Communication-Restricted Exploration for Search Teams

Elizabeth A. Jensen, London Lowmanstone and Maria Gini

Abstract Exploring an unknown environment comes with risks and complications, and in some cases an environment may be too dangerous for humans to explore, but immediate exploration is critical, as in the aftermath of an earthquake. Robots, however, can be deployed to seek out points of interest and report back to the waiting human operators. One aspect of a disaster scenario is that communication is often more limited than we are accustomed to in everyday life, so these robots cannot rely on having constant contact with the outside world, or even with all other robots in the environment. In this paper, we present two algorithms for a small team of robots to explore an unknown environment, and use both simulation and experiments with physical robots to demonstrate the algorithms' performance. We provide proofs of correctness and guarantee full coverage of the environment, even with attrition.

1 Introduction

When a search and rescue team arrives on the scene of a disaster, it may be prevented from entering immediately due to the instability of the environment. Earthquakes may have aftershocks well after the main quake ends, and fires may have burned through the support structure of a building, making it too dangerous for humans to enter. Many researchers have developed robots or the means to use them in an initial exploration, to map points of interest such as the location of survivors or weak supports, and send this information to the human rescuers waiting outside [2, 24].

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

E. A. Jensen (✉) · L. Lowmanstone · M. Gini
University of Minnesota, Minneanapolis, MN, USA
e-mail: ejensen@cs.umn.edu

L. Lowmanstone
e-mail: london4@comcast.net

M. Gini
e-mail: gini@cs.umn.edu

R. Groß et al. (eds.), *Distributed Autonomous Robotic Systems*,
Springer Proceedings in Advanced Robotics 6,
https://doi.org/10.1007/978-3-319-73008-0_2

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, while providing guarantees of full coverage even in the face of individual failures. However, we are also aware that a common feature of a disaster scenario is that communication may be more limited than we are accustomed to in everyday life [27], so these robots cannot rely on having constant contact with the outside world, or even with each other inside the environment. We therefore focused on developing algorithms which can function under various communication restrictions, such as chemical or line-of-sight means of communicating between robots.

Our primary contributions 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. We demonstrate the algorithms' functionality through simulations and experiments using physical robots.

2 Related Work

A multi-robot system has several advantages over a single robot, including cost, efficiency and robustness [5, 11]. While a single robot can be designed to efficiently complete its task, it may then be suitable for only a small set of tasks, and if even a small part fails, the robot may be unable to complete the desired task. In contrast, a multi-robot system comprised of smaller, individually less-capable robots, with several types to complete various parts of the task, can still accomplish their goal even if some robots fail.

There are multiple methods for a team of robots to explore an unknown environment. Gage [10] proposed three types of coverage. Blanket coverage provides simultaneous coverage of the entire area. Barrier coverage sets up a complete perimeter around an area. 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.

Most coverage algorithms are focused on surveillance, and thus aim to achieve either blanket or barrier coverage. However, the number of robots required to achieve either can be prohibitively large. In contrast, sweep coverage can be accomplished with a small team, down to a single robot, if necessary, should all other robots fail [9]. 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.

An additional consideration is whether the system should be centralized or distributed. In a centralized system, a controller issues instructions and keeps the group coordinated. Stump et al. [29], used a robot as a base station, while the other robots formed a communication bridge as they moved into the unknown region. Similarly, Rekleitis et al. [26] used one robot as a stationary beacon for another robot, thus reducing odometry errors. The centralized approach also has the advantage of making it easy to create a global map, which can later be used to direct other agents, human or robot [4, 28, 31]. However, a centralized approach fails completely if the controller fails, and does not scale well.

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 [29], but using broadcast messages rather than point-to-point messages. An alternative approach is to allow the robots to separate to explore distant frontiers, and work towards each other to build a map of the environment [15]. This allows a single sweep through the environment, but also relies on precise measurements from the robots' sensors to fuse the maps together into a global whole.

