Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Rolling Dispersion for Robot Teams

Elizabeth A. Jensen and **Maria Gini**
Department of Computer Science and Engineering
University of Minnesota
{ejensen, gini}@cs.umn.edu

Abstract

Dispersing a team of robots into an unknown and dangerous environment, such as a collapsed building, can provide information about structural damage and locations of survivors and help rescuers plan their actions. We propose a rolling dispersion algorithm, which makes use of a small number of robots and achieves full exploration. The robots disperse as much as possible while maintaining communication, and then advance as a group, leaving behind beacons to mark explored areas and provide a path back to the entrance. The novelty of this algorithm comes from the manner in which the robots continue their exploration as a group after reaching the maximum dispersion possible while staying in contact with each other. We use simulation to show that the algorithm works in multiple environments and for varying numbers of robots.

1 Introduction

In the event of a fire or earthquake, it is not always possible for a rescue team to enter a building immediately, due to safety concerns for the human rescuers. However, a team of small robots could be deployed to explore the building, locate survivors, and mark pathways to the exits. This information can then be relayed back to the human search and rescue team, who can use it to prioritize tasks and plan their actions when it becomes safe for them to enter the building.

There are multiple methods for robots to explore an unknown environment. Gage [1992] proposed three categories of coverage–blanket, barrier and sweep coverage. Choset [2001] later presented an extensive overview of coverage path planning algorithms according to those categories. Most coverage algorithms are focused on surveillance and usually entail creating a sensor network to provide either blanket or barrier coverage of the environment. Attempting to provide blanket coverage can require a prohibitively large number of robots and some algorithms still don't achieve full coverage.

In our scenario, blanket coverage isn't necessary, nor is a persistent sensor network. Since the environment is unknown, the required number of robots for full coverage is also unknown, and, even if known, may well exceed the number of robots available on site. Thus, we have developed an algorithm for rolling dispersion, in which the team of robots completes a single sweep of the environment to locate points of interest that can be relayed to the search and rescue team. We want the robots to disperse, while maintaining communication, and then continue on through the environment as a group to ensure that each point in the environment is viewed at least once.

Our main contribution is a novel distributed algorithm, which guarantees that the entire environment is seen, that the robots maintain communication, and that they return to the entry point upon completion of the task. The key feature of the algorithm is that the robots advance as a group as they explore the environment, leaving behind beacons to mark both explored areas and the path to the exit. The robots use wireless signal intensity to ensure that they stay in communication with at least one other robot at all times, so no robot gets lost or is left behind. The algorithm is fully distributed, and each robot makes its own decisions on which behavior to execute depending on the situation and on the robot's current role, yet the robots operate as a team.

2 Related Work

In recent years, multi-robot systems have gained popularity due to decreases in the cost and size of the components made possible by hardware advances [Arai *et al.*, 2002]. There are several advantages to the use of a multi-robot system over the use of a single robot, including cost, efficiency and robustness. A single robot can be designed to efficiently complete its task, but it may then be suitable for only a small set of tasks, and added functionality increases the cost, size and energy requirements, while reducing maneuverability. In addition, if part of the robot fails, it may fail at the entire task. In contrast, a multi-robot system comprised of 10 or 100 smaller, individually less-capable robots, with several of each type needed to complete the different parts of the task, can still accomplish their goal even if some of them fail. The multi-robot system has an inherent redundancy that increases the system's robustness [Choset, 2001].

In a centralized multi-robot system, either a small set of more powerful robots or an external controller issues instructions to the others and keeps the group organized and coordinated. This requires less of the individual robots, but more of the controller, and the system is also then susceptible to

a complete failure if the central controller goes down, even if the robots are still running. In addition, a centralized approach does not scale as well, because one machine can efficiently control only a limited number of robots at a time. It also reduces the distance the robots can move from the central controller because they must maintain connectivity.

