In IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)
Pages 2849-2854, Lausanne, Switzerland, Oct 2002.

# An Incremental Deployment Algorithm for Mobile Robot Teams

Andrew Howard, Maja J Matarić and Gaurav S Sukhatme

*Robotics Research Laboratory, Computer Science Department, University of Southern California*
*ahoward@usc.edu, mataric@usc.edu, gaurav@usc.edu*

**Abstract**

*This paper describes an algorithm for deploying the members of a mobile robot team into an unknown environment. The algorithm deploys robots one-at-a-time, with each robot making use of information gathered by the previous robots to determine the next deployment location. The deployment pattern is designed to maximize the area covered by the robots' sensors, while simultaneously ensuring that the robots maintain line-of-sight contact with one another. This paper describes the basic algorithm and presents results obtained from a series of experiments conducted using both real and simulated robots.*

## 1 Introduction

This paper describes an algorithm for deploying the members of a mobile robot team into an unknown environment; the algorithm is designed to maximize the area covered by the robots' sensors while simultaneously ensuring that the robots maintain line-of-sight contact with one another. The algorithm is intended for use in applications such as search-and-rescue operations and emergency environment monitoring. Consider, for example, a scenario involving a hazardous materials leak in a damaged structure. We would like the members of our mobile robot team to deploy themselves throughout this structure such that the area 'covered' by the robots' on-board chemical sensors is maximized. The robots can then transmit information about the location and concentration of hazards to a base station located some distance away. A key feature of this scenario is that prior models of the environment are likely to be either unavailable, incomplete or inaccurate. This has two important consequences. First, it is not possible to compute an 'optimal' a priori deployment pattern; the deployment algorithm must instead make decisions based entirely on sensed data from the robots being deployed. Second, in the absence of prior models, it may be difficult to localize the robots (we assume that GPS is unavailable due to signal obstructions or multi-path effects). The robots must therefore use *each other* as landmarks [7, 8], which implies that each robot must retain line-of-sight contact with at least one other robot.

The deployment algorithm described in this paper is both incremental and greedy. Robots are deployed one-at-a-time, with each node making use of data gathered from previously deployed robots to determine its optimal deployment location. The algorithm is *greedy* in the sense that it attempts to determine, for each robot, the location that will produce the maximum increase in coverage area, while simultaneously ensuring that the robot remains within line-of-sight of at least one other robot. Determining the 'optimal' placement – even in a greedy sense – is a fundamentally difficult problem; the deployment algorithm described in this paper therefore relies on a number of heuristics to guide the selection of deployment locations.

The algorithm also addresses another problem: obstruction. Obstruction occurs when one of the robots being deployed cannot reach its deployment location because it is being blocked by another robot. We have developed a relatively simple resolution strategy for overcoming such obstructions that exploits the homogeneity of the robot team; put simply, each robot is allowed to recursively swap roles with any of the robots that are obstructing it.

We have conducted a series of experiments aimed at characterizing the performance of the incremental deployment algorithm, using large numbers of simulated robots. We have also conducted preliminary experiments with real robots aimed at validating the algorithm under controlled real-world conditions. The remainder of this paper describes the basic deployment algorithm in more detail and presents the results obtained from these experiments.

## 2 Related Work

The concept of *coverage* as a paradigm for evaluating multi-robot systems was introduced by Gage [4]. Gage defines three basic types of coverage: blanket coverage, where the object is to achieve a static arrangement of nodes that maximizes the total detection area; barrier coverage, where the object is to minimize the probability of undetected penetration through the barrier; and sweep coverage, which is more-or-less equivalent to a moving barrier. According to this taxonomy, the algorithm described in this paper is a blanket coverage algorithm.

The problem of *exploration* by a single robot in an unknown environment has been considered by a number of authors [14, 15, 16]. The frontier-based approach of Yamauchi *et al.* [14, 15] is particularly pertinent: this exploration algorithm proceeds by incrementally building a global occupancy map of the environment, which is then analyzed to find the 'frontiers' between free and unknown space. The robot is directed to the nearest such frontier. The deployment algorithm described in this paper shares a number of similarities with this algorithm: we also build a global occupancy grid of the environment and direct nodes to the frontier between free and unknown space. Our deployment algorithm, however, must satisfy an additional constraint: the deployment locations must be such that each robot is visible to at least one other robot.