Fusing the maps together is often made easier by using a grid-based approach to the coverage problem [27, 30]. Grid-based approaches can be quite effective in ensuring full coverage of an environment, and are particularly well suited to smaller, indoor spaces [11], but the size of the grid cells can require longer travel distances to achieve full coverage. In contrast, a graph based approach allows the robots more latitude in choosing where to explore [3, 18], and decomposes the environment in a way that is more tuned to that particular environment, but with less structure, there can also be more overlap in the robots' sensor coverage. Brass et al. [3] are able to provide optimal paths for two robots using their algorithms for exploration of undirected graphs, particularly in environments such as office buildings or caves. In our work, we use a graph-based approach to help provide flexibility in how the robots disperse.

Ma and Yang [21] show that the most efficient dispersion of mobile nodes is triangular, producing the maximal overall coverage with minimal overlap or gap. The dispersion formation is achieved through the nodes' local communication, in which they determine distance and bearing to their neighbors. Lee et al. [18] use online distributed triangulation to break the environment into discrete areas, which the robots can then monitor and update in both coverage and patrolling tasks. Liu et al. [19] have shown that repeated location updates can lead to better coverage over time. Similar approaches by Howard et al. [12] and Cortes et al. [7] 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 method is 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 [23] 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. These algorithms are similar to those in [7, 12, 19], but allow for greater variability in the dispersion pattern, including clusters and perimeter formations. Additional work based on insect behavior includes pheromone-based algorithms [1, 16, 22, 25], which rely on items placed in the environment for communication and navigation.

Dirafzoon et al. [8] 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 [17] developed an algorithm in which the team of robots was split into two groups, one of which remained stationary as landmarks while the other moved, and then they traded roles. 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 [14, 20]. 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. Overall, a distributed system allows an individual robot to work independently, while also sharing data with neighbors as necessary. It also does not rely on long distance communication.

3 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 aim for sweep coverage, and thus can function with a single robot, if needed (due to availability or attrition). Our distributed approach takes advantage of the robustness inherent in having multiple robots, and is not impeded by the communication restrictions, since only local communication is required. Lastly, as in 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 that may pass by later.

We assume that the robots can detect and avoid obstacles, communicate with each other in some manner (wi-fi, line-of-sight, chemical, etc.), and can 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 benefit of reducing both the likelihood of robots getting lost and the possibility that part of the environment will be overlooked. Our innovation lies in making the

algorithms independent of the type of communication used, while still making the robot team capable of achieving full coverage in an efficient manner even with some attrition.

3.1 Algorithm Details

In the Rolling Dispersion Algorithm (RDA) the robots are either *explorers*, which move into the frontier, or *sentries*, which maintain a return path to the entrance. Each robot uses connectivity with its neighbors and distance to nearby obstacles to choose which of the following behaviors it will execute on each iteration of the algorithm. We show the finite state machine, using the initials of the behaviors for the node labels in Fig. 1.

Avoid Collisions: Use proximity sensors to avoid obstacles.
Disperse: Move towards open space and the frontier, away from neighbors.
Follow Path: Accept request and follow the path to the requesting robot.
Guard: Stay in place and act as a sentry for other robots.
Retract: Dead-end reached, drop a beacon and return to the frontier.
Seek Connection: Re-establish communication with the rest of the group.

Definition 1 An exploring robot is in a dead-end when every direction in which it might move is towards an obstacle, be it a wall, a beacon, or another robot.

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 (see Definition 1), 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 all robots have moved on to the frontier, leaving the entire path marked as explored by beacons. The direction of the dispersion and exploration is primarily informed by the wireless signal intensities between agents, as in [20], 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 explored.

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. With such restrictions, it is critical to reduce the number and size of messages to be able to ensure full exploration, so the robots decide their next action based on the states of their neighbors. There are two states used only by robots, one state used only by beacons, and five states used by both.

Branch: Robot or beacon marking an intersection.
Call Path: Robot or beacon on the path to the unexplored frontier.
Explorer: Robot moving along a path to answer a call or failure notification.
Failure Path: Robot or beacon along the path leading to a location where an agent failed and needs to be replaced.
Repel: Beacon marking an area as explored, preventing repeated exploration.
Retract Path: Robot or beacon that has received notice of a retracting robot on its way.
Retractor: Robot has reached a dead-end, dropped a beacon to mark the explored area and is in the process of retracting to the nearest branch.
Sentry: Robot or beacon marking a path where there are no active notifications (call, failure, or retract).

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 be missed. 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 Fig. 1. The nodes are labeled with the initials of the states.