In small environments, however, this can be an effective approach. Stump et al. [2008], made a single robot a base station, while the other robots formed a communication bridge as they moved into the unknown region. Similarly, Rekleitis et al. [1997] used one robot as a stationary beacon for another robot, thus reducing odometry error in the robot that was moving through the environment. The centralized approach also has the advantage of providing a global map, used by the central controller to direct the movements of the robots [Burgard *et al.*, 2005; Stachniss and Burgard, 2003; Wurm *et al.*, 2008]. Some previous work assumes that, if the robots do split up, they will be able to make perfect maps of their explorations and it will be trivial to merge these when the team regroups [Hazon *et al.*, 2006; Latimer IV *et al.*, 2002], but this is actually quite difficult to achieve in practice. These approaches require constant monitoring of the individual robots in order to keep the map consistent and the exploration efficient. Further, the range is limited by communication restraints.

Though a centralized system has the advantage of global maps, and thus more knowledge to make coordination decisions, it is effective only in small environments, as it does not scale well. In contrast, distributed coverage algorithms are designed to scale well, and can also better take advantage of the robustness inherent in the multi-robot system, making them more reliable than the centralized systems, which have a single point of failure. Much of the work on distributed methods for coverage come not from the multi-robot field, but from sensor networks research. This research is applicable to multi-robot systems because of the similar nature of the problems, and the similarity of the constraints such as limited communication, sensors and power.

Ma and Yang [2007] show that the most efficient dispersion of mobile nodes is triangular, producing the maximal overall coverage and minimal overlap or gap in the coverage. The dispersion formation is achieved through the nodes' local communication, in which they determine distance and bearing to their neighbors, so that they can move towards the optimal formation. Liu et al. [2005] have shown that repeated location updates can lead to better coverage over time. Similar approaches by Howard et al. [2002] and Cortes et al. [2004] used potential fields and gradient descent, respectively, to disperse the nodes. In simulation, both methods effectively spread the nodes throughout the environment.

A recent trend has been to model distributed algorithms for multi-robot systems on insect behavior. The robots have very little ability individually, but can communicate with local neighbors and use simple distributed algorithms to arrange themselves according to a desired dispersion pattern. McLurkin and Smith [2004] have developed both a physical robot and several algorithms for dispersion and exploration in indoor environments. Their algorithms rely on the robots maintaining connectivity in order to perform correctly, passing information amongst themselves to spread out in particular patterns, such as uniform and cluster dispersions, as well as maintaining a frontier for exploration. The robots keep lists of their neighboring robots, and use gradients based on the network of robots to direct their movement. These algorithms are similar to those in Cortes et al., Howard et al., and Liu et al. [2004; 2002; 2005], but allow for greater variability in the dispersion pattern, including clusters and perimeter formations. The robots can also perform tasks such as frontier exploration and following-the-leader, which is not considered in sensor network research. Additional work based on insect behavior tends more towards pheromone-based algorithms [Batalin and Sukhatme, 2007; Koenig and Liu, 2001; Mamei and Zambonelli, 2007; O'Hara *et al.*, 2008]. However, these rely on items placed in the environment for communication and navigation. While this is simple to implement in simulation, it is much more difficult in a physical environment.

Distributed systems can be deployed over a large area, and are more resilient to failure than a single robot. Instead of a central controller issuing commands and collating data, each robot is responsible for its own movements and data collection, and relies only on local neighbors for coordinating exploration and dispersion. This spreads through the entire group, so that it may seem that the robots are working together on a global scale, but in actuality the decisions are made individually on a local scale. Information can be passed throughout the group, similar to the communication bridge in Stump et al. [2008], but it is more of a broadcast message than a directed message along a single path. On a city-wide scale, a search and rescue team does not need information about the entire city, but only the few blocks under its supervision. A distributed system would allow a local group to work independently, while sharing data with neighboring groups as necessary.