The problem of *multi-robot* exploration has also been explored by a number of authors [1, 11, 12]. In this context, the heuristics used in this paper to select deployment locations are strikingly similar to those used by both Simmons [11] and Burgard [1] to select locations for exploration. In effect, these heuristics state that one should not only explore the boundary of known space, but that one should also bias the exploration towards regions in which a robot is likely to uncover large areas of previously unknown space. Burgard describes an adaptive algorithm for making estimates of these otherwise unpredictable quantities. The deployment problem described in this paper is closely related to the multi-robot exploration problem.

Finally, we note the problem of deployment is related to the traditional *art gallery* problem in computational geometry [10]. The art gallery problem seeks to determine, for some polygonal environment, the minimum number of cameras that can be placed such that the entire environment is observed. While there exist a number of algorithms designed to solve the art gallery problem, all of these assume that we possess good prior models of the environment. In contrast, we assume that no prior models; robots must therefore empirically and incrementally determine the structure of the environment.

## 3 The Incremental Deployment Algorithm

The incremental deployment algorithm relies on a number of key assumptions. First, we assume that the robots are homogeneous, and that every robot is equipped with a range sensor (such as a laser range finder or sonar array) and a broadcast communications device (such as wireless Ethernet). Second, we assume that the environment is static, at least to the extent that gross topology remains unchanged while the robots are deploying. We assume, for example, that open doors remain open. Note that the deployment process *itself* will modify the environment, as the robots will tend to obstruct one another. Third, we assume that the pose of every robot is known in some global (but possibly arbitrary) coordinate system. In our previous work on team localization [7, 8], we have shown how robots can determine their pose in an arbitrary global coordinate system, by using other robots as landmarks. Naturally, this localization method requires that robots maintain at least intermittent line-of-sight contact, which gives rise to an important constraint: each robot must be visible to at least one other robot at its final deployed location. Finally, it should be noted that we *do not* assume the existence of some prior model of the environment. This algorithm is intended for applications in which environment models are unavailable, incomplete or inaccurate, and a key task for the team may be to *generate* such models.

The incremental deployment algorithm is designed to maximize a single performance metric: *coverage*, i.e., the total area visible to the team's sensors. Ideally, we would like to compare the coverage produced by this algorithm with that produced by an optimal solution. Unfortunately, finding the optimal solute for any non-trivial example is extremely difficult, even when we have good a priori maps of the environment. Consequently, in this paper, we make no attempt to find such solutions.

### 3.1 Algorithm Overview

The incremental deployment algorithm has four phases: initialization, election, assignment and execution.

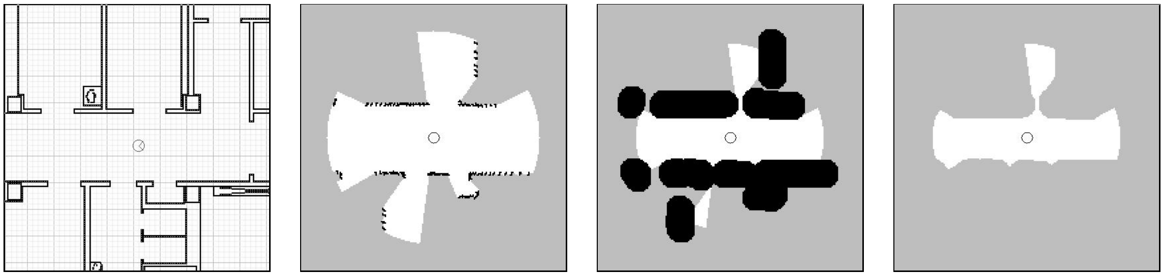
- **Initialization.** Robots are assigned one of three states: *waiting*, *active* or *deployed*. Initially, the state of all robots is set to *waiting*, with the exception of a single robot that is set to *deployed*. This latter robot provides a starting point, or 'anchor', for the team.

- **Selection.** Sensor data from the deployed nodes is combined to form a unified map of the environment. This map is analyzed to select the deployment location, or goal, for the next node.

- **Assignment.** In the simplest case, the selected goal is assigned to the first waiting robot, whose state is then changed from *waiting* to *active*. Assignment is complicated by the fact that deployed robots tend to obstruct the passage of waiting robots, necessitating a more complex assignment algorithm. This algorithm may need to re-assign the goals for any number of already deployed robot, changing their state from *deployed* to *active*.

- **Execution.** Active robots are deployed sequentially to their goal locations. The state of each robot is changed from *active* to *deployed* upon arrival at the goal.

