Small Team Exploration with Communication Restrictions

Elizabeth A. Jensen, Ernesto Nunes, and Maria Gini

Abstract

Exploring an unknown environment comes with many risks and complications, and using a team of robots is a practical means to reduce the risk to humans, while still accomplishing tasks such as creating maps or locating points of interest. In space exploration, these teams of robots may be used to scout and prepare sites for later human outposts. In search and rescue, these robots can be used to locate unstable areas or survivors in the aftermath of an earthquake or fire. One feature of both scenarios is that communication is often limited, either by the disaster, or by distance, so the robots need to be able to operate autonomously. In this paper, we present an algorithm for a small team of robots to explore an unknown environment even if communication restrictions. We provide proofs of correctness and guarantee full coverage of the environment, even in the event of attrition.

Introduction

If we are to expand into space, and set up outposts on the Moon or Mars (NASA 2004), there is much work to be done in preparing the area and constructing preliminary buildings and utilities. One aspect that must not be overlooked is exploring, in detail, the locations and determining if there are complications or dangers to construction that cannot be detected from Earth. Sending a team of robots to scout the planned site of an outpost can give us this information and allow us to make appropriate decisions for building and facilities placements. However, there is a limit on communication with such a team of robots, so the human operators cannot be expected to keep a constant watch over the robots and direct their every move. Instead, as Leitner (2009) argues, the robots used in space exploration must have at least some degree of autonomy, and must be able to coordinate amongst themselves to accomplish their tasks. There has been much research into exploration using robots to map out points of interest, particularly in search and rescue scenarios, where the robots can be used to locate survivors or weak structural supports in areas too dangerous for human search and rescue workers to enter.

While this is necessary, many of these previous avenues of research have considered using only one robot for the exploration, or a pairing of a ground robot and an aerial robot to reach more locations. Instead, we are interested in algorithms which allow teams of robots to enter and explore unknown and dangerous environments, leveraging greater numbers of robots to more quickly cover the environment, but also providing guarantees of full coverage even in the face of individual robot or beacon failures. In addition, a feature common to both disaster scenarios and space exploration, is that communication is more limited than we are accustomed to in everyday life, so these robots cannot rely on having constant contact with operators during the exploration. We therefore focused on developing algorithms which can function under various communication restrictions, even considering chemical or line-of-sight means of communicating between robots.

Our primary contribution in this work are two distributed algorithms for exploration using small teams of robots. The innovation in these algorithms comes from how the robots disperse into and subsequently explore the environment, even with communication restrictions. We provide proofs that the algorithms will achieve full coverage of the environment, return all functioning robots to the entry point, and that points of interest are marked in such a way that the human rescuers can go directly to those points when the environment is deemed safe for them to enter.

Related Work

In recent years, multi-robot systems have gained popularity (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. In addition, if even a small part fails, the robot may be unable to complete the task. In contrast, a multi-robot system comprised of 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. A multi-robot system has an inherent redundancy that increases the system's robustness (Choset 2001).

There are multiple methods for a team of robots to explore an unknown environment. Gage (1992) proposed three types of coverage. In blanket coverage, the robots cover the entire environment simultaneously. In barrier coverage, the robots

set up a perimeter around an area such that nothing can pass into or out of that area without being seen by at least one agent. In sweep coverage, the robots make a pass over the environment and ensure every point has been seen by at least one robot, but don't stay in any one location, instead moving progressively through the environment. Choset (2001) presented an extensive overview of coverage path planning algorithms according to those categories. Most coverage algorithms are focused on surveillance, and thus aim to achieve either blanket or barrier coverage. However, both of these types of coverage require enough robots to provide the full coverage, and that number can be prohibitively large.

In contrast, sweep coverage can be done with a small team, down to a single robot, if necessary, when all other robots on the team have failed (Fazli *et al.* 2010). Since the environment is unknown, the required number of robots for blanket coverage is also unknown, and, even if known, may well exceed the number of robots available on site. Thus, in our approach, we use an exploration algorithm 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.