Dirafzoon et al. [2012] provide an overview of many sensor network coverage algorithms, both centralized and distributed, which can be applied to multi-robot systems as well. However, many of these rely on individual robots knowing the distance and bearing of other robots around them, which requires more sophisticated sensors and defeats some of the purpose of using a team of basic robots. For example, Kurazume and Hirose [2000] developed an algorithm in which the team of robots was split into two groups, one of which remained stationary while the other moved, and then they traded roles. This made for effective movement through an unknown environment, but the robots relied on sophisticated sensors to perform dead reckoning to determine the locations of the stationary robots. At the other end of the spectrum, there is research that has shown that a team of robots can disperse into an unknown environment using only wireless signal intensity to guide the dispersion [Ludwig and Gini, 2006]. This method allows the use of small, simple robots, without the need to carry a heavy payload of sensors, so that the robots can run longer and explore further. Smaller, simpler robots are less expensive, so more robots can be acquired for the same task. However, attempting to provide blanket coverage can require a prohibitively large number of robots and may still be unable to achieve full coverage.

3 Rolling Dispersion Algorithm

In our algorithm, we wish to achieve full coverage–every point in the environment has been viewed by a robot at least once, so that nothing is overlooked–but we wish to do it with a team of small, basic robots. This rules out blanket coverage, due to team size, and dead reckoning methods for determining robot locations, due to limited sensor and computational capability. We have also chosen a distributed method so that we can take advantage of the robust nature of a team of robots.

Keeping the above items in mind in designing the rolling dispersion algorithm, we established one major constraint on the scenario–that there are not enough robots to provide blanket coverage of the environment. We assume that the robots have some sort of proximity sensor to allow them to avoid collisions, a wireless card with a minimum range of 10 meters for communication and a means for carrying and dropping beacons (small devices such as a ZigBee mote or an RFID tag). We also assume a disaster scenario, so the specifics of the current environment are unknown, even if information for the pre-disaster environment (such as a map) is available.

During the exploration of the environment, each robot will execute the algorithm and make decisions as an individual, but it will have input from the beacons and other robots. Beacons are dropped by the robots for three reasons. The first is to mark a path that has been fully explored, thus preventing multiple explorations of the same area. The second is to mark the path to an unexplored area, which was temporarily abandoned to complete exploration in another area where more robots were needed. This path can later be followed back to the frontier to complete the exploration of that area. The third reason is to mark the path to the exit, so that the robots can exit the environment when the exploration is complete.

The robots use the wireless signal not only for communication, but also to direct their movement, both in dispersing to explore a larger area and to return to the starting point. Wireless signal intensity can fluctuate due to obstacles between robots, and may not be the same at every point a set distance from the origin, but this is not critical to the operation of our algorithm. The goal is to maintain communication, so the robots only need to know if the signal intensity is increasing or decreasing to inform their decision on which direction to move. This may not lead to the maximal dispersion, but suboptimal dispersion is acceptable since our main priority is to achieve full coverage without loss of communication. Because the robots are so spread out, each individual robot has few neighbors, keeping bandwidth requirements low even with larger teams.

While our algorithm uses wireless signal intensity to disperse the robots, and beacons to mark locations, the innovation in our approach lies in the manner in which the robots continue the exploration past the bounds of their initial dispersion. The robots are not allowed to move in isolation, but must always stay within communication range of the team. The highest priority of a robot that has lost communication with the team is to reestablish that communication. When there is an area to be explored that is beyond the reach of the robots nearby, because they would have to move out of communication range to reach it, then the entire team of robots will move towards the unexplored area. This approach has two main benefits. First, it means that the robots are less likely to get lost, since they will have a wireless signal to follow to get back to the entrance. Second, the robots will clear each room and corridor in a methodical manner, similar to the pattern used in law enforcement, thus reducing the likelihood of missing an area.

3.1 Algorithm Details

The robot team explores using the Rolling Dispersion Algorithm (Alg. 1). Each robot uses information about connectivity with its neighbors and nearby obstacles to choose which of the following behaviors it will execute on each iteration of the algorithm.

Avoid Collisions: Use the proximity sensors to avoid colliding with walls, objects, and other robots.

Disperse: Move towards open space, checking wireless signal intensity between myself and my sentry. If wireless signal intensity is not decreasing, change direction and continue moving forward. Move away from beacons marking explored areas.

Follow Path: Alert neighbors that I can fulfill the request. Concatenate my path with that of the requesting robot, and follow the path to the requesting robot.

Guard: Stay in place and act as a sentry for other robots.

