University of Minnesota - Twin Cities

EmPowering Mobile Robots

Project for CSci 8551 – Intelligent Agents, Fall 2007

by

Pushkar Modi

A. Introduction & Significance

As Robots become smaller and more sophisticated, their effectiveness has increased in a number of areas including surveillance and reconnaissance missions. However, a decrease in size has meant that they are unable to carry surplus battery resources or mechanisms to recharge themselves independently and be truly self-sufficient.

Through this project, I intend to begin work on an algorithm that will allow ‘N’ robots to co-ordinate amongst themselves and maintain constant vigil at ‘M’ observation points for an extended duration of time. The robots, with varying power capabilities, rely on a central base station for recharging purposes.

To achieve true long-term autonomy, the robot must be self-sustaining in its environment [1]. It would be beneficial to have systems where frequent human intervention is not required. In certain scenarios like army reconnaissance missions for example, no human intervention may be possible.

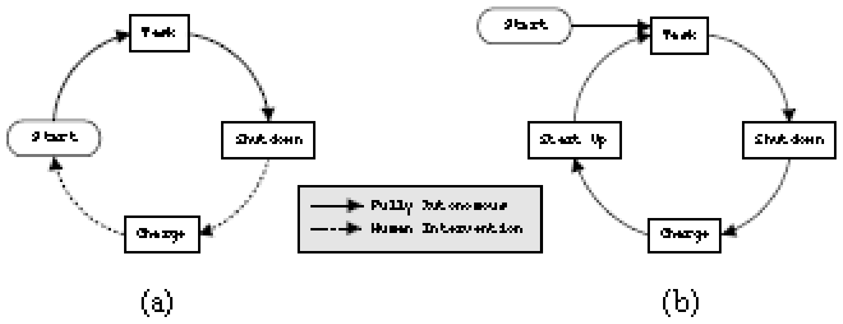


Figure 1: *Robot task cycle comparison.*

Rechargeable batteries are typically used that provide power for only a few hours. Once depleted, the robot/batteries must be connected to a recharger via human intervention. This results in a non-continuous robot task cycle as shown in Figure 1 (a), thereby preventing long-term autonomy.

To achieve true long-term autonomy, the robot must be self-sustaining in its environment. A continuous robot task cycle is then within reach as shown in Figure 1 (b), which can be defined simply as providing the robot with the means to recharge itself, and continue its allotted tasks without human assistance.

B. Past Work

To make robots truly autonomous a number of possible alternatives have been tried out.

Some examples of self-sufficient robots are:

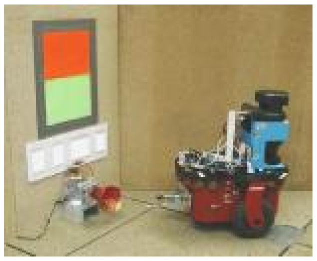
- EcoBot-II - A prototype robot capable of hunting down over 100 slugs an hour and using their rotting bodies to generate electricity is being developed by engineers at the University of West England's Intelligent Autonomous Systems Laboratory. [2]
- Spirit and Opportunity - Twin rover's used by NASA for the Mars Exploration Rover Missions. The rovers have continued to function effectively over fifteen times longer than NASA planners expected. Solar arrays generate about 140 watts for up to four hours per Martian day (sol) while rechargeable lithium ion batteries store energy for use at night. [3]

A number of techniques have been proposed where robots can be supported with their power needs:

- The Tanker Approach - Pawel Zebrowski and Richard T. Vaughan from the Simon Fraser University, British Columbia propose the use of a special-purpose, energy-transporting "tanker" robot that finds and recharges worker robots, extending their working life [4]. However, charging could take a lot of time and the number of supervisor robots needed may turn out to be an overhead.
- Potentially Distributable Energy - Ngo, Trung Dung from Aalborg University proposed robots that are capable of not only self-recharging energy but also exchanging batteries with other robots. Here, each robot has two batteries installed which it can use sequentially. A drained out battery can be swapped with a fully charged battery by other robots. [5] However, this technique requires the robots to be specially designed with spare battery packs.
- Wireless Power – This can be defined as a process where electrical energy is transmitted from a power source to an electrical load, without interconnecting wires. Earlier this year, a team from MIT successfully demonstrated lighting a 60W bulb wirelessly. Although no experiments have been carried out yet for robotic applications, this could be a promising technique going ahead. [6]

An inexpensive, repeatable and reliable docking technique:

This technique developed at the University of Southern California is based on a control architecture and an accompanying recharging mechanism which allows a robot to readily intervene its regular operation with autonomous recharging to stay alive [1]. As seen in the figure, a robot relies on a docking station with a funnel-like design that guides an extended charging node on the robot to an electrical socket.



