, accessible and modifiable products. Use of rapid prototyping with 3D printing in future projects of the WSRNLab is recommended, especially for simple structural hardware.

Algorithm Design

Project scope adjustments

During original project planning it was expected that the WSRNLab's cloud architecture testbed would be updated enabling physical swarm robotics tests to be completed on the eBug 2014 hardware. After several weeks of work it was concluded that this task would take far longer than allowed and so an adjustment to the project scope was agreed upon. The unforeseen difficulties were completely due to the disorganisation and lack of documentation related to the existing cloud architecture testbed software which had been constructed over several years by several student researchers.

The revision involved abandoning the use of the cloud architecture testbed and instead investigating alternative swarm simulation techniques for analysis of swarm control algorithms.

Player/Stage

Several simulation packages were investigated for use in swarm algorithm analysis and the Player/Stage engine was selected because it was open source, had sufficient documentation, was in active development and in widespread use for swarm research. The Player/Stage system is composed of two systems: namely Player, a hardware abstraction library; and Stage, a multi-robot simulator.

A typical simulation consists of a set of hardware descriptions and simulation configuration files and a robot control algorithm. The algorithm utilises the Player libraries to generalise the robot hardware and communicate with either real or simulated sensors and actuators. The configuration files describe to the Player/Stage system the robot hardware and the simulation parameters. Then Stage begins simulating the robots in the specified environment, the control algorithm is started and Player handles requests between the control code and the Stage simulator.

One major advantage of using the Player/Stage system is that Player can also communicate with physical hardware. By loading different drivers the control algorithm can communicate, via the Player abstraction, with a robot's real sensors and actuators in the same way as it does with the Stage simulation. The result is faster conversion from simulations to physical testing as control code can be ported directly to a robot running the Player engine.

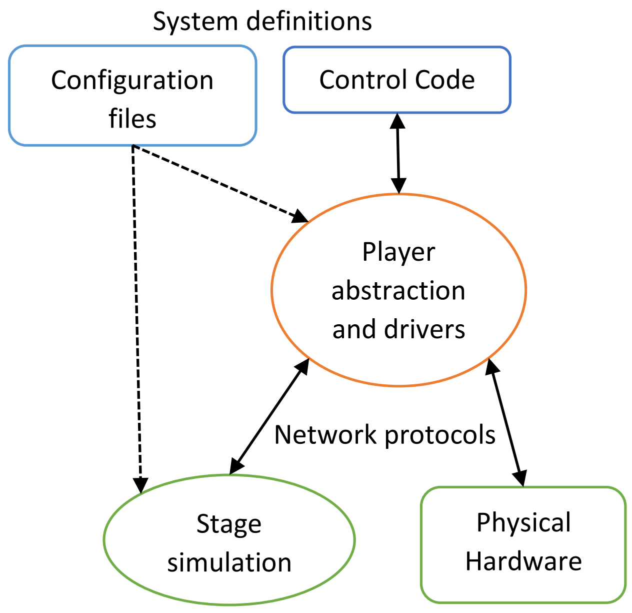


Figure 6 - Diagram showing Player/Stage system

Additional features include the ability to run Player communications through network protocols or couple Stage simulations with physical systems. This could be adapted to the WSRNLab's cloud architecture platform for easier swarm testing on eBug hardware and cloud simulations.

There were some difficulties with installing the Player/Stage software. Firstly, the program was not available pre-compiled for the lab's Kubuntu operating system. After several unsuccessful attempts at obtaining the libraries needed for compilation it was discovered that a pre-compiled version exists for the Fedora operating system in its standard repository. This made installation very easy for computers running Fedora so the Fedora was installed on the WSRNLab's Ebug-master PC along with Player/Stage. The Player/Stage installation process was documented, along with the most successful attempts for the Kubuntu system, for future reference.

Some struggles were also had with using the Player/Stage software. One programming error was identified that caused the Ranger->GetAngularRes() to return a value that was not what was expected. An alternate function was coded and used in place of this and extensive testing was done to ensure all other required functions behaved as expected. Fortunately the program is in active development and hopefully this issue will be remedied. There were also some sample programs for swarm robotics simulations using an alternative Stage 'ctrl' format. This system is mentioned as being better optimised for swarm simulations than the standard method as it utilises multi-threading technology. Unfortunately there was insufficient documentation on this particular feature so it could not be used.

In conclusion the Player/Stage system was sufficient to perform the desired swarm control research for this project. Further familiarity with the software within the WSRNLab, combined with updates from the Player/Stage development team, could lead to better swarm algorithm research in the future.

Goals

This project aims to combine established virtual-force swarm control with innovative map-sharing and frontier seeking methods and evaluate the emergent behaviour using open-source simulation software. In contrast to many existing methods for designing algorithms on extremely minimal hardware, this project takes a top-down approach of matching the algorithm to a specific existing hardware's strengths. The hardware, the eBug 2014, has networking, 360° sensing and moderately capable on-board processing.

The virtual forces method simulates dispersing influence field emanating from obstacles and neighbouring robots. This results in the robot experiencing several forces which are summed, normalised and used to decide on the robot's next motion. The robots also record local map data on explored regions which can be shared under specified conditions.

Since search and rescue is the envisioned application, the metric used to evaluate the swarm behaviour was the time and efficiency of exploration of various maps. In the context of a search and rescue operation this corresponds to supplying location data of an entire disaster area as quickly as possible to coordinators. It was thought that this metric better suits the search and rescue application than the locating of unknown goal particles because in disaster zones it is equally important to confirm that an area is free of humans than locating survivors because rescue resources are limited.

The scope of this research includes simulating the algorithm on various sized swarms up to 10 robots and comparing the virtual forces method to the variation with frontier seeking. Robots with various map sharing abilities will also be simulated and the effect of their wireless limitations evaluated.

Assumptions

This project does not include a comparison of the simulated results to a real-world test on physical hardware. It is assumed that the Player/Stage simulator will perform accurately enough for behavioural observations to be made. As is the normal process with swarm research, the algorithm should be performed on real hardware to verify the simulated results and it is strongly recommended that this be one of the first steps of any follow-up research.

The various algorithms involve some constant parameters that were manually fine-tuned during the design, but remained consistent throughout the experiments. While the effect of adjusting these values was noted during the fine-tuning it was not thoroughly examined. It was assumed that these constant values had no effect on the results.

There was some stochastic behaviour as discussed with the map-sharing experiments. While many of the tests were simulated several times it was assumed that the data here shows a representative behaviour. A sensitivity analysis or a larger sample of tests could be completed to generate a full probability distribution for the various emergent behaviours and results however this project instead uses the small sample of results in light of the consideration of possible random variation.

The simulations included global positioning data without consideration as to how this would be calculated in a real-world scenario. Since there exists hardware that provides global positioning data, and much research has been done on alternate ways for swarm robots to calculate this, it was included in the simplest implementation available.

Simulation set-up

The experiment was designed to emulate the eBug 2014 hardware so initially a hardware description of the eBug was coded into Player/Stage using the built-in libraries. A graphic representation of the simulated robot, while not relevant to the exploration efficiency results, was written for visual observations and display purposes. This simple model can be seen in Figure 7. The current eBug prototype was also weighed and measured and the accurate mass and dimensions were included in the simulation description.

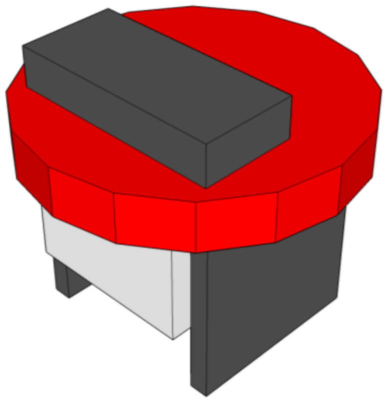


Figure 7 - Visual representation of eBug 2014 in Player/Stage simulator

The eBug is equipped with ports for connecting sensor modules. For the simulations a generic ranger library was used which acts like common laser rangers or ideal sonar. Since this technology is in wide use, and some specific range sensors like the Microsoft Kinect are available for the eBug, it was assumed that simulating the robot with this additional hardware was reasonable. The values of the sensors can be adjusted but for these experiments an array of 35 sensors at 10° increments with 2m reading radius was consistently used.

The world files were specified to emulate a 10 meter by 10 meter space with three different layouts. The simulation calculates robot sensors and movements in two dimensions even though the graphical display is 3D. These files also included information on the number of robots in each simulation and their initial positions. While the swarm number was varied between 1, 2, 5 and 10 robots, the initial positions were kept constant.