Retract: Drop a beacon to mark the explored area and return to my sentry's location.

Seek Connection: First go in reverse to see if a connection can be reestablished. If that doesn't work, turn around and move forward, changing direction occasionally until a connection with another robot is made.

Figure 1 shows a finite state machine of the algorithm and when a robot decides which behavior to apply each iteration. The numbers correspond to lines in Algorithm 1, and the letters are the initials of the behaviors: AC = Avoid Collisions, D = Disperse, FP = Follow Path, G = Guard, R = Retract, and SC = Seek Connection.

At any given time in the exploration, each robot is either a sentry or an explorer, but robots can switch between roles as needed. The sentries provide the backbone of the communication network and do not move while in that role. Explorers

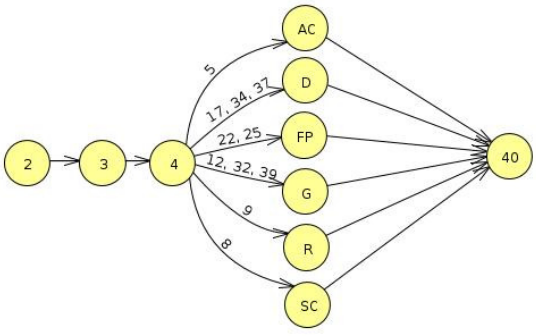


Figure 1: A finite state machine for Alg. 1. Numbers refer to lines in Alg. 1 and letters are behavior initials.

Algorithm 1 Rolling Dispersion Algorithm

```
loop
    Update connectivity graph using signal intensities
    Share new connectivity graph with neighbors
    Check for open paths, and update branch_count
    if I am too close to an obstacle then
        set behavior to Avoid Collisions
    else if I am disconnected from all neighbors then
        set behavior to Seek Connection
    else if I am in a dead end then
        drop a beacon set to explored
        set behavior to Retract
    else if my sentry's intensity is below threshold then
        change status to sentry
        set behavior to Guard
        if my only neighbor is my sentry then
            request additional explorers
    else if I am an explorer approaching a beacon then
        set behavior to Disperse
        if the beacon is marking an explored area then
            pivot before continuing on
    else if I have received a request then
        if I am an explorer then
            drop a beacon set to unexplored
            set behavior to Follow Path
        else if I am a sentry then
            if my only neighbor is my sentry then
                if my branch_count is lower then
                    drop a beacon set to explored
                    change status to explorer
                    set behavior to Follow Path
            else
                set behavior to Guard
                pass the request on to my neighbors
    else if I have reached the requesting robot then
        set behavior to Disperse
    else if if I am an explorer then
        set behavior to Disperse
    else
        set behavior to Guard
    Apply chosen behavior
```

move away from sentries, guided by the wireless signal intensity, with some directional input from their proximity sensors. An explorer becomes a sentry when it reaches the edge of its sentry's wireless range and there are no other sentries or explorers that it can use to stay connected to the group. A sentry can become an explorer under three conditions. One, if the path it was marking has been fully explored and all robots beyond it have returned past it, then it follows a path to an unexplored area and continues on as an explorer there. Two, if there are no explorers left, a sentry with no dependent sentries will become an explorer and explore a different path. Three, if there are no explorers, a sentry marking the path to the entry with only one dependent sentry will become an explorer. A list of data items that robots keep in memory and share with other robots and beacons can be found in Table 1.

As the robots move through the environment, they establish a path of sentries and beacons back to the entrance. This path can also be used to guide a robot to the edge of the explored area, which is the rolling aspect of the dispersion. Robots retract along their path when they have completed the exploration of that branch. During the retraction step, beacons are left at all intersections to mark the explored area to prevent multiple explorations of the same area. These beacons have a status of *explored* which they send to any approaching robot to let the robots know not to go that direction. When a robot has retracted to the beginning of its path, it then moves out along another path, which may already have some robots exploring it, or it may be a completely unexplored path. When all the robots have become sentries, but there are still areas to be explored, a sentry with no child sentries will drop a beacon with the status of *unexplored* and move to explore another path. When there are not enough robots to both maintain the path to the entrance and continue the exploration, the robot at the entrance will drop a beacon to mark the path, and then move down the path to the unexplored area. This beacon will have a status of *entry*, to differentiate it from the other beacons. A beacon's status can be changed from *unexplored* to *explored* when the robots retract past it after exploring the area. A list of beacon data, including status (*explored*, *unexplored*, *entry*), closest sentry, and path back to the entry is also shown in Table 1.