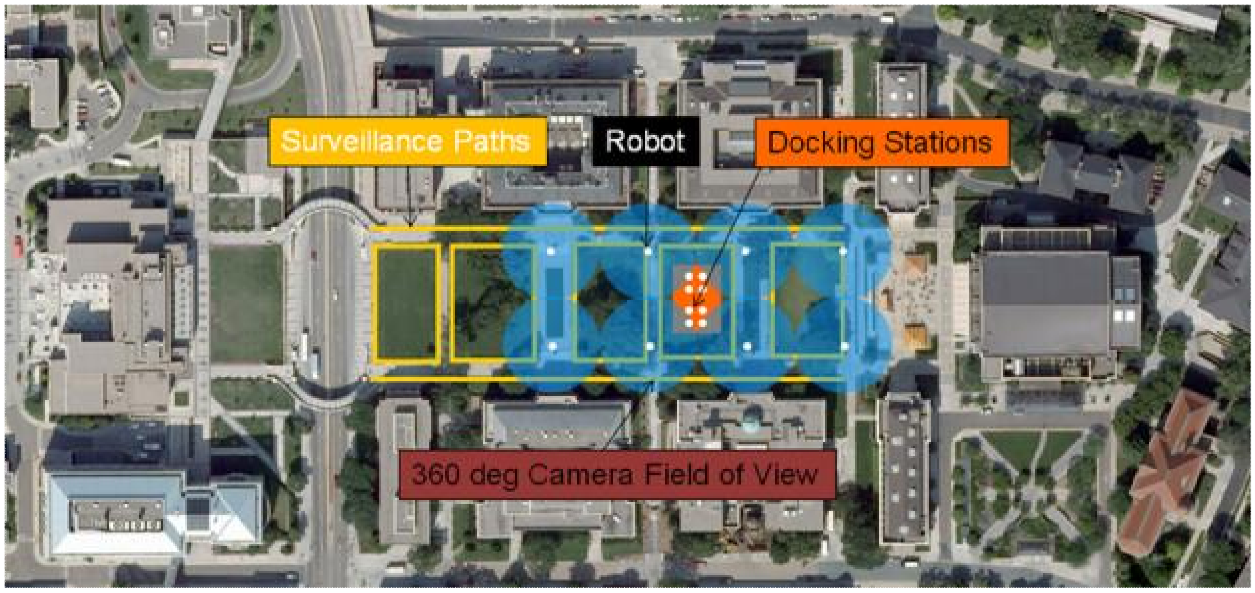
The system is able to operate for long periods of time without operator intervention. Results from 100 trials showed a 99% success rate for mechanical docking, and a 97% success rate for electrical docking.

The one mechanical failure was due to a software bug that was later fixed. The 3% electrical docking errors were centered down to flaws in the design panel, which the authors express confidence in ironing out with some work.

C. Goals

Consider a scenario where multiple cameras / motion sensors must be deployed over an area for surveillance. In situations like these, it may be advantageous to mount these cameras / sensors on mobile robots and have them go to the desired positions.

This would avoid costs and time that would otherwise be spent in setting up / dismantling the infrastructure manually. Also, expenses related to cameras and sensors can be limited by redeploying the same equipment elsewhere.



Consider a set of defined observation points that must be monitored constantly using twice the number of robots.

1. The purpose of this project is to be able to identify and implement an algorithm that ensures that:

   · There is at least one robot present at an observation point at any given time.

   · A stationed robot should be able to detect that it is low on power and send a distress call to the base station will in time for a substitute robot to come and take over before the stationed robot has to leave.

   · When the substitute robot arrives and takes position, the original robot can return to the recharging station and recharge to re-join the system as a substitute robot.

2. Simulate using Player/Stage (or an equivalent simulation environment) an algorithm that implements the chosen approach.

3. Gather data from multiple simulations and use statistical analysis to prove that this theory works well in most cases.

D. Experimental Setup

For the sake of restricting the scope of this project, the assumptions made are:

1. No. of observation positions ‘M’ are optimal and fixed. This assumption is based on existing research as described in Supporting Work.

2. No. of robots ‘N’ is twice the number of observation points i.e. N = M x 2. Although this may seem like an overhead to begin with, I am hoping that with optimization the number of substitute robots can be decreased.

3. There is a central, fixed base station with ‘N’ Docking Points i.e. one dedicated for every robot. The robot and docking stations are equipped with the mechanisms described in the previous section for autonomous docking and recharging.

4. Every robots uses ‘X’ units of power while standing still and twice of that while in motion. If the robot does not reach a docking station before running out of battery supply, it stops in its tracks.

5. A Robot recharges instantaneously on docking. Gradual recharge will allow a more realistic simulation experience as it directly affects the availability of the secondary robots. Although this is a serious assumption, I make it for the following reasons:

    a. Player / Stage cannot simulate real-time battery levels for specific robots and hence representation of the battery levels must happen through code based on verified data.

    b. The purpose of this project is to ensure that primary robot reports a threshold to the base station. Calculation of an optimal threshold would allow the secondary robot to reach the observation point just before the primary robot leaves for a recharge.

    c. An instant recharge is practically possible if an individual simply swaps the battery as soon as a robot arrives at the base station.