Swarm control algorithm

A control algorithm was implemented and several variations forked from this. All algorithms required each individual robot to keep a local occupancy map and update it with all new range data, even if it wasn't used to determine motion.

The most basic control algorithm re-implemented an established motion control method where each robot individually calculated virtual repulsive forces from neighbouring robots and detected walls. These virtual forces were then summed and the result normalised before using it to determine that robots next wheel motion.

The frontier seeking variation introduces a third type of virtual force, an attractive force from unexplored frontier points marked on the robot's local map.

The map sharing variations added the ability for robots to transfer its local map data into every robot within a set proximity. The transferred maps were used to update the receiver's local map including deleting frontier points that were marked as explored by the sender. The global map sharing method extended the allowed transfer range to infinity resulting in all robots having identical maps at all times, effectively sharing a global map. This was made to behave as a best-case map sharing scenario.

Algorithm implementation

The algorithm was coded in C++, utilising the LibPlayer++ library for access to simulation data and the OpenCv library for displaying robots' mapping data. Many basic algorithms were constructed to test features and functions. The final set of algorithms were constructed using an identical set of functions and program structure with the variations encoded by omitting or adding only the code relevant to the feature being studied. This means that all experiments ran in very similar ways and used identical methods for tasks unrelated to the experiments like memory management and map exploration processing.

The source-code for both the control algorithm and simulation set-up can be downloaded from the WSRNLab's GitLab repository[4] which includes detailed comments. The general flow of the control algorithm is also outlined here using pseudo code and mathematical representations.

Figure 8 - Swarm simulation control pseudo code

```
Initialise memory for each robot and its local map
Initialise memory for the combined global map
Loop forever
{
     For each robot
     {
          Update robot's simulated sensor data

          Record on local map any new walls, explored areas and
               frontier points
          Copy local map data into combined global map

          *Share local map with neighbours or copy global map

          Calculate virtual forces from walls
          Calculate virtual forces from neighbours
          *Calculate virtual forces from frontiers on local map

          Sum virtual forces and normalise into motion vector
          Set robot's motion from motion vector

          Calculate and print exploration percentage and
               simulation time step
     }
}
```

Figure 8 displays the main processes of the algorithm with lines marked with an asterisk indicating functions that were adjusted or disabled to generate the variations. Each line in Figure 8 represents a single function, some of which are expanded upon below:

Record on local map any new walls, explored areas and frontier points

This was performed by scaling the simulated ranger data into equivalent coordinates on the local map matrix. OpenCv drawing functions were used to mark empty space as lines from the robot's position along the sensor headings. The maps were implemented as 500x500 occupancy grids storing information with numeric values:

[4] WSRNLab's GitLab repository including source-code access: https://gitlab.com/groups/wsrnlab

- 1: Identified environment wall
- 0: Unexplored region
- -1: Explored empty space
- -2: Explored empty space marked as a frontier

A frontier mark was placed if the corresponding range sensor returned its maximum value and the region was previously unexplored. This meant that frontier points were recorded at the boundary of explored and unexplored regions as sensor reading were taken and overwritten when that area was later fully explored.

**Share local map with neighbours or copy global map*

For non-map sharing variations this was not included. For map sharing this read the wireless proximity value and copied the robot's local map into every neighbour within the specified Euclidian distance. For the global map sharing variation the local map data was updated from the combined global map.

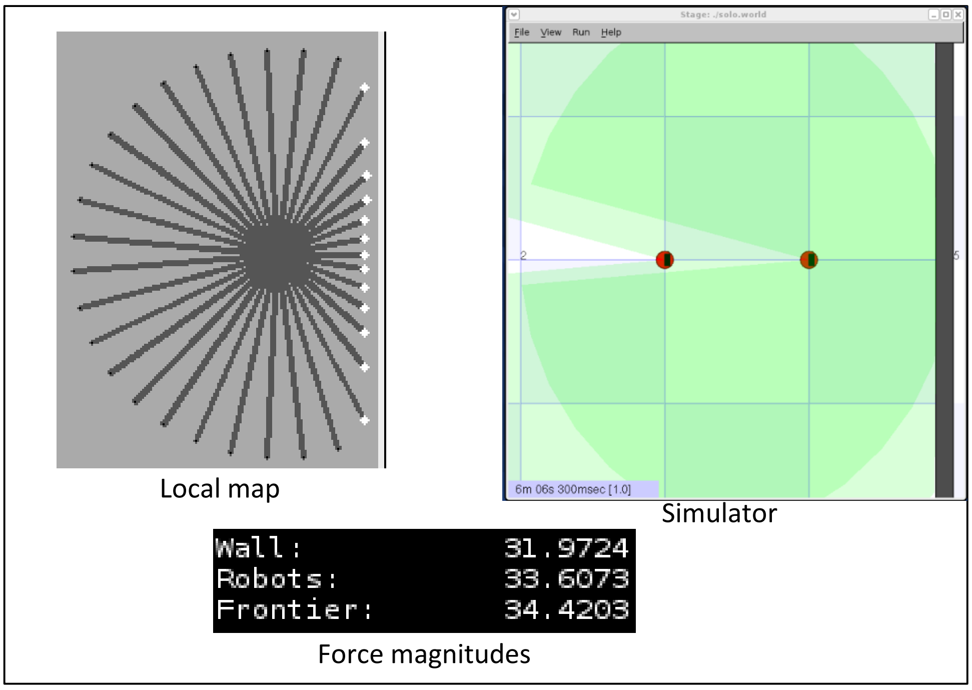
Calculate virtual forces

The virtual force from each object onto the robot is calculated using the following vector formula:

$$\overrightarrow{|F|} = \frac{Q}{d^2}$$

Equation 1

Where $\overrightarrow{|F|}$ is the amplitude of the virtual force vector, *Q* is the constant point charge of the wall, robot or frontier object and *d* is the Euclidian distance from the robot to the object. The polarity of *Q* was set to make wall and robots give off repulsive forces while the frontiers were attractive. The bearing of $\vec{F}$ is equal to the bearing of the object from the robot's perspective. This effect is similar to the gravitational effect of nearby massive objects or the electromagnetic force from charged point particles. A separate charge value, *Q*, was selected for each of the three object types. These values were fine-tuned manually to give approximately equal influences so no force would dominate the robot's behaviour. The values were chosen so a robot at 1m, a semicircle frontier at the maximum ranger range of 2m and a perpendicular wall at 1m generated forces of equal magnitude. A robot experiencing three forces of equal magnitude is shown in Figure 9.



```
Wall:          31.9724
Robots:        33.6073
Frontier:      34.4203
```

Figure 9 - A robot (right in simulator) experiencing three forces of equivalent magnitude from a wall, neighbour and frontier (darkest points in map on left)

Set robot's motion from motion vector

After all virtual forces have been summed and the result normalised to have an amplitude of 1, the following formula is used to calculate the robot's new rotational and forward speeds:

$$Forward\ Speed = M_x K_{FWD} + B_{FWD}$$

Equation 2

$$Rotational\ Speed = M_y K_{ROT}$$

Equation 3

M is the motion vector, calculated by normalising the sum of virtual forces. Vector *M* consists of perpendicular components in the robot's forward direction, M_x, and to the robot's left, M_y. Factors K_{FWD} and K_{ROT} scale these vector components to forward and rotational motions with range: 0 to 0.1 m/s, and 0 to 0.5pi rad/s respectively. The bias factor B_{FWD} introduces a constant forward tendency of 0.15 m/s. This gives the robot a forward motion even when forces are balanced and results in a backwards virtual force causing the robot to slow down rather than immediately reverse. These factor values were set using manual fine tuning to produce movements that moved consistently and evenly without rotational oscillations (from too large K_{ROT}) or collisions (from too small K_{ROT}).

Calculate and print exploration percentage and simulation time step

The simulation step time, indicating the simulated seconds that have elapsed since the last control loop, was requested directly from the Player/Stage engine. The percentage of explored map was calculated by counting non-zero values (explored cells) on the combined global map. This value was divided by the total number of non-wall values giving a percentage of map exploration with 100% indicating the swarm has explored every open space in the simulated environment. These values were printed and logged for analysis.