In summary, the algorithm works by first having the robots disperse. This movement is primarily directed by the wireless signal intensities between the robots, as in Ludwig and Gini's [2006] work, though the robots' direction is also influenced by the proximity sensors, to avoid collisions. Once the robots have reached the maximum dispersion coverage without losing communication with at least one other robot, the majority of the robots will stay in place, while a few leave their frontier and move along another path to complete the exploration of that path. When a path has been fully explored, those robots will retract until they reach a robot that marks an intersection with unexplored paths, dropping beacons as appropriate, and then continue along one of the unexplored paths to complete the exploration. When there are no more paths to explore (i.e. every point in the environment has been covered at least once), the robots will retract to the entry.

3.2 Algorithm Correctness

The primary goal of the algorithm is to achieve full coverage of the environment by having each point in the environment

Table 1: Robot and beacon data.

	Stored by		Shared with	
Data	**Robot**	**Beacon**	**Robot**	**Beacon**
ID	Yes	Yes	Yes	Yes
Path	Yes	Yes	Yes	No
branch_count	Yes	Yes	Yes	Yes
Sentry	Yes	Yes	Yes	Yes
Neighbors	Yes	No	Yes	No
Robot Status	Yes	No	No	No
Beacon Status	No	Yes	Yes	No

viewed at least once by a robot. To ensure that this occurs, the robots explore along a path until they are very close to the wall in front of them (to help reach the corners) before turning around. Additionally, the beacons are left at the explored side of an intersection to push robots away from that path, but leaving other paths off that intersection open for exploration. When the robots retract, they go back to the last intersection with open paths and explore those paths before retracting further. If a path was temporarily abandoned, the beacons will be set as unexplored markers, and any robots along the path will request explorers to complete the exploration. When the robots reach the end of the environment and begin retracting towards the exit, if they run across a beacon marking an unexplored path, they will then complete the exploration there.

Another concern in completing the exploration is preventing infinite loops, where the robots end up going around an obstacle or cycling through a set of rooms repeatedly. This is accomplished through the use of both beacons and sentries. If a robot is exploring an area and comes upon a robot that was not previously its neighbor, they exchange paths to the entry point, and if those do not match up, the robots have met on two different paths to that point. The robot on the longer path (or the explorer, if one is a sentry), will begin retracting, and drop a beacon to mark an explored area. If an explorer finds a beacon marking an unexplored area, on a path that wasn't intentionally leading to that beacon, it begins retracting and drops a beacon at the previous intersection to mark an explored area. Anything beyond the beacon marking the unexplored area will be explored when the robots return along the original path to that beacon. Thus far, these two actions have been sufficient to prevent infinite loops.

The last issue is to deal with the possibility of robot or beacon failures. Given the disaster scenario we are considering, where the environment is too dangerous for a human to enter, there is a high likelihood that a robot or beacon may be destroyed during the exploration. If the failure occurs at the edge of the explored region, then a failed beacon means re-exploring a small area. A failed sentry would be on the frontier only if it was waiting for more explorers, so the loss will have little impact. A failed explorer would simply need to be replaced. If a robot fails in the middle of a path, then a buddy system, where every sentry is actually two robots or a combination of a robot and a beacon, would create redundancy to cover a single failure. If a beacon fails in the middle of a path, then retracting robots will treat the empty area as an unexplored region and explore it as usual.

4 Experimental Results

We ran our experiments in the Player/Stage [Gerkey *et al.*, 2003] simulation environment. Each experiment used the Pioneer robot model for the mobile robots, and a modified Pioneer robot for the beacons. Each robot was equipped with 16 sonar sensors and a laser rangefinder for obstacle detection.