6. Robot has enough cameras / sensors for a 360 degree field of coverage. This seems like a practical assumption to make given the purpose at hand. If a change is required here, it would not affect the algorithm in any way.

7. Every robot is capable of communicating with the base station although no communication is required between robots. Communication is required solely for the purpose of sending out distress signals and threshold for ensuring that the substitute robot can reach the observation point in time. In other words, if this algorithm is used for an application where no substitution is required, we could safely do away with the requirement of communication.

8. Other assumptions as required for simulation purposes. Player/Stage has a number of practical and technical limitations which I had to come around during simulation.

E. Supporting Work

- Optimal Camera Placement for Automated Surveillance Tasks [8]
  By Robert Bodor, Andrew Drenner, Paul Schrater, and Nikolaos Papanikolopoulos

  Camera placement has an enormous impact on the performance of vision systems, but the best placement to maximize performance depends on the purpose of the system. As a result, this paper focuses largely on the problem of task-specific camera placement. This paper proposes a new camera placement method that optimizes views to provide the highest resolution images of objects and motions in the scene that are critical for the performance of some specified task

- Allocating Tasks in Extreme Teams [9]
  By Paul Scerri, Alessandro Farinelli, Steven Okamoto and Milind Tambe

  This paper proposes a novel distributed task allocation algorithm for extreme teams, called (Low Communication Approximate Distributed Constraint Optimization (LA-DCOP), which incorporates three key ideas:

  First, LA-DCOP's task allocation is based on a dynamically computed minimum capability threshold which uses approximate knowledge of overall task load. Second, LA-DCOP uses tokens to represent tasks and further minimize communication.

  The third idea is not relevant to this project.

F. Approach

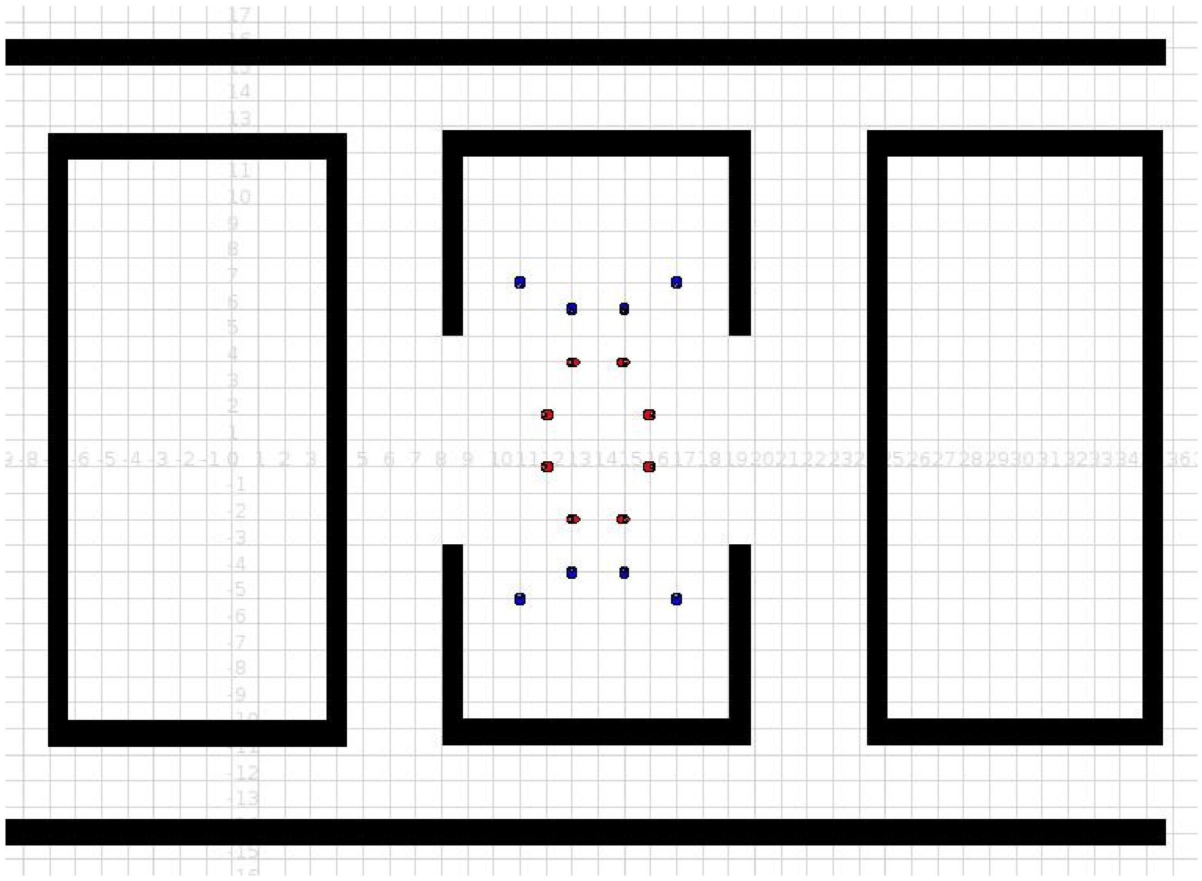
Simulation Environment

Operating System:	Ubuntu v7.10 (pronounced oo-boon-too)
Simulation Environment:	Player v2.0.4 & Stage v2.0.3
Compiler / Editor:	g++ v4.1.3 with gedit
Programming Language:	C++ for Unix with corresponding Player/Stage API's