Results

Expectations

The algorithm was designed to adapt the established swarm control principle of virtual forces with the additions of frontier seeking and map sharing. Because of this origin it was expected that the simulated robots would exhibit behaviour similar to other virtual forces models but with increased map coverage efficiency. This included an even dispersion of the swarm, from the initial cluster to a maximal spacing with even map coverage.

The frontier seeking tendencies were expected to cause robots to move towards unexplored map areas in preference to explored and the map sharing in local proximity was included to increase exploration efficiency by overwriting frontier points from one robot with explored data from other robots in the local area, thus eliminating the attractive influence towards areas explored by other robots. It was thought that this effect of increased mapping efficiency would still be measurable even when map sharing was limited to robots within a limited proximity because robots in the local area would be the most likely to have mapping data from that local area.

Combining these design intensions with the swarm principles of scalability and flexibility the measurements of exploration rate were expected to be:

- Faster for larger swarms in similar situations because of the benefit from swarm parallelism and scalability.
- Faster for frontier seeking methods than non-frontier seeking alternatives because of the goal-oriented influence.
- Faster for methods with the least restrictions on map sharing as robots can refine the goals of their neighbours.
- Effective for covering a range of environment types because of the non-specific algorithm leading to flexible emergent behaviour.

Measurements

Tests were performed in the Player/Stage simulation engine, running on a desktop computer. The simulation engine was slowed down to allow a constant simulated time of 200ms±100ms between each update cycle of the algorithm. The update cycle included updating sensor and mapping data and processing the next movement for each robot. This was done to produce the same effect with the sequential simulator as real robots running local processing in parallel.

Each variation of the algorithm, map and number of robots was run separately and the results were automatically logged by the computer. The data recorded for each experiment were the percentage of the floor space that had been explored by the combined swarm and the simulation time that had passed since the algorithm started, recorded every processing cycle. This data was collated in Microsoft Excel after which data plots were generated.

All the experiments were repeated on three different maps which were all scaled in the simulation to 10m × 10m. The image files with the mapping information can be obtained from the WSRNLab's Git repository and are also pictured in Figure 10. The reasons behind selecting these layouts are:

- Square: An open arena with no obstacles to test dispersion behaviour. This can also be used as a best-case scenario for the mapping efficiencies.
- Rooms: A simple floorplan for evaluating the algorithms in an indoor urban layout. This was chosen to assess the appropriateness of applying the swarm to an urban search and rescue situation.

- Caves: Clumps of unevenly shaped obstacles. This design was taken from the Player/Stage source code and is used in many other swarm research projects that use Player/Stage. This map tests the algorithm in an uneven, varied environment and may make comparisons to other Player/Stage research easier due to its widespread use.

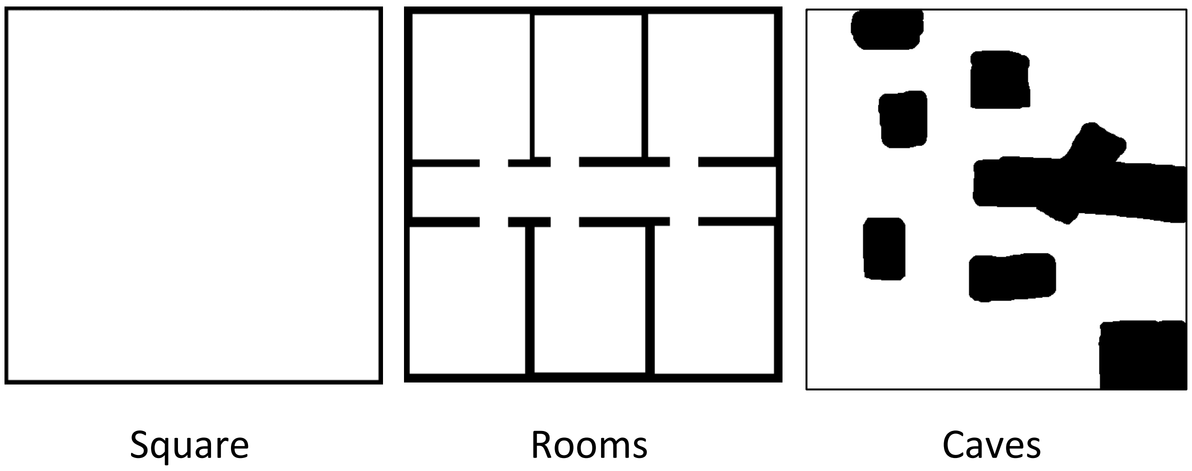


Figure 10 - The three different maps used in the simulations

All experiments began with the robots clustered in the bottom left corner (Figure 11) and were continued until the map was completely explored or until all the robots were stuck in small circular movements. Some screen-shots of interesting states were also recorded during the experiments.

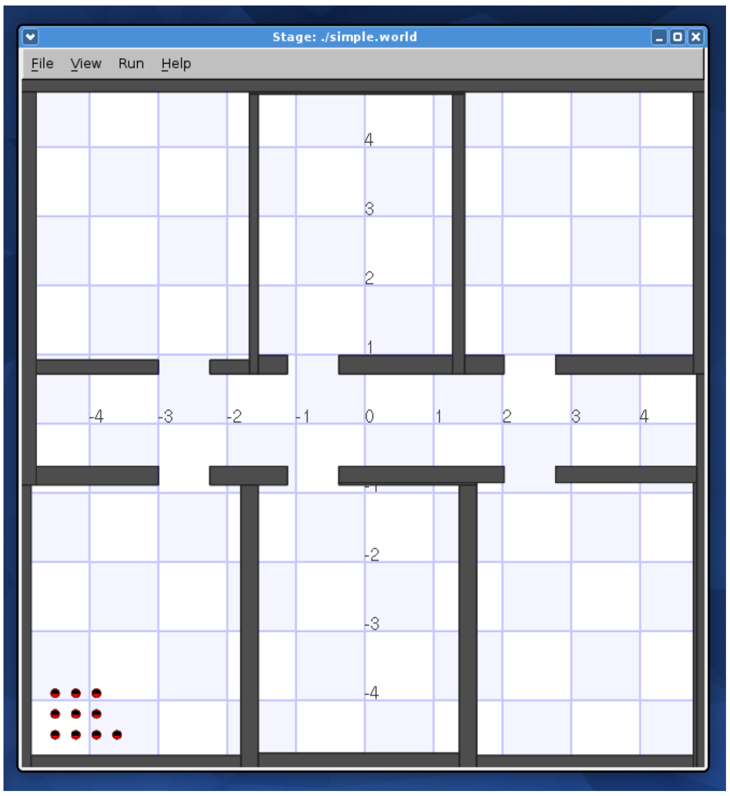


Figure 11 - 10 robots arranged in the stating position

Analysis

Effect of frontier seeking on exploration rate

Several observations were made by studying the exploration over time data and by watching the emergent behaviour during the simulations. Firstly, the established virtual forces method with no frontier consideration was compared to the modified version with frontier detection. Neither of these methods involved map sharing between individuals in the swarm. For swarms with no frontier influence this produced the expected swarm dispersion patterns found in other studies. On the square map the robots moved away from each other and eventually settled in evenly spaced positions around the open area (Figure 12). This lead to fast and efficient map coverage for both the 10 and 5 robot swarms. The final state with the robots evenly spread about the map would be ideal for applications where robots will be used as a networked senor array after the mapping phase.

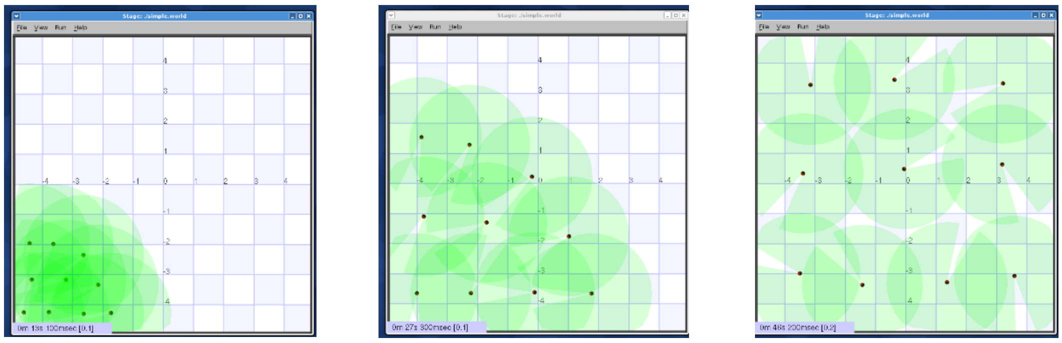


Figure 12 - Even dispersion and settling of a 10 robot swarm with no frontier seeking

The smaller swarms settled before completely exploring the map resulting in the 1, 2 and 5 robot swarms without frontier seeking reaching a maximum exploration size of less than 100% on the square map. The introduction of frontier seeking pulled the robots away from this even formation and settled state and towards the unexplored areas producing better final exploration coverages. For the 2 robot swarm on the Square map this was 100% final coverage where the non-frontier seeking 2 robot swarm settled in diagonally opposite corners with only 62% coverage. The 5 robot swarm with frontier seeking did not cover the entire square map, missing some corner area, but still covered more than the non-frontier seeking counterpart. The exploration coverage over the simulation time is shown graphically in Figure 13.