4.1 Experiment 1: Simple Corridor

The first set of experiments used the environment in Figure 2 to test the main properties of the algorithm. The environment has a simple topological structure, but it is large enough to require nine robots for blanket coverage. We did our experiments with two and four robots. It took an average of 275 and 192 seconds, respectively, for the robots to completely explore the environment using the rolling dispersion algorithm. The robots dropped six beacons in total, marking the entrances to each room.

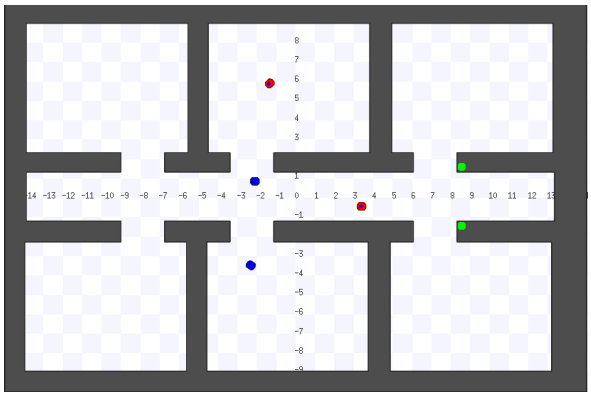
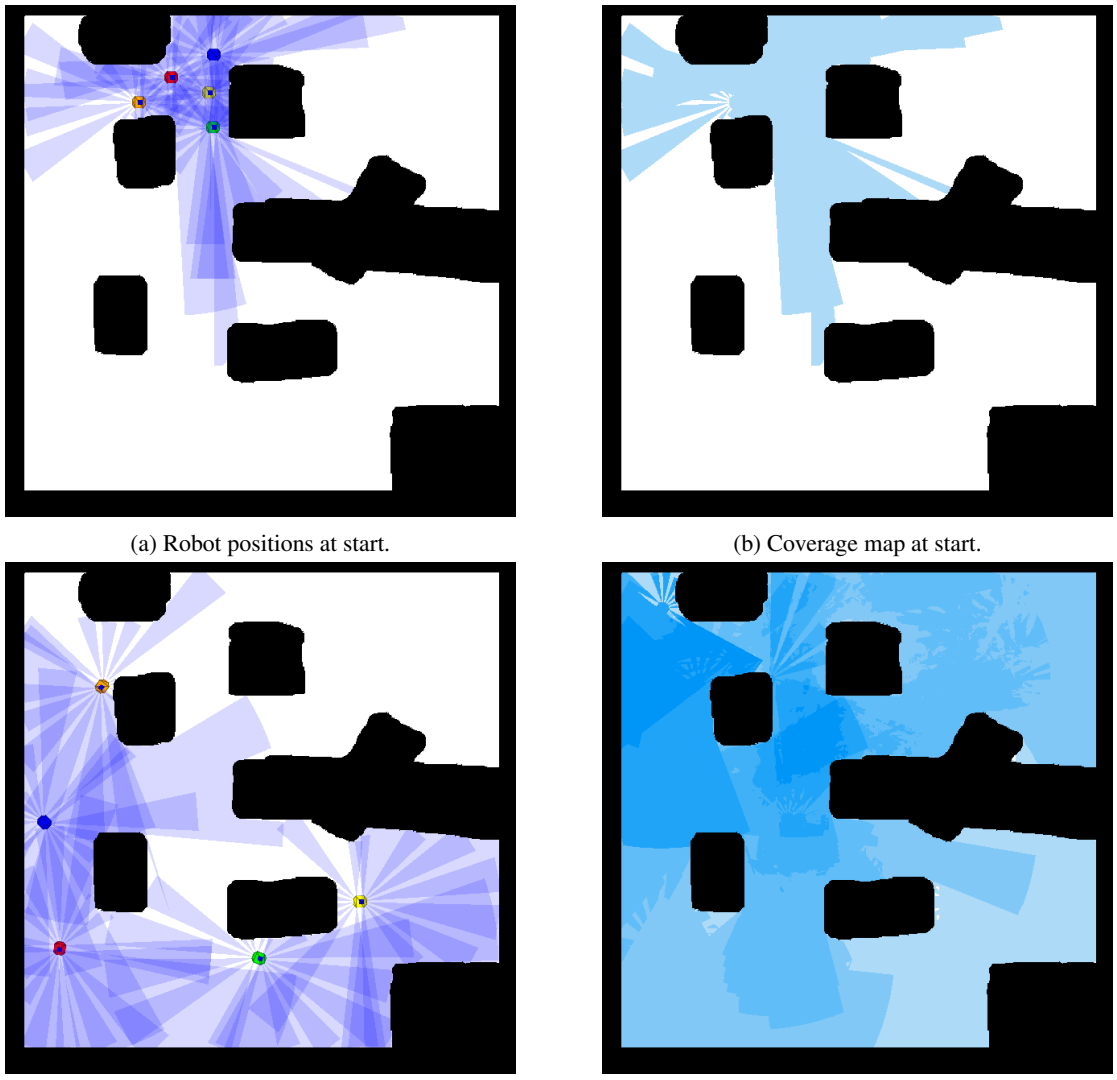