Although setting up Ubuntu with Player / Stage is a task in itself, the completion does not come with any reqards. Player / Stage is riddled with a complex set of API's without proper documentation or community driven samples. Also, the g++ compiler had certain limitations inspite of having the latest version installed.

Empirical Work (with approach explained inline)

1. Robots are initially placed in defined positions. Red robots go out first. Secondary Robots are robots that seek distress calls and help out other robots.

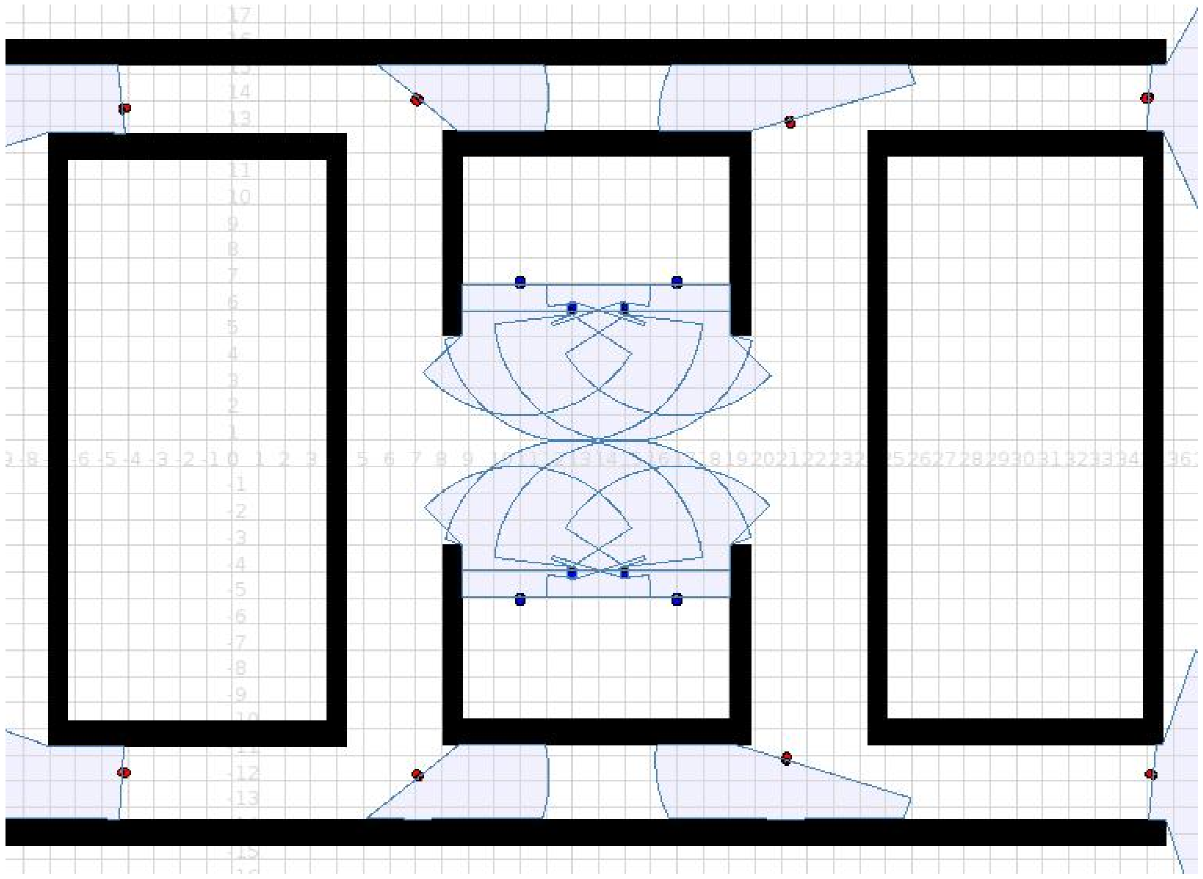


Debug Log when robots start moving out of their initial positions (with their battery levels):

```
Robot1 going from Docking Station 1 to Observation Pt 1. ** Battery: 100
Robot2 going from Docking Station 2 to Observation Pt 2. ** Battery: 100
Robot3 going from Docking Station 3 to Observation Pt 3. ** Battery: 100
Robot4 going from Docking Station 4 to Observation Pt 4. ** Battery: 100
Robot5 going from Docking Station 5 to Observation Pt 5. ** Battery: 100
Robot6 going from Docking Station 6 to Observation Pt 6. ** Battery: 100
Robot7 going from Docking Station 7 to Observation Pt 7. ** Battery: 100
Robot8 going from Docking Station 8 to Observation Pt 8. ** Battery: 100
```