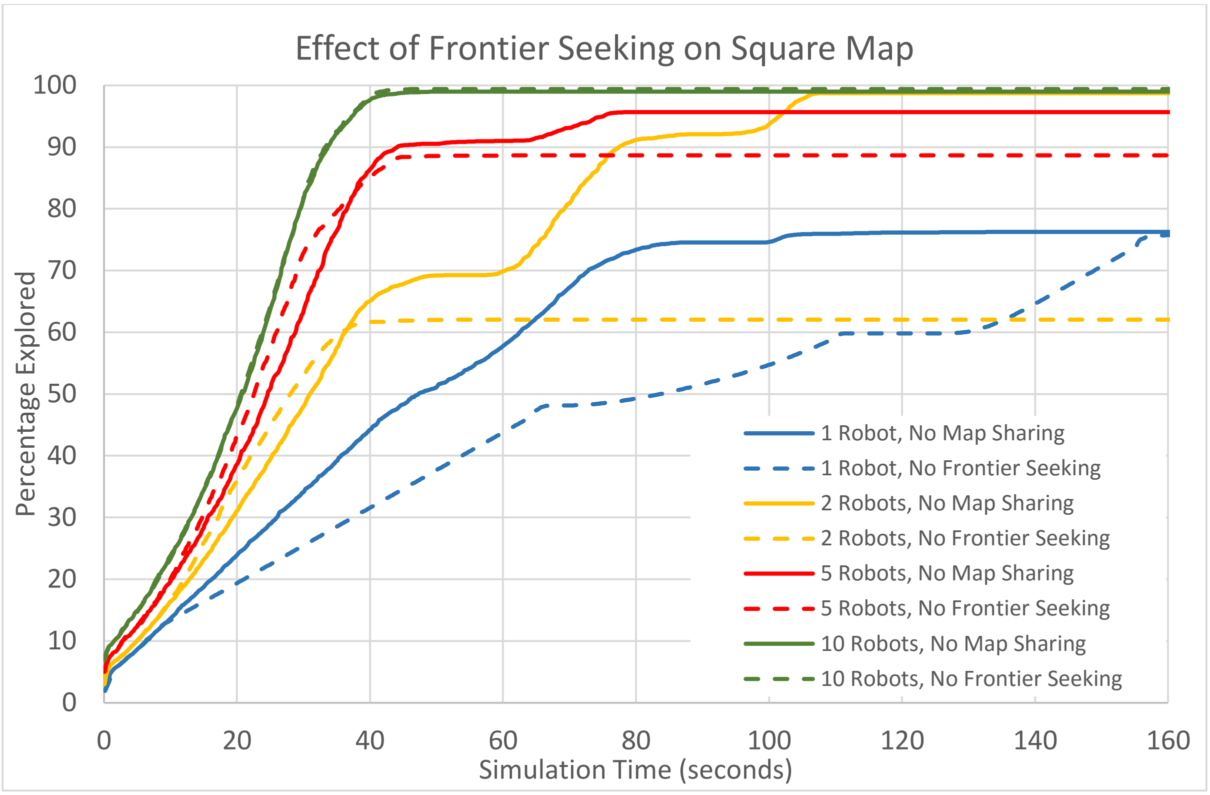
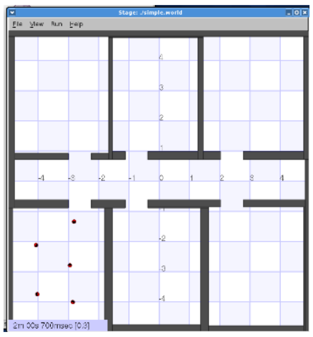
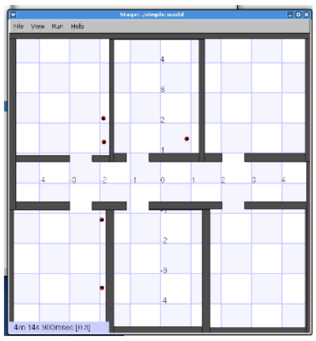


Figure 13 – Exploration percentage over time for different sized robot swarms on the square map

This effect was especially significant for the Rooms map (Figure 15) with 1, 2 and 5 robots where non-frontier seeking robots settled at below 20% map coverage, usually without any robots leaving the starting room, while the frontier seeking robots managed to explore three or more rooms covering double to triple the area. This was probably due to the difficulty for non-frontier seeking robots to move through doorways. The repulsive force from both walls around a narrow doorway often caused robots to move away from the doorway (Figure 14). The addition of frontier seeking forces pulled some robots through resulting in better map coverage.



Non-frontier seeking swarm settles without leaving initial room



Frontier seeking robots are pulled through doorways to explore other rooms

Figure 14 - Comparison of settling states of non-frontier seeking and frontier seeking 5 robot swarms in the rooms map

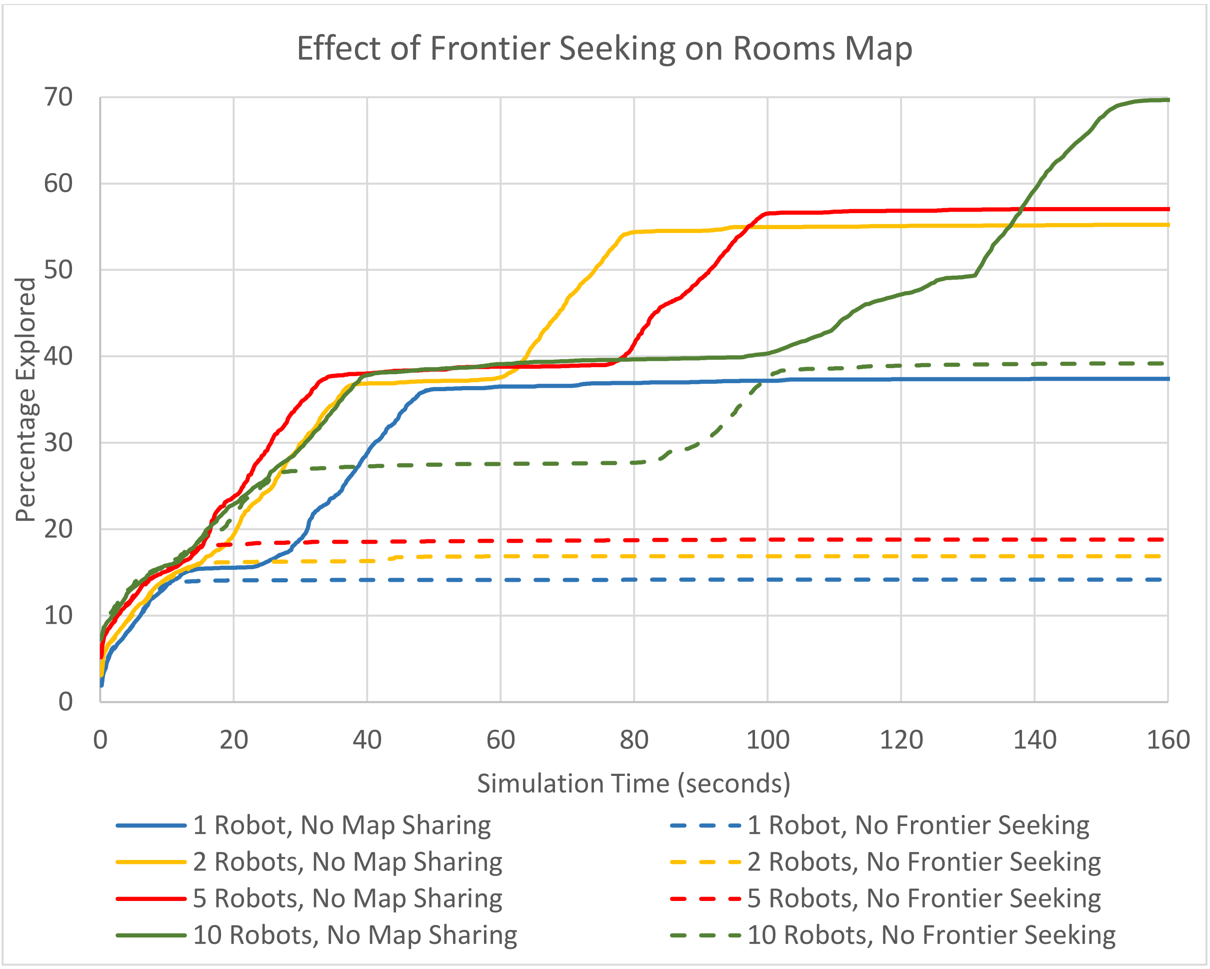


Figure 15 – Exploration percentage over time for different sized robot swarms on the rooms map