A robot in the Explorer state will move away from the rest of the robots into the unexplored frontier, while a robot in the Retractor state will return along the path of robots and beacons to the previous intersection (see Definition 2) and then become an Explorer again. A robot in any of the other states will remain in place, marking the path and providing other robots with the status of those robots further down the path (passing along calls for additional Explorers, or notices of retraction or failures). Robots in intersections use the state Branch to alert the Retractors that they have reached the point where there is again a frontier to explore.

Definition 2 Intersections are defined as locations where either there is an obstacle creating multiple paths (such as a corner), or where the robot could move in perpendicular directions without obstruction.

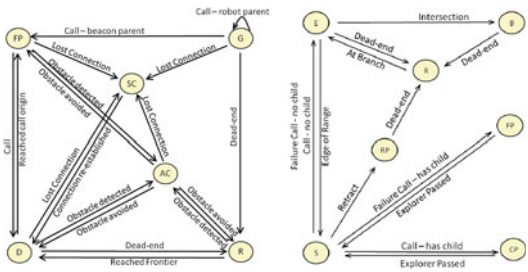

Fig. 1 (left) Rolling dispersion algorithm finite state machine. (right) Sweep exploration algorithm finite state machine

3.2 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.

Though SEA has more restrictions on communication than RDA, both algorithms operate in the same manner at their core, so the following properties and proofs apply to both algorithms. The differences in the algorithms shows in how the robots move (multiple or one at a time), and how much information is shared between robots.

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. This produces an immediate contradiction because, when an exploring robot reaches a dead-end it drops a beacon to mark the explored area. Any re-explorations are prevented by the presence of these beacons.

Second, in the case when a beacon marking an explored area fails, the area around that beacon becomes unmarked. If the failed beacon is surrounded by beacons marking the path as explored, then no robot will reach that area, so it will not be re-explored. If, however, the beacon was bordering unexplored regions, then the robots will have to re-explore the now unmarked area until reaching a dead-end, and then once again mark the area 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. Therefore, 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: first assuming that the beacons do not fail; and then assuming 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 can fail, the area surrounding a beacon's location becomes unmarked. If there are still surrounding beacons marking the area as explored, then there will be no effect on the robots' exploration. If the beacon is

neighbors with a robot or the path to the frontier or 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 have 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 algorithms achieve full exploration when only one robot explores the environment. The proof 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.

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 agent at all times, and when that connection is lost, the robots will immediately stop exploring and retreat in order to reconnect. This is required for ensuring complete coverage, because the connectivity means that robots will not miss any area, and will not re-explore areas 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. 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, means that all the still 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 this functionality.

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

3.3 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 representing locations that are separated by the distance of the communication range, in order to keep it to a finite number of nodes. We show a possible graph as an overlay on the simulation environment map in Fig. 2, though this depends on how the robots move about the environment, so subsequent runs may lead to different locations for the intersections and dead-ends. 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, and that number is bounded by the number of robots, in the case where the total number of robots is less than n'.

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 consumption, and may not be feasible 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 put bounds on the number of messages being sent, which makes the coordination easier as well. In SEA, we require only eight message types to complete the exploration, which correspond directly to the eight states listed previously. Because of this, the robots can complete the exploration with very little required in terms of information sharing, which is often one of the more expensive aspects of multi-robot systems.

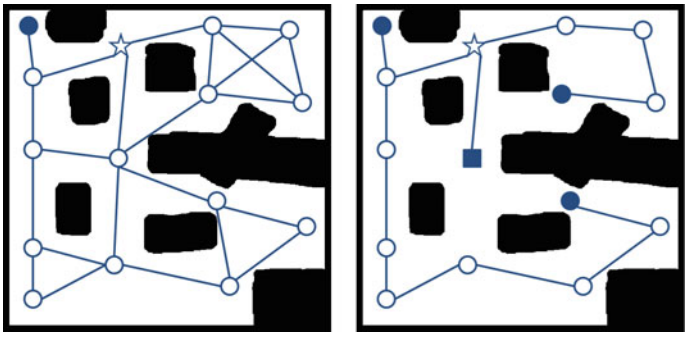

Fig. 2 (left) Cave-like environment used in simulations, showing overlay graph. (right) One possible exploration. The star marks the start, the square marks a location where exploration was begun, then left to finish a different path, resulting in the filled circles becoming dead-ends