- The placement of the docking points is important to ensure that the robots do not tread over each other while moving in or out. Initially, I had the robots places extremely close to each other but Player/Stage (and I assume the real world) requires some margin for error.
- The robots need some room to manouver turns. Often I see them bumping into walls inspite of an earlier detection by the Laser Sensor for detecting objects. This point must be kept in mind while designing a map for the robots.

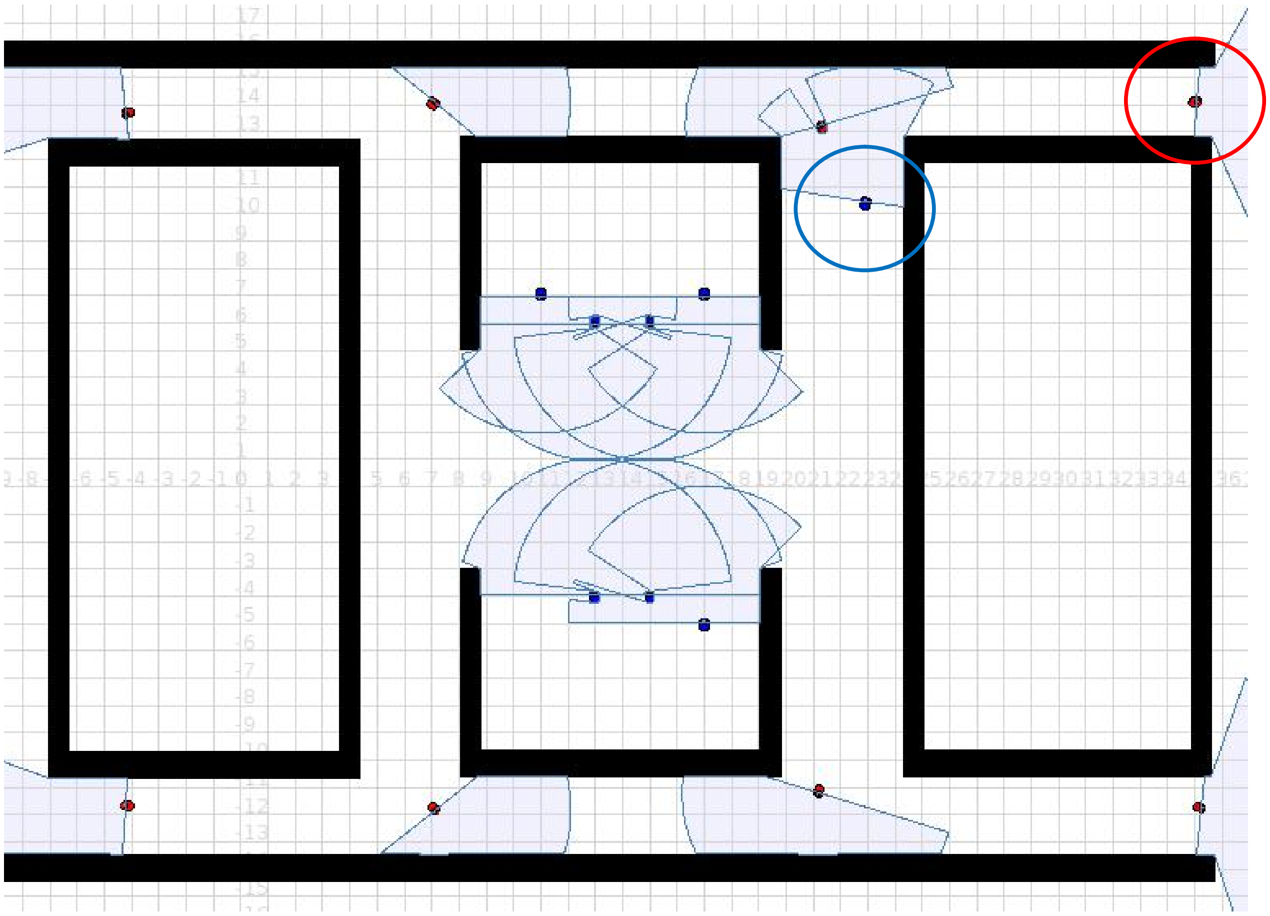
2. Primary Robots after reaching their initial positions. The blue semi-circles indicate a Laser range finder for detecting obstacles. Each incoming robot to an observation point calculates a threshold which acts like a guidance system:
    - For itself – to ensure that it leaves the observation point with enough power left to reach it's docking station.
    - For calculating when a distress call must be sent out so that the secondary robot has enough time to reach the observation point, take over surveillance and relieve the primary robot.
    - As each robot is expected to have varying amounts of power based on their activities and recharging cycles, the threshold is also used by each secondary robot to ensure that it has enough power to go out to an observation point, stay there for a defined time length and come back. By design, the secondary robot skips a distress call if it does not meet the threshold to help.