In addition to the type of coverage provided, one must consider whether a centralized or distributed algorithm would provide more benefits to the overall exploration. In a centralized multi-robot system, an external controller issues instructions to the others and keeps the group coordinated. While this requires less of individual robots, if the controller fails, then the entire system will fail, even if the robots are still functional. In addition, a centralized approach does not scale well, because one machine can control only a limited number of robots at once. In small environments, however, a centralized approach can be effective. 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. The centralized approach also has the advantage of making it easy to create a global map, which can be used to direct other agents, human or robot (Burgard *et al.* 2005; Stachniss and Burgard 2003; Wurm *et al.* 2008). These approaches require constant monitoring of the robots in order to keep the map consistent and the exploration efficient.

A distributed approach, on the other hand, is inherently more scalable and can also take better advantage of the robustness of having multiple robots. Each robot is responsible for its own movements and data collection, and relies on only local neighbors for coordinating exploration and dispersion. 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 (Stump *et al.* 2008), but using broadcast messages rather than point-to-point messages. A distributed system thus allows an individual to work independently, while also sharing data with neighbors as necessary.

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, these methods successfully spread the nodes throughout the environment to achieve blanket coverage, but a sufficient number of robots may not be available outside of simulation.

Another recent trend has been to model distributed multi-robot algorithms on insect behavior. The robots have very little individual ability, but can communicate with local neighbors and arrange themselves according to a desired dispersion pattern. McLurkin and Smith (2004) have developed robots 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. These algorithms are similar to those in (Cortes *et al.* 2004; Howard *et al.* 2002; Liu *et al.* 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. Additional work based on insect behavior includes pheromone-based algorithms (Batalin and Sukhatme 2007; Koenig and Liu 2001; Mamei and Zambonelli 2007; O'Hara *et al.* 2008), which rely on items placed in the environment for communication and navigation.

Dirafzoon et al. (2012) provide an overview of many sensor network coverage algorithms 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. 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. On the other hand, research 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; Jensen and Gini 2013). This method allows the use of 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 also less expensive, so more robots can be acquired for a task.

Communication-Restricted Exploration

Our primary objective is for our algorithms to achieve full exploration of an unknown environment using a team of robots. Our algorithms can function with a single robot, if needed (due to availability or attrition), and with robots

that have limited individual capability. This means we cannot provide blanket or barrier coverage, but sweep coverage is all that is necessary to locate and relay back information about points of interest. Using a distributed approach takes advantage of the robustness inherent in having multiple robots, and is not impeded by the communication restrictions, especially since only local communication and information is needed for the robots to complete the exploration. Lastly, based on some of the insect-based algorithms, the robots carry and drop off beacons (such as ZigBee motes or RFID tags) to provide longer lasting trails and information to mobile agents, human or robot, that may pass by later.

We assume that the robots used have proximity sensors to avoid collisions, some capability for communication (wi-fi, line-of-sight, chemical, etc), and the means to carry and drop off beacons. We also assume that the specifics of the environment are currently unknown, even if pre-disaster information, such as a map, is available.

Our algorithms use the communication signal intensity to direct the robots' movements, keeping them linked as a group during the entire exploration. This provides the benefits of reducing the likelihood of robots getting lost and the possibility that part of the environment will be missed. Our innovation lies in making the algorithms independent of the type of communication used, and still making the robot team capable of achieving full coverage in a manner that is robust and complete even with some attrition.

Algorithm Details

The first algorithm, the Rolling Dispersion Algorithm (RDA), is presented in Algorithm 1. In this algorithm, the robots are either *explorers*, which move into the frontier, or *sentries*, which maintain a path to the entrance. Beacons are used to block off explored areas, or mark the path to a frontier that had to be temporarily abandoned in favor of fully exploring a different frontier. 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. Move away from beacons marking explored areas.

Follow Path: Alert neighbors that I can fulfill a request. 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: Re-establish communication with the rest of the group.

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.