The narrow areas in the Caves map had a similar effect, causing all non-frontier seeking swarms to settle with less than 75% map coverage while the 2, 5 and 10 robot swarms with frontier seeking behaviour all reached over 90% map coverage. The single robot case settled before covering the entire map, even with frontier seeking forces, as the combination of obstacles and frontiers happened to form ‘sinks’ where forces were balanced and the single robot settled into a small rotational motion (Figure 16). The final coverage was still an improvement from 20% with one non-frontier seeking robot to 42% coverage with frontier seeking on the Caves map. A visual comparison of these cases on the Caves map is in Figure 17.

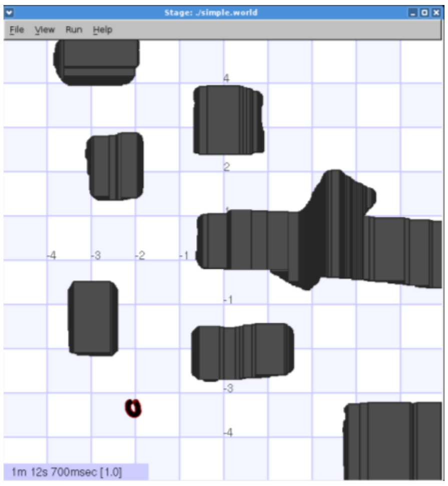


Figure 16 - The circular path of a single robot stuck in a 'sink' of evenly balanced forces

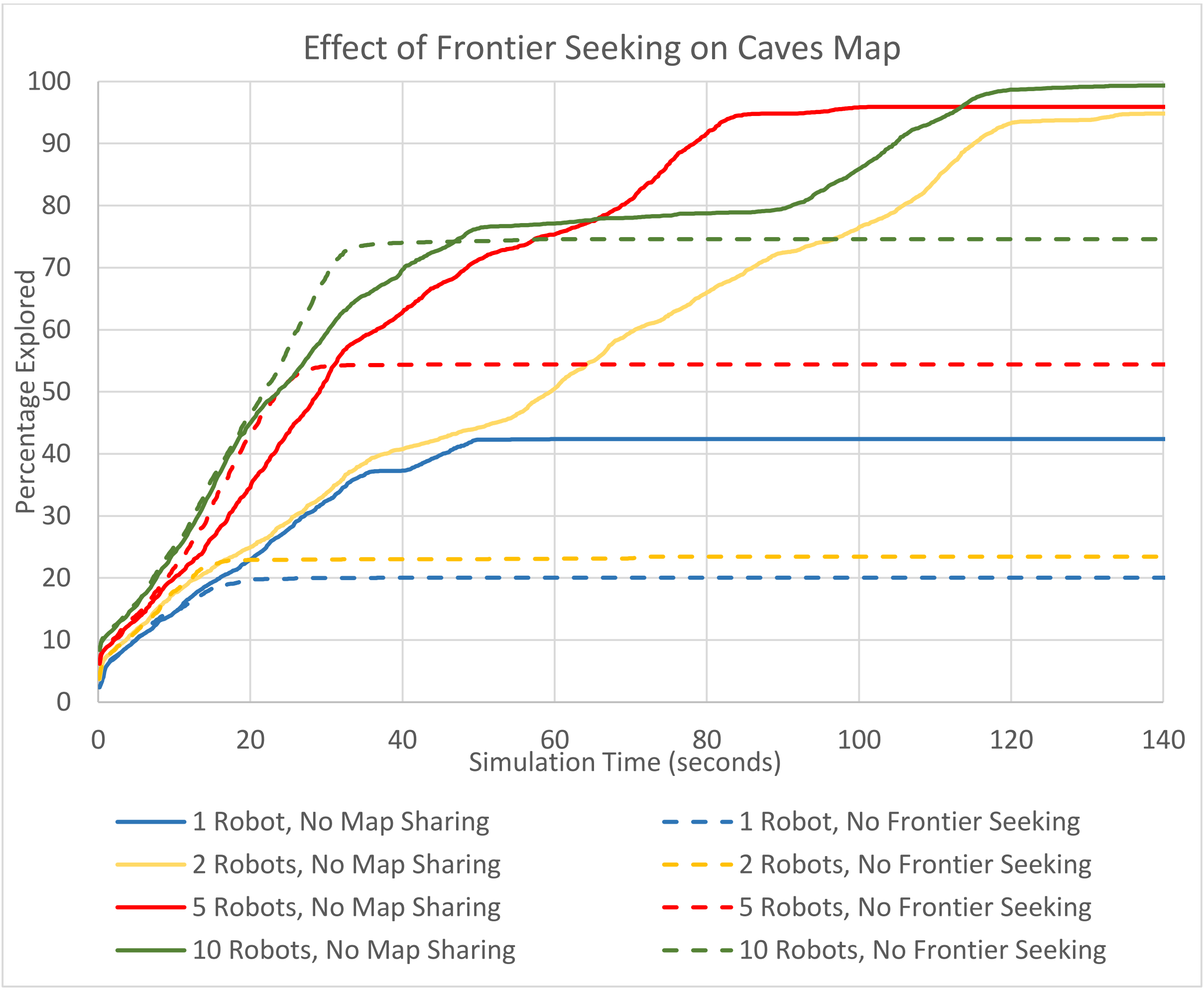


Figure 17 – Exploration percentage over time for different sized robot swarms on the caves map

Scalability and parallelism

The parallel nature of the swarm was assessed by comparing swarms of size 1, 2, 5 and 10 robots. The steepness of the curves in Figure 13 indicate the exploration rate over time on the Square map. Looking at the initial stages of the simulation, from 0 to 40 seconds, with the frontier seeking swarms, the curves get steeper as the number of robots in the swarm increases. This is a product of the parallel nature of the algorithm meaning when more robots are working simultaneously the map can be covered more quickly. This is a desired property for swarm applications as a range of coverage rate requirements can be met by varying the number of robots involved in the solution. A table showing the rate of exploration as percentage points per second, can be calculated from a line of best fit for the first 40 seconds of each data set. This is recorded in Table 1.

Table 1– Initial exploration rates for frontier seeking swarms of various sizes

(% per sec)	**Square Map**	**Rooms Map**	**Caves Map**
1 Robot	1.0	0.5	0.9
2 Robots	1.6	0.8	0.9
5 Robots	2.1	0.9	1.5
10 Robots	2.6	0.7	1.6

This table (Table 1) shows the significant gain in using larger swarms for the initial exploration of the Square and Caves maps. The Rooms map shows very little improvement in increasing the swarm size beyond 2 robots for this stage. This is most likely because the first 40 seconds on this map are spent exploring the starting room and the room opposite which takes approximately the same time to complete regardless of the swarm size as a single robot's 2m radius range sensors can cover an entire room almost simultaneously. This indicates that the application of this swarm algorithm is not well suited to floor plans similar to the Rooms map as there are no significant benefits in using a swarm over one or two robots for initial exploration rate.

On the other hand, the Square and Caves maps show a significant benefit to increasing the swarm size. The application of this frontier seeking algorithm on a searching swarm in mostly open areas is promising. Testing more map layouts could give a better indication as to which specific map features this method maintains the parallelism benefit and which hinder it.