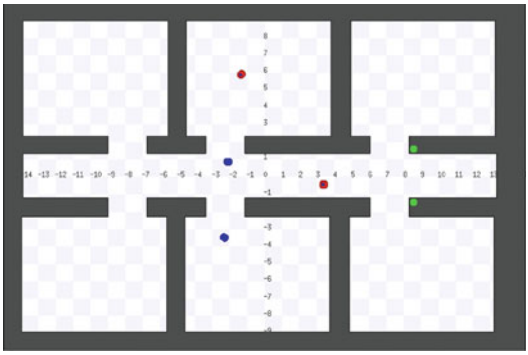

Fig. 3 A simple environment with RDA exploration partially completed. The robots (red and blue) are moving right to left, and have dropped two beacons (green) so far

4 Simulation 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 environments, shown in Figs. 2 and 3, provide two very different types of spaces–one very open, with several large obstacles at varying intervals and in non-uniform shapes that leave wide open areas and potential loops, and one a series of rooms off a single long hallway. The robots start from the same location for each run in each environment. The robots choose their directions with some randomness to avoid hitting each other, especially at the start of each run, so results are averaged over 15 runs of each environment and number of robots.

Figure 4 shows the rate of coverage for the RDA and SEA algorithms in the cave environment. While in most cases the simulations show fairly steady increase in the percentage of coverage, the RDA with 5 robots does show a plateau, 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. The SEA algorithm does not have this plateau because the robots explore along only 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

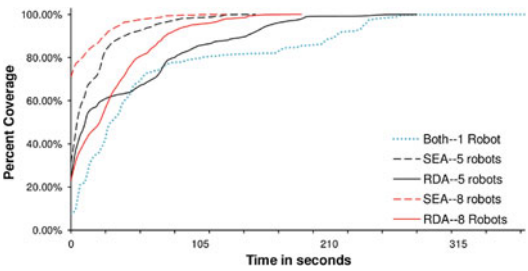

Fig. 4 Average time to full exploration for simulations using 1, 5 and 8 robots with each algorithm in the cave environment simulation

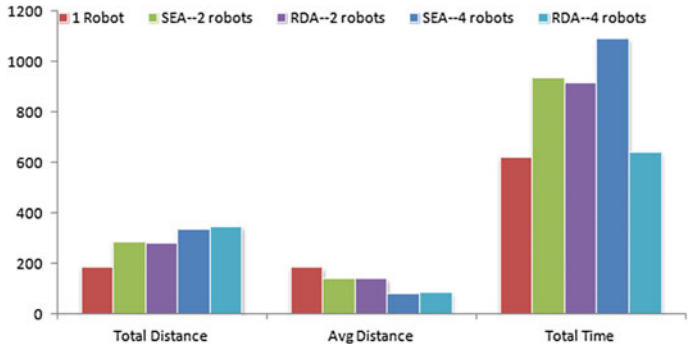

Fig. 5 Total distance, average distance per robot, and total time for simulations using 1, 2, and 4 robots with each algorithm in the office environment

the same starting percentage to full coverage, SEA is 1.35 times faster at achieving full coverage than RDA. SEA outperforms RDA in environments where there are long paths, because SEA fully explores only one path at a time, while RDA will attempt to explore many paths simultaneously, but will run into delays when robots have to be pulled from other paths to fully explore long paths.

In the office building environment, with its long corridor and single rooms off of it, both methods performed the same for 1 and 2 robots (shown in Fig. 5), because with only two robots, they are constantly leap-frogging around each other to continue exploring. However, with 4 robots RDA took significantly less time, even though the robots traveled approximately the same total distance as the 4 robots running SEA. While the robots using RDA could explore in multiple directions simultaneously, the robots using SEA had wait for each individual path and room to be cleared, and then had to move back to the corridor one at a time.

For our experiments, we used the Scribbler robots with the IPRE Fluke [13] attachment. We made use of the camera to read beacons and IR sensors to locate intersections. Due to the limitations of the robots, only SEA could be implemented. However, an advantage to using the Scribblers in testing is that we will be able to run experiments with 10–30 robots in the future. The Clearpath Jackal [6] would be a more appropriate robot for search and rescue operations.