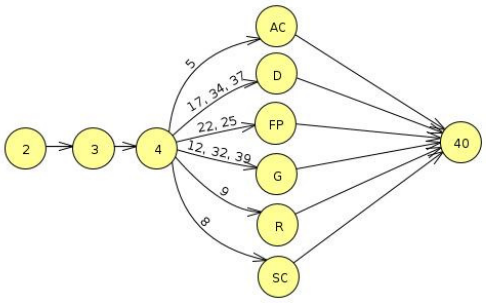
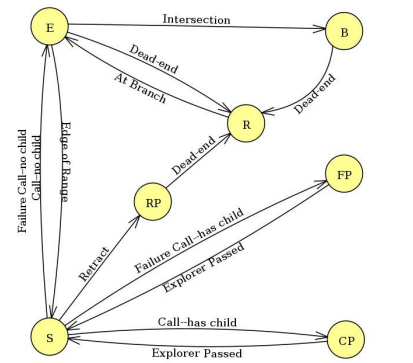


Figure 1: A finite state machine for Alg. 1. Numbers refer to lines in Alg. 1 and letters are behavior initials.

Algorithm 1 Rolling Dispersion Algorithm

```
 1: loop
 2:     Update connectivity graph using signal intensities
 3:     Share new connectivity graph with neighbors
 4:     Check for open paths, and update branch_count
 5:     if I am too close to an obstacle then
 6:         set behavior to Avoid Collisions
 7:     else if I am disconnected from all neighbors then
 8:         set behavior to Seek Connection
 9:     else if I am in a dead end then
10:         drop a beacon set to explored
11:         set behavior to Retract
12:     else if my sentry's intensity is below threshold then
13:         change status to sentry
14:         set behavior to Guard
15:         if my only neighbor is my sentry then
16:             request additional explorers
17:     else if I am an explorer approaching a beacon then
18:         set behavior to Disperse
19:         if the beacon is marking an explored area then
20:             pivot before continuing on
21:     else if I have received a request then
22:         if I am an explorer then
23:             drop a beacon set to unexplored
24:             set behavior to Follow Path
25:         else if I am a sentry then
26:             if my only neighbor is my sentry then
27:                 if my branch_count is lower then
28:                     drop a beacon set to explored
29:                     change status to explorer
30:                     set behavior to Follow Path
31:             else
32:                 set behavior to Guard
33:                 pass the request on to my neighbors
34:     else if I have reached the requesting robot then
35:         set behavior to Disperse
36:     else if if I am an explorer then
37:         set behavior to Disperse
38:     else
39:         set behavior to Guard
40:     Apply chosen behavior
```

Table 1: Sweep Exploration Algorithm Transitions and Finite State Machine

Current State	Input	Next State	Side Effect	Finite State Machine
Explorer	Intersection	Branch	Send call	
	Edge of Range	Sentry	Send call	
	Dead-end	Retractor	Drop explored beacon	
Sentry	Retract	Retract Path	Send retract	
	Call–no child	Explorer	Drop unexplored beacon	
	Call–has child	Call Path	Send call	
	Failure call–no child	Explorer	Drop unexplored beacon	
	Failure call–has child	Failure Path	Send failure call	
Branch	Dead-end	Retractor		
Call Path	Explorer passed by	Sentry		
Failure Path	Explorer passed by	Sentry		
Retract Path	Dead-end	Retractor	Drop explored beacon	
Retractor	At branch	Explorer		

The robots initially disperse to the furthest extent of their communication range. When the robots can no longer move apart without losing communication with another robot, they call for reinforcements, which leap-frog their way to the frontier, leaving behind beacons to mark the path to the entrance and any unexplored regions as necessary. When robots encounter a dead-end, which can be any location in which any direction the robot might move is blocked either by an obstacle (such as a wall) or another agent (robot or beacon), they drop off a beacon to mark the area as explored, and retract to the previous intersection before moving to a new frontier. This retraction process is repeated for every robot along that path until the last robot along that path turns onto a new path at the intersection, leaving the entire path marked as explored by the sequence of beacons left behind. The direction of the dispersion and exploration is primarily informed by the the wireless signal intensities between agents, as in Ludwig and Gini's (2006) work, though the individual robots also make decisions based on their proximity sensors to avoid collisions. When there are no more paths left to explore, the robots will retract back to the entry and we can then guarantee that all parts of the environment have been covered at least once.