Debug Log when robots reach their observation points (with their battery levels).

```
Robot7 has reached Observation Pt 7 ** Battery: 88.8993
Observation Pt 7 threshold: 11.1007
Robot3 has reached Observation Pt 3 ** Battery: 87.6992
Observation Pt 3 threshold: 12.3008
Robot2 has reached Observation Pt 2 ** Battery: 87.4992
Observation Pt 2 threshold: 12.5008
Robot6 has reached Observation Pt 6 ** Battery: 87.2992
Observation Pt 6 threshold: 12.7008
Robot5 has reached Observation Pt 5 ** Battery: 83.649
Observation Pt 5 threshold: 16.351
Robot1 has reached Observation Pt 1 ** Battery: 83.549
Observation Pt 1 threshold: 16.451
Robot4 has reached Observation Pt 4 ** Battery: 82.899
Observation Pt 4 threshold: 17.101
Robot8 has reached Observation Pt 8 ** Battery: 78.3987
Observation Pt 8 threshold: 21.6013
```

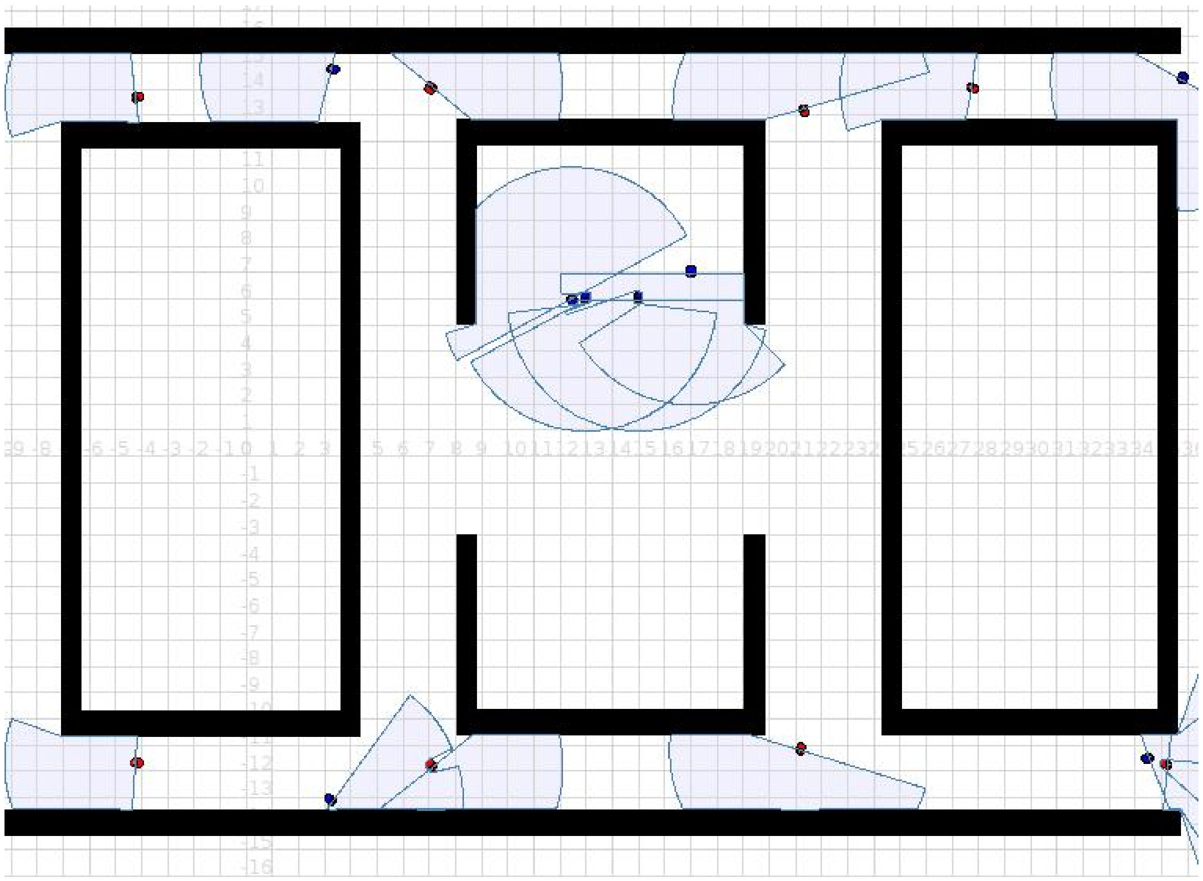
3. An SOS call sent out by one of the primary robots makes a secondary robot go out and help!