The simulation software could not practically process swarms larger than 10 robots in the current set-up. The scalability of the algorithm can be assessed by evaluating which swarm sizes successfully complete the goals. For both the Square and the Caves maps the swarms of size 2, 5 and 10 all explored over 90% within 120 simulated seconds. Using this as a benchmark for acceptable behaviour, since this would reduce a search and rescue load by 90% in a short time, the method can be marked as scalable up to at least 10 robots on these maps. More experiments should be done to evaluate the scalability properties for even larger swarm sizes as there may be a size that causes collisions or overcrowding that stop the method from being effective.

Map sharing techniques

The frontier seeking algorithm was tested with four different map sharing techniques. These variations combined the frontier seeking with the ability for robots to transfer map data wirelessly either to neighbours within a 1m or 3m radius, globally, or not at all. This variation was selected to simulate how wireless technology could improve or limit the effectiveness of the search. The global map sharing allows all robots to access a commonly shared map. This is not yet possible to implement in a saleable way but advances in cloud processing and network technology move towards this ideal making it an appropriate theoretical best-case. The limited map sharing methods simulate a limited wireless networking range where robots can only reach the signal strength and bandwidth when within a set proximity. This study did not consider the effect of environment obstacles on signal strength or the concept of complex message passing or networking beyond peer-to-peer.

The results of these techniques, displayed in Figure 18, Figure 19 and Figure 20, did not reveal any obvious significant advantage to using a particular map sharing method for exploration. With the open Square map (Figure 18) the optimal global map sharing method had no significant benefit over the method with no map sharing. Since the additional requirement of map sharing corresponds to additional communication hardware and costs, it is more cost effective to use the algorithm without map sharing.

Curiously, the proximity sharing techniques had worse exploration performance than the method without map sharing. The exact cause of this behaviour is unknown but could be due to some random or chaotic nature in the algorithm or simulation system.

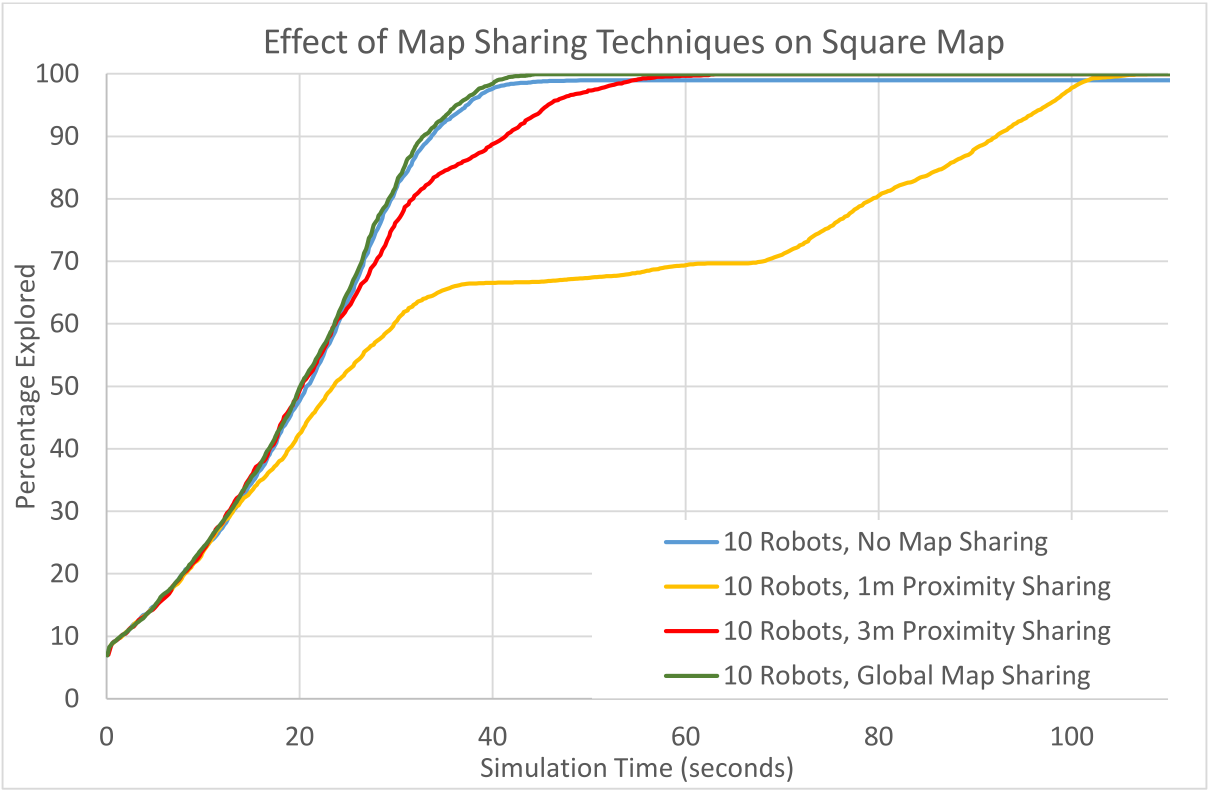


Figure 18 – Exploration percentage over time for different map sharing techniques on the square map

The other maps, Rooms and Caves, show some small advantages for using map sharing with a slightly faster exploration rate on the Caves map (Figure 20) and a higher settling value on the Rooms map (Figure 19). Additionally, the 3m proximity limitation produces a faster map coverage than the 1m limitation on both of these maps. However, the unusual result of the 3m limitation outperforming the global (unlimited) map sharing on the Rooms map and the underperformance of the 1m limitation compared to no amp sharing on the Caves confirm that there is more to discover about the behaviour of the swarm. It is possible that some chaotic factors are causing varied results on this particular scale making direct map-sharing effects impossible to identify. This can be investigated by running a larger number of simulations.

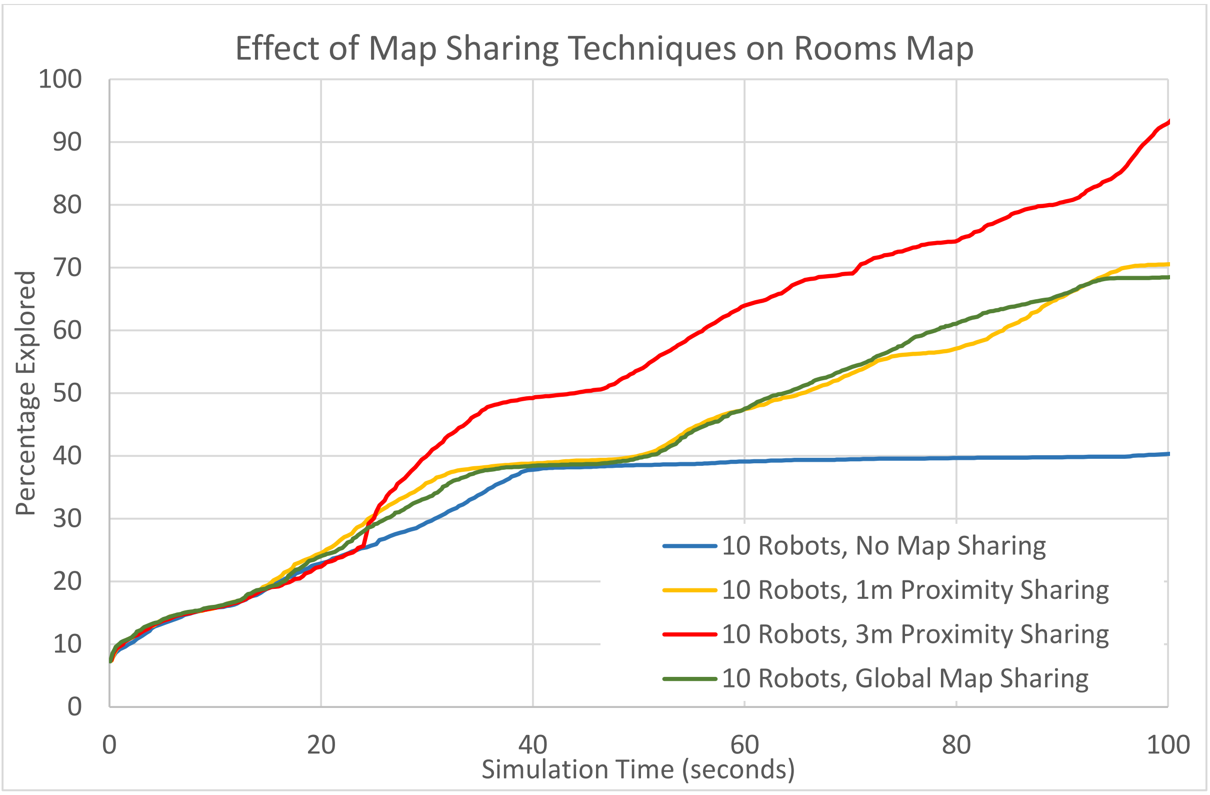


Figure 19 – Exploration percentage over time for different map sharing techniques on the rooms map