The second algorithm, the Sweep Exploration Algorithm (SEA), is based on RDA, but is intended for use in scenarios with much more restrictive communication, such as chemical signals, or line-of-sight using a camera and color LEDs, or even in cases where there is too much loss in the signal to reliably send long messages. With limited means of communication, it is critical to reduce the number and size of messages to ensure full exploration. However, this also means that only one robot can be moving at a time, or the messages get mixed up and parts of the environment may not be explored. Therefore, instead of the robots initially dispersing in any direction as in RDA, they travel one at a time down a single path, until it is completely explored, and then retract and explore a new path. We show the finite state machine for robots using SEA and give the transitions in Table 1. In the finite state machine, the nodes are labeled with the initials of the states, which are listed in the transition table. We provide the full transition table because some of the transitions have side effects beyond the robot changing states, such as dropping a beacon or changing a beacon's state. SEA uses the same general method of exploring the environment as RDA, and the changes are mainly in how and what the agents communicate, though there is also the limitation on the number of robots moving at any given time.

Algorithm Correctness

We present here formal proofs that the robots running our algorithms will, even with communication restrictions, complete the exploration without missing any point in the environment, will not end up in infinite loops (so that the robots exit when done), and can succeed in these goals even with robot and beacon failures. We will later show results of running the algorithms in simulation.

Lemma 1. *The algorithms avoid unnecessarily repeated exploration.*

Proof. We will do this proof in two parts: first assuming that the beacons do not fail and then assuming that they may fail. In either case we will prove by contradiction that the algorithms will avoid unnecessary repeated exploration.

First, assume that the robots explore an area that has been previously explored. However, as previously explained, in our algorithms, if an exploring robot reaches a dead-end it drops a beacon to mark the explored area. Any robot that subsequently approaches that area would detect the beacon, receive the message that the area had been explored, and turn away to explore a new area. Thus, any re-explorations are prevented by the presence of these beacons.

Definition 1. *An exploring robot is in a dead-end when every direction it might move in is towards an obstacle, be it a wall, a beacon marking an explored area, or towards another robot, either the previous robot along the exploring robot's path, or a robot on a different path (in the case of a loop).*

Second, in the case when a beacon marking an explored area fails, the area around that beacon is unmarked. If the beacon is down a path with other beacons marking the path as explored, then the area will not be re-explored, because

the other beacons act as a buffer. If, however, the beacon was on the edge of either an unexplored area or the path to the entrance, the robots will have to re-explore the now unmarked area until reaching a dead-end, at which point they will drop new beacons to once again mark it as explored, preventing future unnecessary visits. The important caveat here is that we need to assume a finite number of beacon failures, otherwise robots would re-mark areas infinitely, which would lead to other areas not being explored or the robots not returning to the entrance to report the completed exploration. Hence, given a finite number of beacon failures, which is a reasonable assumption, we again derive a contradiction. □

Lemma 2. *The algorithms avoid infinite loops.*

Proof. We will again consider two cases in this proof: (1) assume that the beacons do not fail; and (2) assume that they may fail. In both cases, we will prove by contradiction that the algorithms will not get stuck in infinite loops.

First, assume that a robot is repeatedly exploring a loop in the environment. This is immediately a contradiction of Lemma 1 because the point at which the exploring robot first closed the loop (by reaching a previously explored location), it would have detected it was in a dead-end, and dropped a beacon to mark the area as explored. That beacon will break the loop, since the robots will treat it as an impassable obstacle no matter the direction from which they approach.