Debug Log when a robot sends out a distress (SOS) call:

```
** SOS from Observation Pt 8 ** Battery: 43.1965
Robot9 will rescue Observation Pt 8
Robot9 going from Docking Station 9 to Observation Pt 8. ** Battery: 100
```

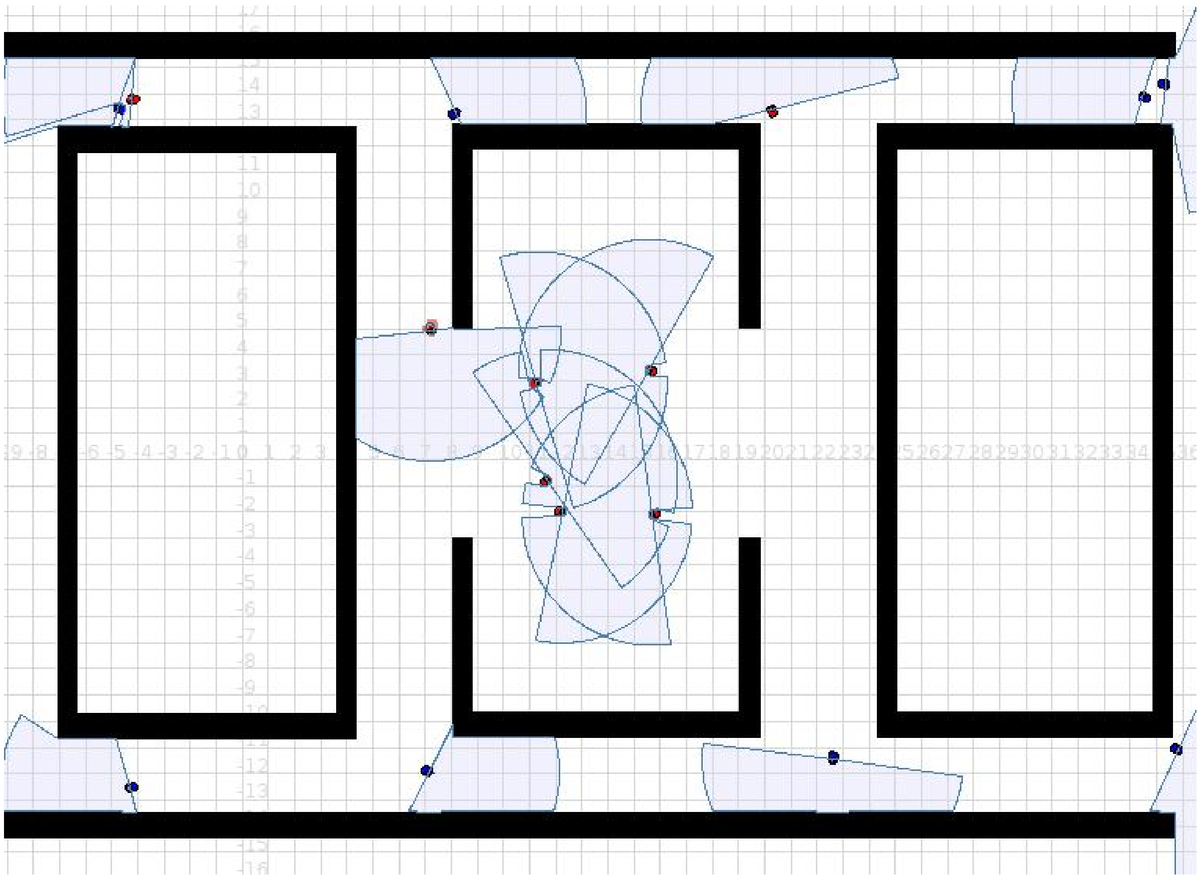
4. The primary robot's position is taken over of by the secondary robot and the secondary robot is free to go and recharge! Note: Incoming secondary robot calculates a new threshold value.



Debug Log when robots secondary robot replaces primary robot:

```
Robot9 has reached Observation Pt 8 ** Battery: 64.9979
Observation Pt 8 threshold: 35.0021 (new threshold!)
Robot9 replacing Robot8
Robot8 going from Observation Pt 8 to Docking Station 8. ** Battery:
21.121
```

5. Primary Robot reaches base station to recharge and takes the role of a secondary robot (starts looking for distress calls.) This cycle repeats indefinitely.



```
Robot8 has reached Docking Station 8 ** Battery: 2.82105
*** Robot8 battery RECHARGED! ***
Robot8 will rescue Observation Pt 7
Robot8 going from Docking Station 8 to Observation Pt 7. ** Battery: 100
```

6. Although I tried running simulations over extended periods of time certain bugs in my code, performance related issues with Player/Stage and the robots constantly dying due to bumping into each other I could not run it beyond a certain time-frame.