5 Experimental Results

Our experimental environment has a more structured lay-out, similar to a house or apartment floor-plan, with small doorways and multiple rooms and hallways, shown in Fig. 6. We ran the experiments with 1, 2 and 3 robots, which was the maximum number of robots that could maneuver given the size of the environment. We measured the time to complete the exploration, number of messages sent, and distance traveled

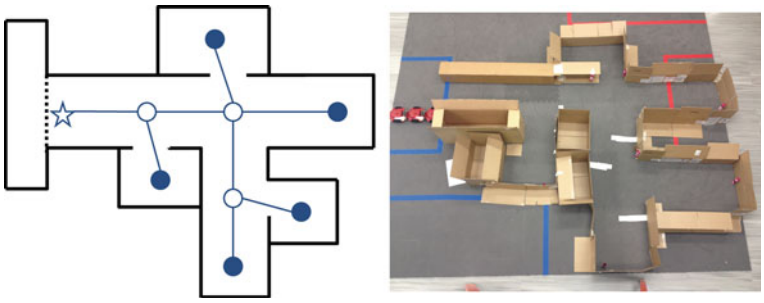

Fig. 6 (left) Overlay graph of experimental set-up (dotted line denotes start area). The star is the first point in the environment, and the filled circles mark dead-ends. (right) Experimental set-up with robots in start and cardboard walls

(see Fig. 7). SEA does not perform well in this environment due to the restrictions on movement. The short paths mean that by the time additional robots arrive, the first has completed the branch and moved onward, leading to significant overlap in areas covered.

6 Conclusion

We have presented two distributed algorithms for multi-robot exploration of unknown environments. Both algorithms make use of 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

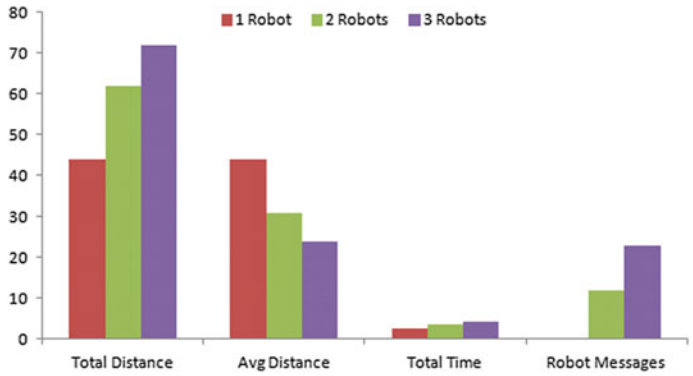

Fig. 7 Experimental results runs with 1, 2, or 3 robots using SEA

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.

In future work, we plan to test the algorithms in other types of environments, including larger ones with many loops, open areas that create the potential for loops, and varying path lengths. We will also include human interaction during the exploration and rescue stages.