Second, in the case where beacons fail, the area surrounding the beacon's location becomes unmarked. If there are still beacons on either side marking the area as explored, then there will be no effect on the robots' exploration. If the beacon is neighbors with a robot or beacons on the path to the entrance, then the area will be re-explored, as in Lemma 1, but will again be stopped when a dead-end is once again located. We must assume that there will be a finite number of beacon failures, which is a reasonable assumption, so we again derive a contradiction. □

Lemma 3. *The algorithms achieve full coverage with a single robot.*

Proof. Assume we start with only one robot and an infinite number of beacons. In both algorithms, the robot will advance, leaving beacons to mark the return path and areas that have been explored. The explore/retract behaviors are the same as Depth-First search, which is complete in a finite search space when repeated states and loops are avoided. By Lemmas 1 and 2, we have proven that our algorithms avoid repeated states and loops. Our environment is finite. Thus, our algorithms will achieve full coverage with only one robot. □

Theorem 1. *The algorithms will achieve full coverage of the environment with multiple robots, and all functional robots will return to the entrance.*

Proof. We prove by induction that the algorithms function correctly when multiple robots are used.

Base Case: The base case is that the algorithm achieves full exploration when only one robot explores the environment. The proof for this case is given in Lemma 3.

Induction Step: Assuming that the algorithms achieve full exploration with k robots and the remaining functional robots return to base, we want to prove that the algorithms achieve full exploration when 1 more robot is added. Suppose that there are $k+1$ robots, then we need to show that: (1) the robots will not continuously explore overlapping areas, and (2) that the robots will not miss an area because they lost contact with the other agents, and (3) that no robot will be stranded in the environment.

First, in Lemma 1, we have already shown that there will not be unnecessary repeated exploration of an area, so long as the beacons remain active. Every robot that approaches the area will detect the beacons and move to a different area, and this remains true no matter how many robots are added.

Second, both algorithms enforce the restriction that the robots remain in contact with at least one other robot (or beacon, when there is only one robot) at all times, and when that connection is lost, the robots will immediately stop exploring and retreat in order to reconnect to the rest of the group. This is required to ensure complete coverage, because connectivity means robots will not miss an area, and will not re-explore areas that have been previously covered.

Third, the connectivity keeps the robots from being stranded in the environment, because only the exploring robot in a dead-end will mark an area as explored. The retraction step then brings the robot back into the group before the next robot marks an area as explored, so that no robot is left behind or trapped between two beacons blocking the path. In addition, the fact that the robots explore a path and then retract to the previous intersection, and then repeat the process, similar to Depth-First search with backtracking, means that all functional robots will eventually retract back to the entrance when the exploration is complete. Once again, adding another robot to the team does not change the functionality of the algorithms.

Thus, with $k+1$ robots, the algorithms still achieve full exploration, and the remaining functional robots return to the entrance when the exploration is complete. □

Algorithm Properties

In addition to the previous proofs, there are several important properties of the algorithms. Using multiple robots reduces the individual load on each robot, but the coordination adds costs in location visits and number of messages.

We can represent the environment as a graph, in which nodes represent locations that are separated by the distance of the communication range, in order to keep it to a finite number of nodes. With a single robot, each node is visited at most twice (leaf nodes are visited only once), assuming there are no failures. With multiple robots, each intermediate node n is visited at most $2n'$ times, where n' is the number of nodes beyond node n on that path. In the case where the total number of robots is less than n', the number of visits to each intermediate node is $2r$, where r is the number of robots.

In RDA, the robots must send many messages to confirm that they are still within communication range of each other, since multiple robots can move at the same time. This is quite costly in terms of bandwidth, processing, and power

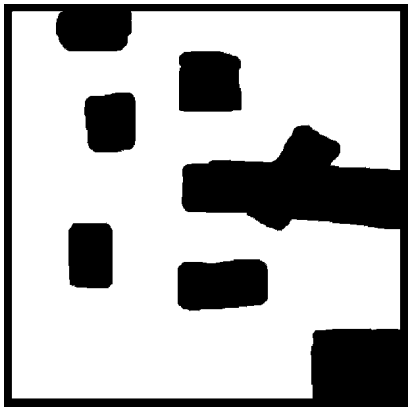


Figure 2: Cave-like environment used for experiments.