7. One one of the better runs that lasted over 15 minutes I saw 6 out of 8 points covered at all times and no point left uncovered for more than a minute. 3 Robots out of 16 died due to bumping into others.

8. I also tracked 20 robots come in and recharge with the following levels left at the time of docking in with an Average of 11.36 units.

*2.82105, 5.97106, 0.471052, 5.67106, 2.*84605, 3.52106, 13.6976, 32.8476, 13.347, 6.92106, 4.12105, 2.32105, 2.14605, 1.97105, 15.3715, 13.1961, 65.3479, 14.0461, 4.84605, 15.72

G. Future Work

Necessary Optimizations

1. Optimize number of secondary robots and docking points for a given set of observation points – Although I could have had 2 robots dedicated per observation point, I chose to go for a more risky algorithm where any robot reaches out to assist any other robot in trouble. This approach leaves room for me to reduce the number of secondary robots over time.

2. Develop social laws – From empirical studies, I could see that the robots often play chicken with each other and die as they are unable to unlock themselves from a head-on position with each other and reach the base station in time. Developing a social law to have them follow the right-side of the track may stop this from occurring.

3. Plan for surprises in the environment – Every robot estimates that the amount of time it requires to reach an observation point is what it would need to return back to the base station. However, this may not be true in situations where a different path exists for going and returning or when the usual path closes due to a surprise element. In this case, I hope to take into account multiple experiences from robots following the same path to come up with better threshold quantities.

4. Prioritize Observation Points – Out of all the possible observation points, some could have a higher priority over the others. It would be nice to have the robots know that in a worst-case scenario certain observation points can be abandoned to ensure 100% coverage of the more important ones.

5. For the sake of this simulation, my map has been fairly restrictive as it limits the movement of robots along the paths. I would like to change the map and see the performance of the algorithm while maintaining all the parameters of the current setup.

6. Optimize calculations for distress calls and threshold values – These values are currently calculated based on the battery levels of individual robots and not the overall status of the group which would be ideal.

7. Work around Player/Stage issues and guarantee 100% Surveillance over a 24 hour simulation. By reducing the number of robots on the map for better system performance, tweaking the algorithm and improving the map conditions, I think I should be able to achieve this target with some hard work.

Possible Enhancements

1. Reposition observation points over time – In a truly dynamic environment where the observation points change periodically, the idea of reporting back threshold values can be utilized by relative positioning of the new point vs. the older ones.

2. Make the base station(s) mobile – For surveillance activities (military applications, planetary explorations, etc.) the base station may be moving around as well. In this case, the robot should be able to calculate and extend it's threshold based on thresholds provided by the base station. Perhaps have the robots survey over an elliptical path and return to the moving base station. In a scenario where more than one base station exists, a robot should be able to reach the closest one.

H. References

1. Staying Alive Longer: Autonomous Robot Recharging Put to the Test
by Pawel Zebrowski and Richard Vaughan.
http://cres.usc.edu/pubdb_html/files_upload/374.pdf
http://www-robotics.usc.edu/~boyoon/project/docking/videos/docking.avi

2. EcoBot-II: An artificial agent with a natural metabolism
by Ioannis Ieropoulos, Chris Melhuish, John Greenman and Ian Horsfield
http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-2/295-300.pdf

3. NASA: Mars Exploration Rover Mission
http://marsrovers.jpl.nasa.gov/home/index.html

4. Potentially Distributable Energy: Towards Energy Autonomy in Large Population of Mobile Robots. In *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, June 20-23, 2007, Jacksonville, Florida, USA
https://ras.papercept.net/conferences/scripts/abstract.pl?ConfID=10&Number=56

5. Recharging robot teams: A tanker approach
by Pawel Zebrowski and Richard Vaughan. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Seattle, Washington, July 2005.
http://www.cs.sfu.ca/~vaughan/doc/zebrowski_icar05_submitted.pdf

6. Goodbye wires… MIT experimentally demonstrates wireless power transfer
http://www.physorg.com/news100445957.html
http://web.mit.edu/newsoffice/2006/wireless.html

7. The player/stage project: Tools for multi-robot and distributed sensor systems
by Brian Gerkey, Richard T. Vaughan, and Andrew Howard. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*, Coimbra, Portugal, June 2003.
http://www.cs.sfu.ca/~vaughan/doc/gerkey_icar03.pdf

8. Optimal Camera Placement for Automated Surveillance Tasks
by Robert Bodor, Andrew Drenner, Paul Schrater and Nikolaos Papanikolopoulos
Accepted but not yet published (used here with special permission from Andrew)

9. Allocating Tasks in Extreme Teams
by Paul Scerri, A Farinelli, Steven Okamoto, Milind Tambe. *4th Int'l Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, July 2005.
http://www.cs.cmu.edu/~pscerri/papers/LADCOP-AAMAS05.pdf