The algorithm iterates through the selection, assignment and execution phases, terminating only when all robots have been deployed, or the environment is completely covered.



(a) (b) (c) (d)

***Figure 1:*** *(a) A fragment of the simulated environment containing a single robot. (b) Occupancy grid: black cells are occupied, white cells are free, gray cells are unknown. (c) Configuration grid: black cells are occupied, white cells are free, gray cells are unknown. (d) Reachability grid: white cells are reachable, gray cells are unreachable.*

### 3.2 Selection Phase

The selection phase determines the next deployment location, or goal. Ideally, the selected goal should maximize the coverage metric while simultaneously satisfying the visibility constraint. In practice, of course, there is no way of determining the 'optimal' goal from the incomplete information we have available (lacking a prior model of the environment, we must rely entirely on sensed data from the previously deployed nodes). We therefore eschew such reasoning and instead make use of a number of relatively simple goal selection *policies* that rely on heuristics to guide the selection process.

As a first step, sensor data from the deployed robots are combined to form an *occupancy grid* [2, 3]. Each cell in this grid is assigned one of three states: *free*, *occupied* or *unknown*. We use a standard Bayesian technique [3] to determine the *probability* that each cell is occupied, then threshold this probability to determine the state of each cell. Any cell that can be seen by one or more robots will be marked as either free or occupied; only those cells that cannot be seen by *any* robot will be marked as unknown.

The basic occupancy grid is then analyzed to produce two more grids: a *configuration grid* and a *reachability grid*. As the name suggests, the configuration grid is a representation of the robots' *configuration space* [9]. Each cell in the configuration grid can have one of three states: *free*, *occupied* and *unknown*. A cell is *free* if and only if all the occupancy grid cells lying within one robot radius are also free, and *occupied* if there are one or more occupancy grid cells within one robot radius that are occupied. All other cells are marked as *unknown*. A robot can safely be placed in any free cell in the configuration grid. Not all such cells, however, will be *reachable*; a robot may, for example, be able to see free space through an opening that is too narrow to allow passage. Therefore, we further process the configuration grid to derive the reachability grid. This is done by applying a flood-fill algorithm to free space in the configuration grid, starting from the location of each deployed robot in turn. Cells in the reachability grid are thus labeled as either *reachable* or *unreachable*.

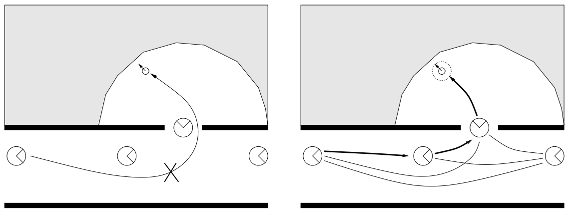
Figure 1 shows an example of the occupancy, configuration and reachability grids generated for a single node in a simulated environment. Note that the set of reachable cells is a subset of the set of free configuration cells, which is in turn a subset of the set of free occupancy cells. Thus, by selecting a goal that lies within a reachable cell, we simultaneously ensure that the deploying robot will be visible to at least one other robot, that it will not be in collision, and that there exists some path such that the robot can reach the goal.

Having determined the reachability space, the selection algorithm makes use of two heuristics to guide final goal selection: a *boundary* heuristic and a *coverage* heuristic. The boundary heuristic states that robots should deploy to the boundary between free and unknown space. This heuristic seeks to place nodes in such a way that there is minimal overlap between sensory fields, thereby maximizing the coverage metric. The coverage heuristic states that nodes should deploy to the location at which they will 'cover' the greatest area of presently unknown space. This heuristic seeks to place nodes at the location at which they have the greatest *potential* to increase the coverage area, given that we make the optimistic assumption that all unknown areas are, in fact, free space.

In and of themselves, these heuristics do not necessarily specify a unique goal. They can, however, be incorporated into a number of goal selection *policies*; we have implemented four such policies:

- **P1:** randomly select a location in free (reachable) space.
- **P2:** randomly select a location on the free/unknown boundary.
- **P3:** select the free space location that maximizes the coverage heuristic.
- **P4:** select the free/unknown boundary location that maximizes the coverage heuristic.

These policies express all possible combinations of the two heuristics, including the 'control' case (P1) in which



(a) (b)

***Figure 2:*** *(a) A typical obstruction problem, with a waiting robot unable to reach its deployment location. The gray area indicates the region of space that is not yet covered by the team. (b) The obstruction is resolved by re-assigning the deployment location to another robot.*

neither heuristic is used (the goal selection is random). Note that P4 is a special case of P3; it is included partly for completeness, and partly because it can be computed much more rapidly than P3. In Section 4, we will compare the performance of these four policies in an experimental context, and attempt to determine the relative contributions of the underlying heuristics.