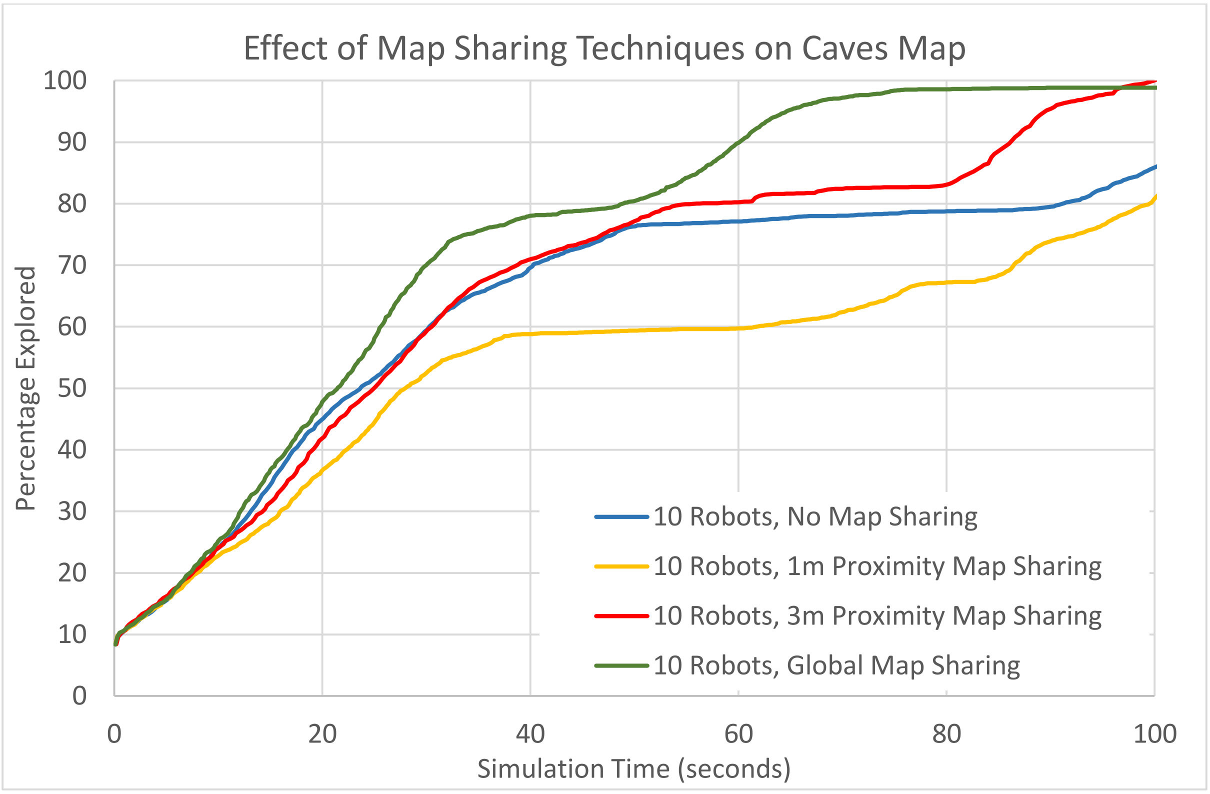


Figure 20 – Exploration percentage over time for different map sharing techniques on the caves map

Limitations

Several limitations were identified before and during the project. The most significant of these are listed here along with remedies for future work to consider:

- The effect of map sharing on exploration rate was inconclusive. Performing a very large number of simulations may identify the range of the stochastic nature in the algorithm and simulation software. An in-depth probability analysis on the algorithm may also be useful for establishing the distribution of possible results.
- Real-life robotics includes several attributes that were not simulated. These include sensor noise, wheel slippage, battery power effects, processor limitations and communication limitations. Adding more features to the simulation could give helpful results for some of these issues, like simulated sensor noise, but it is recommended to run similar tests on actual physical hardware. This practice of following simulations with physical tests is common in swarm robotics research and practically validates simulation results.
- The Player/Stage software was not designed in this project and as such was trusted to be accurate and effective based on its widespread use in respectable research. The software and hardware used must have limitations of some form and these were not analysed. Testing the algorithm on alternate simulation software and hardware will highlight any platform dependencies that effect results, as will running the algorithm on real life hardware.
- Swarm algorithms often suffer from an upper bound on scalability where congestion, collisions or other factors significantly hinder performance. With the swarms of up to 10 robots this was not reached and with the current simulation software and hardware these tests could not be easily conducted. An optimising of the software or an upgrading of simulation hardware would enable testing these limits.
- While a range of environments were tested the results showed that the map has a significant impact on the swarm behaviour. More environments should be tested to identify specific layout features and their effects on the emergent behaviour before the algorithm is applied to search and rescue.
- Other swarm behaviours such as robustness and flexibility were only tested from some of the many applicable perspectives. Different types of simulations should be done to examine robustness and flexibility under dynamic environments, individual robot failures and external agent influence as these are all relevant to real search and rescue events.

Conclusions

The frontier seeking modification to the virtual forces swarm control algorithm produced effective map exploration for covering over 90% of empty and open maps in less time than the non-frontier seeking algorithm. Difficulties were identified for maps that include narrow spaces which caused inefficient coverage and unsatisfying end states. The algorithm was shown to scale to 10 robots with increasing efficiency as the swarm size was increased. Map sharing variations were applied with results showing some stochastic nature making it impossible to draw further conclusions about the efficiency of these variations with the current sample size.

Limitations were identified and some suggestions for further work were mentioned with the aim of coming closer to creating a swarm robotics system for search and rescue. The most immediate next steps to build upon this project would be:

- More simulation testing to address limitations and confirm conclusions presented here.

- Development of algorithm innovations, such as path-finding, to overcome narrow doorway inefficiencies and explore alternate solutions.
- Real-world hardware testing to identify other significant factors that were not simulated.

Project Conclusions

The project succeeded in improving the WSRNLab's eBug swarm robotics research platform and produced useful observations about emergent behaviour of swarm robots under a new control algorithm. The process of rapid prototyping with 3D printing produced a new eBug chassis that was lighter and enabled easy battery access while maintaining the WSRNLab's open source and accessible philosophy. The endeavour also showed that the method is an effective design strategy while highlighting many of the benefits and difficulties that were encountered.

The use of Player/Stage for simulating a new frontier seeking algorithm and its map-sharing variations showed that frontier seeking can improve the map exploration rate for swarms in open areas. Map sharing tests showed that chaotic nature requires larger sample sizes and stochastic analysis before practical conclusions can be made.

Limitations of the simulation-based analysis were identified including the inability to test swarm sizes beyond 10 robots and the lack of real-world noise and inconsistencies. This meant that scalability limits were not found and that many robustness factors are yet to be identified. Algorithm testing on physical hardware is the recommended next step and the portability of the Player/Stage system should ease this endeavour. Further simulations will also confirm presented conclusions and allow analysis of emergent behaviour with unidentified causes.

References

[1] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence 7(1),* pp. 1-41, 2013.

[2] L. Locchi, D. Nardi and M. Salerno, “Reactivity and Deliberation: A Survey on Multi-Robot Systems,” *Lecture Notes in Computer Science: Balancing Reactivity and Social Deliberation in Multi-Agent Systems,* pp. 9-32, 2013.

[3] E. Şahin, “Swarm Robotics: From Sources of Inspiration to Domains of Application,” *Lecture Notes in Computer Science: Swarm Robotics,* pp. 10-20, 2005.

[4] G. Beni, “From swarm intelligence to swarm robotics,” *Swarm robotics,* pp. 1-9, 2005.

[5] N. R. Hoff III, A. Sagoff, R. J. Wood and R. Nagpal, “Two Foraging Algorithms for Robot Swarms Using Only Local Communication,” in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, Tianjin, 2010.

[6] C. W. Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model,” *COmputer Graphics, Vol 21, No 4,* pp. 25-34, 1987.