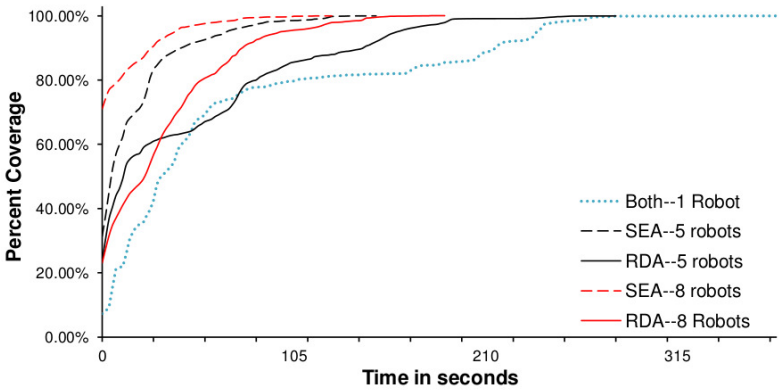


Figure 3: Average time to full exploration using 1, 5 and 8 robots with each algorithm.

consumption, and not practical with some kinds of communication. If we use chemical signals, flooding the environment with those chemicals will cause us to lose new messages in the old ones. But in restricting the communication in SEA, we lose the ability for multiple robots to move at the same time. However, it does reduce the number of messages being sent, which in turn makes the coordination easier as well. In SEA, we require only seven message types to complete the exploration, which correspond directly to the seven states shown in the transition table.

Consider starting with only two messages/states. The explorer and retract path are the most straightforward, and both are needed in order to create the backbone of paths to the frontier and back to the entrance. But there is no way to request additional robots to the frontier when the last explorer reaches the edge of the communication range with its nearest neighbor. So we need a third state, the call path, which both makes the request and leads the new explorer to the frontier. But then the explorer hits a dead-end, and without the repel state, the robots might explore the same area over and over, or go into an infinite loop. This brings us to four states. Without the retractor state, the agents on the retract path wouldn't know when to either change to the repel state (for beacons) or retract themselves (for robots). Without the branch state, each retractor robot would go all the way back to the beginning, and a branch might be cut off from further exploration because the retraction protocol requires that the robots leave repel beacons along the way. Lastly, since we are working in an unknown and potentially dangerous environment, it is essential that we have a means of identifying and replacing failed agents along the frontier and path back to the entrance. Thus, we require seven messages to complete the exploration.

Simulations and Results

We have conducted experiments in Player/Stage and ROS/Stage, using the same robot models and movement/sensor attributes, in order to test the viability of our algorithms. The testing environment, shown in Figure 2, is very open, with several large obstacles at varying intervals and in non-uniform shapes that leave wide open areas and potential loops. In all experiments, the robots start in a cluster between several obstacles near the upper left corner.

Figure 3 shows the rate of coverage for the RDA and SEA algorithms. With a single robot, both algorithms perform in the same manner, so only one line is plotted to show the rate of coverage with a single robot, and our algorithms perform only 10 seconds slower, at 337 seconds, than the shortest path to achieve full coverage at the same speed. Upon completing the coverage, it then takes an average of 45 seconds for the robot to return to the entrance, which varies by the path taken to complete the coverage.

With multiple robots, both algorithms improve upon this time. While in most cases the simulations show fairly steady increase in the percentage of coverage, the RDA with 5 robots does show some plateaus, due to the fact that the robots initially disperse in all directions, and then must wait for others to leap-frog out to the frontier before additional progress can be made. The SEA algorithm, on the other hand, does not show these plateaus as much because the robots start by exploring along a single path at a time. The experiments using SEA start with a higher initial coverage, due to the fact that the algorithms use different initial dispersion methods, and using the exact same cluster at the start led to many robot collisions. Accounting for that difference, and comparing times from the same starting percentage to full coverage, SEA is 1.35 times faster at achieving full coverage than RDA.