### 3.3 Assignment Phase

The assignment phase attempts to assign the newly selected goal to a waiting robot. This process is complicated by the fact that robots may find themselves unable to reach some parts of the environment due to obstruction by previously deployed robots. Such obstruction becomes increasingly likely as the size of the robots approaches the size of openings in the environment. There is, fortunately, a very natural solution to this problem that exploits the homogeneity of the robot team: an obstructed robot may swap goals with the robot obstructing it. Thus, if robot $A$ is obstructed by robot $B$, robot $B$ can move to $A$'s deployment location, while $A$ replaces $B$ at its original deployment location. For complex environments, with many obstructions, this resolution strategy may need to be applied recursively: $A$ replaces $B$, $B$ replaces $C$, $C$ replaces $D$ and so on.

The assignment phase uses a slightly modified version of this procedure. First, we construct a graph in which each vertex represents a robot and each edge represents a reachability relationship between two robots (i.e. robot A can reach robot B's position, and vice versa). The length of each edge corresponds to the distance between the robots, and the goal is represented by a dummy vertex. Second, we find the shortest path from the first waiting robot to the goal (the length of any path through the graph is given by the sum of edge lengths). Finally, we mark every node on the shortest path as active, and assign each robot the goal of reaching the position currently occupied by the next robot along the path.

This algorithm is illustrated in Figure 2, which shows a prototypical graph with the shortest path highlighted. While the solution is somewhat sub-optimal (it is not strictly necessary for all robots on this path to move), the potential obstructions have been successfully resolved.

### 3.4 Execution Phase

During the execution phase, active robots are deployed to their goal locations. Robots are deployed using sequential execution; i.e., we wait for each robot to reach its goal before deploying the next robot. Robots are deployed in the order in which they were assigned goals: the first robot will move to the new deployment location, the second will move to take up the first robot's old location, and so on. Since there is only one robot in motion at any given point in time, and since the goal resolution algorithm ensures that each successive goal is unobstructed, there is no possibility for interference among robots.

## 4 Experiments

We have conducted a series of realistic simulated experiments aimed at measuring the overall performance of the algorithm, as well as determining the relative merits of the four goal selection polices described in Section 3.2. We have also conducted an experiment with real robots, for the purpose of validating the algorithm in the presence of real sensor and actuator nose.

### 4.1 A Simulated Experiment

The simulated experiments were conducted using the Stage multi-agent simulator [13, 5]. Stage simulates the behavior of real sensors and actuators with a high degree of fidelity, such that algorithms developed using Stage can usually be transferred to real hardware with little or no modification. The team for this experiment consisted of 50 robots, each equipped with a scanning laser range finder with a 360 degree field-of-view and a maximum range of 4m. The team was placed in the environment shown in Figure 3; this is a fragment of a much larger environment representing a single floor in a hospital. Localization information for this experiment was provided by the simulator, which is used in place of the team localization method described in [7, 8]; this latter method has not yet been merged with the incremental deployment algorithm.

We conducted a large set of trials, varying for each trial the selection policy and starting location. For policies P1 and P2 (which are stochastic), we conducted 10 trials from each of 10 initial location (a total of 100 trials for each policy). For policies P3 and P4 (which are deterministic), we conducted a single trial for each of the 10 initial locations.

The results of these trials are summarized in Figure 3(c), which shows the coverage (averaged across all trials for each policy) plotted against the number of deployed robots. Since all four curves are approximately linear, we