Figure 2: A simple environment with exploration partially completed. The robots (red and blue) are moving right to left, and have dropped two beacons (green) so far.

4.2 Experiment 2: Cave

In the second set of experiments, we used a cave-like environment, which is more complicated in part because of how open the area is, and how many cycles are possible in the exploration. Though the area to be covered is smaller than that of the simple corridor environment, the cave environment requires ten robots (one more than the simple environment) for blanket coverage due to the many corners and odd angles of the obstacles. We ran these experiments with five and eight robots, with ten runs each. Figure 3 shows the simulation view and coverage map for the start and end of a simulation run with five robots. In the simulation view, one can see the five robots as well as their sensors' field of view. The coverage maps show the area that had been viewed by the robots' sensors, and locations that had been viewed multiple times are shaded darker than locations that had been viewed only once.

With five robots, it took an average of 192 seconds, with a standard deviation of 31 seconds, to complete the exploration. With eight robots, it took an average of 167 seconds, with a standard deviation of 29, to complete the exploration. Figure 4 shows the percentage of the environment covered over time on average by 5 and 8 robots, with error bars to show the standard deviation at individual time points. The eight robots start more slowly, as it takes them more time to spread apart, but then they cover the environment more quickly than the five robots. The five robots show high variability in progress between 50 and 150 seconds, which is the result of waiting on retraction when too many robots went down a short path. This did not show up in the eight robot runs, mostly because there were enough robots that every path generally had enough robots to continue exploring while robots on another path were retracting.



(a) Robot positions at start.

(b) Coverage map at start.

(c) Robot positions at full coverage.

(d) Coverage map at full coverage.

Figure 3: Five robots in the cave environment.

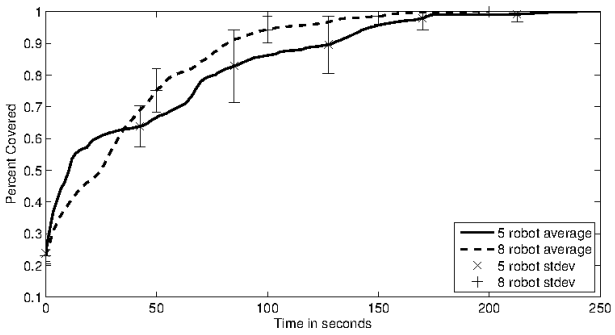


Figure 4: Average percent of the environment covered over time for five and eight robots (ten runs each). Error bars show standard deviation at that time point.

5 Conclusions

We have developed an algorithm to disperse a small team of robots into an unknown environment to completely explore the space while staying connected at all times during the exploration. The algorithm requires fewer robots than would be needed for blanket coverage, but still provides the necessary information about the environment. Our experiments demonstrate that the algorithm works in multiple environments of varying complexity.

In future work, we will test the algorithm in larger and more complex environments, and tune the algorithm to better handle a large number of robots in a small area. Additional extensions to the algorithm include using the beacons to guide mobile survivors to the exit, and working with human search and rescue members in performing the exploration. Formal proofs of correctness have also been left to future work.