[7] N. R. Hoff III, A. Sagoff, R. J. Wood and R. Nagpal, “Two Foraging Algorithms for Robot Swarms Using Only Local Communication,” in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, Tianjin, 2010.

[8] D. Puig, M. Garcia and L. Wu, “A new global optimization strategy for coordinated multi-robot exploration: Development and compataive evaluation,” *Robotics and Autonomous Systems,* pp. 635-653, 2011.

[9] K. Senthilkumar and K. Bharadwaj, “Multi-robot exploration and terrain coverage in an unknown environment,” *Robotics and Autonomous Systems,* 2012.

[10] N. P. Tani, A. Blatt, D. A. Quintn and A. Gopinathan, “Optimal cooperative searching using purely repulsive interactions,” *Journal of Theoretical Biology,* pp. 159-164, 2014.

[11] A. Dirafzoon, J. Betthauser, J. Schornick, D. Benavides and E. Lobaton, “Mapping of Unknown Environments using Minimal Sensing from a Stochastic Swarm,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, Chicago, IL, USA, 2014.

[12] M. S. Couceiro, D. Portugal and R. P. Rocha, “A Collective Robotic Architecture in Search and Rescue Scenarios,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, New York, USA, 2013.

[13] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. P. and M. Dorigo, “Self-organized flocking with a mobile robot swarm: a novel motion control method,” *Adaptive Behavior,* pp. 1-18, 2012.

[14] S. Hou and C. Cheah, “Dynamic compound shape control of robot swarm,” *IET Control Theory and Applications,* pp. 454-460, 2011.

[15] G. K. Fricke, K. M. Lieberman and D. P. Garg, “Swarm formations under nonholonomic and numerosity constraints,” in *ASME 5th Annual Dynamic Systems and Control Conference joint with the JSME 2012 11th Motion and Vibration Conference*, Fort Lauderdale, Florida, USA, 2012.

[16] D. Krupke, M. Ernestus, M. Hemmer and S. P. Fekete, “Distributed Cohesive Control for Robot Swarms: Maintaining Good Connectivity in the Presence of Exterior Forces,” *arXiv preprint arXiv:1505.03116,* pp. 1-8, 2015.

[17] A. E. Turgut, H. Çelikkanat, F. Gökçe and E. Şahin, “Self-organized flocking in mobile robot swarms,” *Swarm Intelligence,* pp. 97-120, 2008.

[18] Y. Liu and G. Nejat, “Robotic Urban Search and Rescue: A Survey from the Control Perspective,” *Journal of Intelligent & Robotic Systems,* pp. 147-165, 2013.

[19] M. S. Couceiro, P. A. Vargas, R. P. Rocha and N. M. Ferreira, “Benchmark of swarm robotics distributed techniques in a search task,” *Robotics and Autonomous Systems,* pp. 200-213, 2013.

[20] C. Luo, A. P. Espinosa, D. Pranantha and A. D. Gloria, “Multi-robot search and rescue team,” in *Proceedings of the 2011 IEEE International Symposium on Safety,Security and Rescue Robotics*, Kyoto, Japan, 2011.

[21] M. Bodi, R. Thenius, M. Szopek, T. Schmickl and K. Crailsheim, “Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST,” *Mathematical and Computer Modelling of Dynamical Systems,* pp. 87-100, 2012.

[22] A. Levy, C. Keitel, S. E. and J. McLurkin, “The Extended Velocity Obstacle and Applying ORCA in the Real World,” *IEEE International Conference on Robotics and Automation (ICRA),* pp. 16-22, 2015.

[23] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni and M. Birattari, “AutoMoDe: A novel approach to the automatic design of control software for robot swarms,” *Swarm Intelligence,* pp. 89-112, 2014.

[24] A. Reina, M. Salvaro, G. Francesca, L. Garattoni, C. Pinciroli, M. Dorigo and M. Birattari, “Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks,” in *Adaptive Hardware and Systems (AHS), 2015 NASA/ESA Conference on.*, Bologna, Italy, 2015.

[25] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari and e. al., “Swarmanoid, A Novel Concept for the Study of Heterogeneous Robotic Swarms,” *IEEE Robotics & Automation Magazine ,* pp. 60-71, 2013.

[26] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. Pinciroli and M. Dorigo, “Self-organized flocking with a mobile robot swarm: a novel motion control method,” *Adaptive Behaviour 20(6),* pp. 460-477, 2012.

[27] V. Trianni and A. Campo, “Fundamental Collective Behaviors in Swarm Robotics,” *Swarm Intelligence in Optimization and Robotics,* pp. 1377-1391, 2015.

Appendix A – 3D Printed Chassis Design Change Log

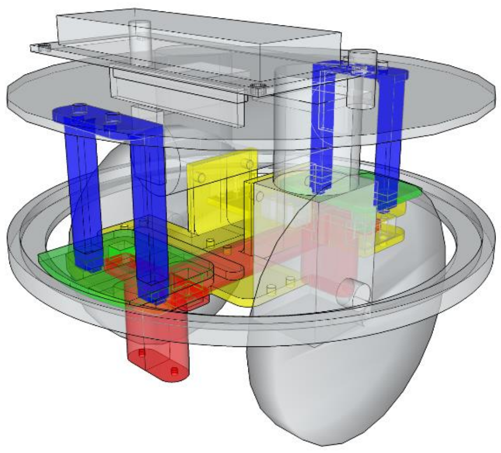


Figure 21- Reference image for 3D printing change log

Part numbers/colours: Part 1: BLUE, Part 2: YELLOW, Part 3: GREEN, Part 4: RED

- 4/8/2015 (Matthew Boys)
    - Part 1: Changed to print on side so printing layer weakness doesn't affect pins which kept snapping off
    - Part 1,2,3,4: Changed all pins from cylinders to 50x50mm square prisms to match new Part 1
    - Part 1,2,4: Added some chamfer on the corners that experience most stress for stability and to reduce flex/wobble
- 13/7/2015 (Matthew Boys)
    - Bumper: arms breadth reduced from 45mm to 40mm to match physical version
    - Bumper: Inner thickness reduced from 2mm to 1.7mm
    - Sensor: New part to match physical piece. Dimensions from http://www.switches-manufacturer.com/DM3-02P.htm
    - Part 2: Centre central bump sensor screw holes so screws will not collide with part 4
    - Part 2: Enlarge all bump sensor screw holes from 1.8mm to 2mm diameter
    - Part 2: Remove notch from wall as no-longer needed
- 16/6/2015 (Matthew Boys)
    - Part 2: Move screw holes for central bump sensors towards the centre axis by 3mm to
    - Part 2 & 3: Shaved 1mm from the wall the retracting bumper hits to allow more bumper movement. From both top and bottom pieces
    - Part 1: Reduced the pentagonal pin diameter from 4mm to 3.8mm to allow better fit into hole (was too tight)
    - Part 2: Adjusted motor wall cut-out to be 6mm x 3mm with triangle above for printing stability and avoid collision with sensors.
    - Part 3: Extended the top bumper guide out by 7mm to reduce wobble of bumper.

- 7/6/2015 (Matthew Boys)
    - Part 2: Enlarged screw holes for all bump sensors from 1.5mm to 1.8mm diameter
    - Part 3: Cut edges of plate and bumper guide down to a maximum radius of 50mm to avoid hitting the bumper
    - Part 2: Moved screw holes of central two bump sensors back 2mm to allow more bumper movement
    - Part 2: Added cut-out in motor wall for wires from bump sensors
    - Part 4: Added wall between two wheel supports for more stability
    - Part 1 & 2: Moved support columns back 2mm to avoid bumper hitting edge
    - Part 1 & 2: Changed pins on support column from circles to pentagons (so they won't go on backwards by mistake)
- 25/5/2015 (Matthew Boys)
    - Part 1: Wider top screw holes to make screws fit better. Hole diameter changed from 2.5mm to 2.8mm
    - Part 4: Slightly deeper screw holes for attaching feet rollers
    - Part 2: Slightly wider motor support walls to avoid breakages near the screw holes
    - Part 2: Slightly thicker motor support walls to reduce change of breakage
- 18/5/2015 (Matthew Boys)
    - All: Rebuild from scratch at 100x scale to reduce errors in files
    - All: Add rounded corners and screw holes
    - Part 1: Thicker support posts to reduce chance of breakages
- 23/3/2015 (Matthew Boys)
    - Part 3: Import existing file to SketchUp
    - All: Recreate eBug hardware in SketchUp
    - All: Develop initial design in four types of pieces