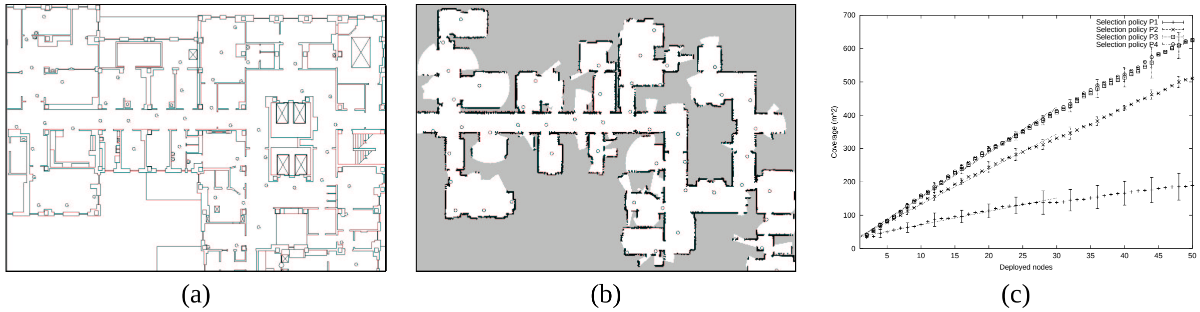



*__Figure 3:__ (a) A fragment of the simulated environment. (b) Occupancy grid produced by a typical deployment (policy P4). (c) Total coverage for selection policies P1 to P4; most of the error bars have been suppressed for clarity.*

can determine, for each policy, a value $\alpha$ that measures the average area 'covered' by each robot; i.e., $\alpha$ is such that the total coverage is approximately equal to $\alpha n$, where $n$ is the number of deployed robots. The $\alpha$ values for policies P1 through P4 are as follows:

| Policy | $\alpha$ | Policy | $\alpha$ |
|---|---|---|---|
| P1 | $4.01 \pm 0.20$ | P3 | $13.31 \pm 0.11$ |
| P2 | $10.56 \pm 0.13$ | P4 | $13.42 \pm 0.09$ |

Comparing these results, it is clear that the three goal selection policies that incorporate one or more of the heuristics described in Section 3.2 (policies P2 to P4) perform significantly better than the control case (policy P1). Policies P3 and P4, in fact, produce a 3-fold improvement over simple random deployment. It is also apparent that most of this improvement can be achieved using the boundary heuristic alone: policy P2 (which uses only the boundary heuristic) is almost as good as policy P3 (which uses only the coverage heuristic). Furthermore, policies P3 and P4 are almost indistinguishable, suggesting that the coverage heuristic will, in almost all situations, deploy nodes to the free/unknown boundary. It therefore makes sense to use policy P4 in preference to policy P3, since the latter requires much more time to compute and produces negligible improvement in coverage.

These results also suggest that there remains room for improvement in the deployment algorithm. The upper limit on $\alpha$ for a laser range-finder with a $360°$ field-of-view and range of 4m is $50.27\mathrm{m}^2$; our best policies are achieving around one-fifth of this value. While we do not expect this upper bound to be achievable in practice (nor in principle, since this bound ignores packing considerations) we would like to explore the relationship between $\alpha$, sensor range and environmental complexity. This topic is, however, beyond the scope of this paper.

### 4.2 A Real-World Experiment

The robot team for the real-world experiment consisted of four Pioneer 2DX mobile robots equipped with SICK LMS200 scanning laser range-finders. The robots have an on-board Pentium-class processor and communicate using 802.11 wireless Ethernet. Each robot runs the Player [6, 5] robot server, which allows robots to be controlled remotely over the network. For this experiment, all four robots were controlled by a single 450MHz PIII workstation.

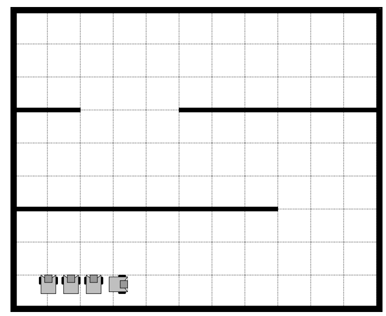


*__Figure 5:__ The environment used for the real-world experiment.*

The environment for this experiment was constructed in the laboratory from wooden partitions. The layout of the environment is shown in Figure 5. Since this environment is less than 7m across, we artificially limited the range of the laser range finders to 4m rather than their usual 8m (thus making the deployment more difficult). Localization was provided by a beacon-based system that was once again used in place of the team localization method described in [7, 8].