References

1. Batalin, M.A., Sukhatme, G.S.: The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. IEEE Trans. Robot. **23**(4), 661–675 (2007). https://doi.org/10.1109/TRO.2007.903809
2. Birk, A., Carpin, S.: Rescue robotics - a crucial milestone on the road to autonomous systems. Adv. Robot. **20**(5), 595–605 (2006)
3. Brass, P., Cabrera-Mora, F., Gasparri, A., Xiao, J.: Multirobot tree and graph exploration. Trans. Robot. **27**(4), 707–717 (2011). https://doi.org/10.1109/TRO.2011.2121170
4. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005). https://doi.org/10.1109/TRO.2004.839232
5. Choset, H.: Coverage for robotics – a survey of recent results. Ann. Math. Artif. Intell. **31**, 113–126 (2001). https://doi.org/10.1023/A:1016639210559
6. ClearpathRobotics: Jackal ugv. Technical report (2015)
7. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004). https://doi.org/10.1109/TRA.2004.824698
8. Dirafzoon, A., Emrani, S., Salehizadeh, S.M.A., Menhaj, M.B.: Coverage control in unknown environments using neural networks. Artif. Intell. Rev. 237–255 (2012)
9. Fazli, P., Davoodi, A., Pasquier, P., Mackworth, A.K.: Complete and robust cooperative robot area coverage with limited range. In: IEEE/RSJ Int'l Conference on Intelligent Robots and Systems (IROS), pp. 5577–5582. IEEE (2010)
10. Gage, D.W.: Command control for many-robot systems. In: 19th Annual AUVS Technical Symposium, pp. 22–24 (1992)
11. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. **61**(12), 1258–1276 (2013)
12. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, pp. 299–308 (2002)
13. IPRE: Myro hardware. Technical report (2011)
14. Jensen, E.A., Gini, M.: Rolling dispersion for robot teams. In: Proceedings Int'l Joint Conference on Artificial Intelligence (IJCAI), pp. 2473–2479 (2013)
15. Khawaldah, M.A., Nuchter, A.: Enhanced frontier-based exploration for indoor environment with multiple robots. Adv. Robot. **29**(10), 657–669 (2015)
16. Koenig, S., Liu, Y.: Terrain coverage with ant robots: a simulation study. In: Proceedings Fifth Int'l Conference on Autonomous Agents, pp. 600–607. ACM, New York, NY, USA (2001). http://doi.acm.org/10.1145/375735.376463
17. Kurazume, R., Hirose, S.: An experimental study of a cooperative positioning system. Auton. Robot. **00**, 43–52 (2000)
18. Lee, S.K., Fekete, S.P., McLurkin, J.: Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers. Int. J. Robot. Res. **35**(10), 1234–1260 (2016)

19. Liu, B., Brass, P., Dousse, O., Nain, P., Towsley, D.: Mobility improves coverage of sensor networks. In: MobiHoc '05: Proceedings 6th ACM Int'l Symposium on Mobile ad hoc Networking and Computing, pp. 300–308. ACM, New York, NY, USA (2005). http://doi.acm.org/10.1145/1062689.1062728
20. Ludwig, L., Gini, M.: Robotic swarm dispersion using wireless intensity signals. In: Proceedings Int'l Symposium on Distributed Autonomous Robotic Systems (DARS), pp. 135–144 (2006)
21. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. IEEE Trans. Comput. **56**(7), 946–958 (2007). https://doi.org/10.1109/TC.2007.1054
22. Mamei, M., Zambonelli, F.: Pervasive pheromone-based interaction with rfid tags. ACM Trans. Auton. Adapt. Syst. **2**(2), 4 (2007). https://doi.org/10.1145/1242060.1242061
23. McLurkin, J., Smith, J.: Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In: Proceedings Int'l Symposium on Distributed Autonomous Robotic Systems (DARS) (2004)
24. Murphy, R.R.: Disaster Robotics. The MIT Press (2014)
25. O'Hara, K.J., Walker, D.B., Balch, T.R.: Physical path planning using a pervasive embedded network. IEEE Trans. Robot. **24**(3), 741–746 (2008). https://doi.org/10.1109/TRO.2008.919303
26. Rekleitis, I., Dudek, G., Milios, E.: Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In: Proceedings Int'l Joint Conference on Artificial Intelligence (IJCAI), vol. 2, pp. 1340–1345. Morgan Kaufmann Publishers, Inc., Nagoya, Japan (1997)
27. Rooker, M.N., Birk, A.: Multi-robot exploration under the constraints of wireless networking. Control Eng. Pract. **15**(4), 435–445 (2007)
28. Stachniss, C., Burgard, W.: Exploring unknown environments with mobile robots using coverage maps. In: Proceedings Int'l Joint Conference on Artificial Intelligence (IJCAI) (2003)
29. Stump, E., Jadbabaie, A., Kumar, V.: Connectivity management in mobile robot teams. In: Proceedings IEEE Int'l Conference on Robotics and Automation (ICRA), pp. 1525–1530 (2008). https://doi.org/10.1109/ROBOT.2008.4543418
30. Viet, H.H., Dang, V.H., Choi, S.: Bob: an online coverage approach for multi-robot systems. Appl. Intell. **42**(2), 157–173 (2015)
31. Wurm, K.M., Stachniss, C., Burgard, W.: Coordinated multi-robot exploration using a segmentation of the environment. In: Proceedings IEEE/RSJ Int'l Conference on Intelligent Robots and Systems, pp. 1160–1165 (2008). https://doi.org/10.1109/IROS.2008.4650734