Upon further examination, it became clear that SEA will outperform RDA in environments where there are long paths, because SEA fully explores only one path at a time, while RDA will attempt to explore as many paths at a time as possible, but will run into issues when robots have to be pulled from other paths to fully explore long paths. In that case, RDA shows the plateaus in coverage rate because the robots are having to traverse longer paths to complete one path before going back to complete the exploration of another. In an environment such as an office building, which

is primarily made up of corridors with single room offices, RDA would have the advantage, because many rooms would then be explored at simultaneously, without long paths requiring lots of sentries. SEA does not do as well in this case, because it still allows only one robot to move at a time, so adding more robots does not provide the same benefit in time to completion as is seen in RDA.

Conclusion

We have presented here two distributed algorithms for multi-robot exploration of unknown environments. Both algorithms make use of their communication signal intensity to direct the movement of the robots, and beacons are used to mark explored areas in the environment in addition to creating a trail to the entrance and other points of interest within the environment. We have provided formal proofs that each of the algorithms will allow a team of robots to fully explore the environment, so long as at least one member of the robot team is still functional at the end of the exploration. Since space exploration is, by nature, exploration of unknown environments, we feel that our algorithms can provide promising means of utilizing small teams of rovers. In addition, the ability of the algorithms to function even with communication restrictions, while also keeping communication a priority, meshes well with intermittent and limited communication between such robot teams and human operators on Earth.

In future work, we plan to test the algorithms in other types of environments that include multiple loops, very large open areas, and many short paths. We will also run experiments with physical robots, and bring human interaction into both the exploration stage to make changes to the robots' planned course of exploration, such as to prevent a robot from trying to traverse a hole.

References

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

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. *Robotics, IEEE Trans. on*, 23(4):661–675, Aug. 2007.

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

H. Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and A.I.*, 31:113–126, 2001.

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

A. Dirafzoon, S. Emrani, S. M. Amin Salehizadeh, and M. B. Menhaj. Coverage control in unknown environments using neural networks. *AI Review*, pages 237–255, 2012.

P. Fazli, A. Davoodi, P. Pasquier, and A. K. Mackworth. Complete and robust cooperative robot area coverage with limited range. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 5577–5582, 2010.

D. W. Gage. Command control for many-robot systems. In *19th AUVS Technical Symposium*, pages 22–24, 1992.

A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proc. Int'l Sym. on Distributed Autonomous Robotic Systems*, pages 299–308, 2002.

E. A. Jensen and M. Gini. Rolling dispersion for robot teams. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 2473–2479, 2013.

S. Koenig and Y. Liu. Terrain coverage with ant robots: a simulation study. In *Proc. Fifth Int'l Conf. on Autonomous Agents*, pages 600–607, 2001.

R. Kurazume and S. Hirose. An experimental study of a cooperative positioning system. *Autonomous Robots*, pages 43–52, 2000.

Jürgen Leitner. Multi-robot formations for area coverage in space applications. 2009.

B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley. Mobility improves coverage of sensor networks. In *MobiHoc '05*, pages 300–308, 2005.

L. Ludwig and M. Gini. Robotic swarm dispersion using wireless intensity signals. In *Proc. Int'l Sym. on Distributed Autonomous Robotic Systems*, pages 135–144, 2006.

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

M. Mamei and F. Zambonelli. Pervasive pheromone-based interaction with rfid tags. *ACM Trans. Autonomous Adaptive Systems*, 2(2):4, 2007.

J. McLurkin and J. Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Proc. Int'l Sym. on Distributed Autonomous Robotic Systems (DARS)*, 2004.

NASA. The vision for space exploration. Technical Report Technical Report: NP-2004-01-334-HQ, NASA HQ, 2004.

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

I. Rekleitis, G. Dudek, and E. 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, August 1997.

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

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.

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, Sept. 2008.