We conducted a single deployment trial using policy P3. Figure 4 shows a series of 'snap-shots' taken during this trial. Each snap-shot shows the occupancy grid generated by the deployment algorithm, with the position of each robot superimposed. The path taken by robots between snap-shots is also indicated. The trial starts with four robots in the bottom-left corner of the environment, with the right-most robot being used to anchor the team (i.e., this robot remains stationary). Robots deploy sequentially, pushing back the free/unknown with each successive deployment. Note that since the topology of the environment is effectively linear, the robots move in a 'Conga line': as the lead robot moves forward, the robot immediately behind it steps forward to take its place; this

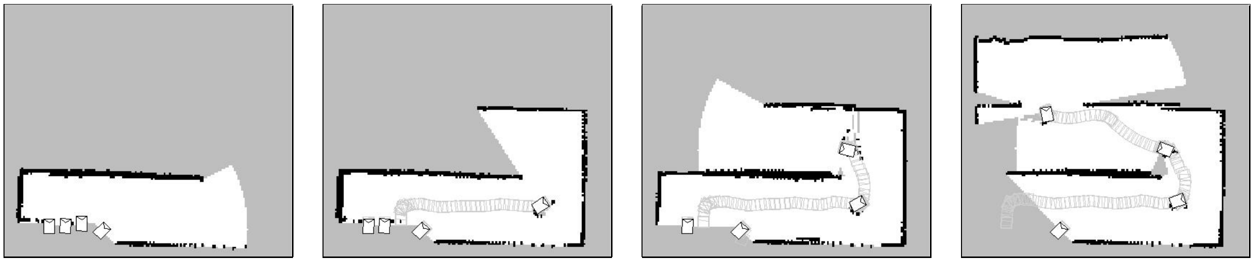


***Figure 4:*** *Results for the real-world experiment: occupancy grid generated with one, two, three and four deployed robots.*

robot is in turn replaced by the one behind it, and so on.

While this experiment is limited in scope, it clearly demonstrates that the incremental deployment algorithm can be implemented on real hardware and function under (controlled) real-world conditions.

## 5 Conclusion and Further Work

The experiments described in Section 4 clearly establish the utility of the incremental deployment algorithm and the heuristics on which it is based. Furthermore, while we have not yet fully characterized the scaling properties of the algorithm, we have empirically demonstrated that this is a practical algorithm for teams containing up to 50 robots (our simulation experiments were performed in real-time on an individual workstation).

The key weakness of these experiments is their reliance on global localization mechanisms other than the team localization method for which the incremental deployment algorithm was designed. We are currently integrating this method, and expect to demonstrate a combined system in the near future.

Our experiments are also far from exhaustive. There remain many issues to explore, including: how does the algorithm scale with team size (in terms of computational cost, bandwidth requirements, and physical deployment time)? How does the algorithm perform in different environments? And what is the impact of changing the sensor range or the physical size of the robot (thereby increasing or decreasing the number of obstructions)? These issues remain the subject of ongoing research.

### Acknowledgments

This work is supported in part by the DARPA MARS Program grant DABT63-99-1-0015, ONR grant N000140110354, and ONR DURIP grant N00014-00-1-0638.

## References

[1] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, volume 1, pages 476–81, 2000.

[2] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, 1987.

[3] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*. Morgan Kaufmann Publishers, Inc, July 1990.

[4] D. W. Gage. Command control for many-robot systems. In *AUVS-92, the Nineteenth Annual AUVS Technical Symposium*, pages 22–24, Hunstville Alabama, USA, June 1992. Reprinted in Unmanned Systems Magazine, Fall 1992, Volume 10, Number 4, pp 28-34.

[5] B. Gerkey, R. Vaughan, and A. Howard. Player/Stage homepage. http://robotics.usc.edu/player/, September 2001.

[6] B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme, and M. J. Matarić. Most valuable player: A robot device server for distributed control. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS01)*, pages 1226–1231, Wailea, Hawaii, Oct. 2001.

[7] A. Howard, M. J. Matarić, and G. S. Sukhatme. Localization for mobile robot teams: A maximum likelihood approach. Technical Report IRIS-01-407, Institute for Robotics and Intelligent Systems Technical Report, University of Sourthern California, 2001.

[8] A. Howard, M. J. Matarić, and G. S. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Switzerland, September 2002. To appear.

[9] T. Lozano-Perez and M. Mason. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.

[10] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, August 1987.

[11] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 852–858, 2000.

[12] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2000)*, volume 1, pages 321–328, 2000.

[13] R. T. Vaughan. Stage: a multiple robot simulator. Technical Report IRIS-00-393, Institute for Robotics and Intelligent Systems, University of Southern California, 2000.

[14] B. Yamauchi. Frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, pages 146–151, 1997.

[15] B. Yamauchi, A. Shultz, and W. Adams. Mobile robot exploration and map-building with continuous localization. In *Proceedings of the 1998 IEEE/RSJ International Conference on Robotics and Automation*, volume 4, pages 3175–3720, San Francisco, U.S.A., 1998.

[16] A. Zelinksy. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(2):707–717, 1992.