References

[Arai *et al.*, 2002] T. Arai, E. Pagello, and L. E. Parker. Guest editorial advances in multirobot systems. *IEEE Robotics and Automation Magazine*, 18(5):655–661, October 2002.

[Batalin and Sukhatme, 2007] M. A. Batalin and G. S. Sukhatme. The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. *IEEE Trans. on Robotics*, 23(4):661–675, August 2007.

[Burgard *et al.*, 2005] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *IEEE Trans. on Robotics*, 21(3):376–386, June 2005.

[Choset, 2001] Howie Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[Cortes *et al.*, 2004] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. on Robotics and Automation*, 20(2):243–255, April 2004.

[Dirafzoon *et al.*, 2012] Alireza Dirafzoon, Saba Emrani, S. M. Amin Salehizadeh, and Mohammad Bagher Menhaj. Coverage control in unknown environments using neural networks. *Artificial Intelligence Review*, 38:237–255, October 2012.

[Gage, 1992] Douglas W. Gage. Command control for many-robot systems. In *19th Annual AUVS Technical Symposium*, pages 22–24, Huntsville, Alabama, June 1992.

[Gerkey *et al.*, 2003] B. Gerkey, R.T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proc. 11th Int'l Conf. on Advanced Robotics*, volume 1, pages 317–323, 2003.

[Hazon *et al.*, 2006] N. Hazon, F. Mieli, and G. A. Kaminka. Towards robust on-line multi-robot coverage. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1710–1715, May 2006.

[Howard *et al.*, 2002] Andrew Howard, Maja J. Matari, and Gaurav S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proc. Int'l Symp. on Distributed Autonomous Robotic Systems*, pages 299–308, 2002.

[Koenig and Liu, 2001] Sven Koenig and Yaxin Liu. Terrain coverage with ant robots: a simulation study. In *Proc. Int'l Conf. on Autonomous Agents*, pages 600–607. ACM, 2001.

[Kurazume and Hirose, 2000] Ryo Kurazume and Shigeo Hirose. An experimental study of a cooperative positioning system. *Autonomous Robots*, 8:43–52, 2000.

[Latimer IV *et al.*, 2002] D. Latimer IV, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 1, pages 961–967, 2002.

[Liu *et al.*, 2005] Benyuan Liu, Peter Brass, Olivier Dousse, Philippe Nain, and Don Towsley. Mobility improves coverage of sensor networks. In *Proc. Int'l Symposium on Mobile Ad Hoc Networking and Computing*, pages 300–308. ACM, 2005.

[Ludwig and Gini, 2006] Luke Ludwig and Maria Gini. Robotic swarm dispersion using wireless intensity signals. In Maria Gini and Richard Voyles, editors, *Proc. Int'l Symp. on Distributed Autonomous Robotic Systems*, volume 7, pages 135–144. Springer Japan, 2006.

[Ma and Yang, 2007] Ming Ma and Yuanyuan Yang. Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Trans. on Computers*, 56(7):946–847, July 2007.

[Mamei and Zambonelli, 2007] Marco Mamei and Franco Zambonelli. Pervasive pheromone-based interaction with rfid tags. *ACM Trans. on Autonomous and Adaptive Systems*, 2(2):4, 2007.

[McLurkin and Smith, 2004] J. McLurkin and J. Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In Maria Gini and Richard Voyles, editors, *Proc. Int'l Symp. on Distributed Autonomous Robotic Systems*, volume 7. Springer Japan, 2004.

[O'Hara *et al.*, 2008] K. J. O'Hara, D. B. Walker, and T. R. Balch. Physical path planning using a pervasive embedded network. *IEEE Trans. on Robotics*, 24(3):741–746, June 2008.

[Rekleitis *et al.*, 1997] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, volume 2, pages 1340–1345. Morgan Kaufmann Publishers, Inc., August 1997.

[Stachniss and Burgard, 2003] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2003.

[Stump *et al.*, 2008] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1525–1530, May 2008.

[Wurm *et al.*, 2008] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 1160–1165, September 2008.