Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Efficient Exploration Without Localization

Maxim A. Batalin and Gaurav S. Sukhatme
Robotic Embedded Systems Laboratory
Center for Robotics and Embedded Systems
Computer Science Department
University of Southern California
Los Angeles, CA 90089, USA
maxim@robotics.usc.edu, gaurav@usc.edu

***Abstract*— We study the problem of exploring an unknown environment using a single robot. The environment is large enough (and possibly dynamic) that constant motion by the robot is needed to cover the environment. We term this the *dynamic* coverage problem. We present an efficient minimalist algorithm which assumes that global information is not available to the robot (neither a map, nor GPS). Our algorithm uses markers which the robot drops off as signposts to aid exploration. We conjecture that our algorithm has a cover time better than $O(n \log n)$, where the n markers that are deployed form the vertices of a regular graph. We provide experimental evidence in support of this conjecture. We show empirically that the performance of our algorithm on graphs is similar to its performance in simulation.**

I. INTRODUCTION

The coverage problem [1] has been defined as the maximization of the total area covered by a robot's sensors. Coverage is important in many contexts such as tracking unfriendly targets (e.g military operations), demining or monitoring (e.g. security), and urban search and rescue (USAR) in the aftermath of a natural or man-made disaster (e.g. building rubble due to an earthquake or other causes).

The problem of coverage can be considered as a *static* or more generally as a *dynamic* problem. The *static* coverage problem is addressed by algorithms [2], [3], [4] which are designed to deploy robot(s) in a static configuration, such that every point in the environment is under the robots' sensor shadow (i.e. covered) at every instant of time. Clearly, for complete static coverage of an environment the robot team should be larger than a critical size (depending on environment size, complexity, and robot sensor ranges). Determining the critical number is difficult or impossible if the environment is unknown *a priori*. *Dynamic* coverage, on the other hand, is addressed by algorithms which explore and hence 'cover' the environment with constant motion and neither settle to a particular configuration [5], nor necessarily to a particular pattern of traversal.

In this paper we consider a single robot in an unknown, planar environment. The environment is assumed to be large enough, so that complete *static* coverage of the environment is not possible. The robot must thus continually move in order to observe all points in the environment frequently. In other words we study the *dynamic* coverage problem with a single robot.

Exploration, a problem closely related to coverage, has been extensively studied [6], [7]. The frontier-based approach [6] concerns itself with incrementally constructing a global occupancy map of the environment. The map is analyzed to locate the 'frontiers' between the free and unknown space. Exploration proceeds in the direction of the closest 'frontier'. The multi-robot version of the same problem was addressed in [8]. The problem of coverage was considered from the graph theoretic viewpoint in [9], [10]. In both cases the authors study the problem of *dynamic* single robot coverage on an environment consisting of nodes and edges (a graph). The key result was that the ability to tag a limited number of nodes (in some cases only one node) with unique *markers* dramatically improved the cover time. We note that [9], [10] consider the coverage problem, but in the process also create a topological map of the graph being explored.

Our algorithm differs from the above mentioned approaches in a number of ways. We use neither a map, nor localization in a shared frame of reference. Our algorithm is based on deployment of static, communication-enabled, markers into the environment by the robot. For purposes of analysis in this paper we treat this collection of markers as nodes of a graph even though no explicit adjacency lists are maintained at each marker. There are four key differences between our algorithm and the work reported in [9], [10]:

1) We do not assume the robot can navigate from one marker to another (i.e. from one node to another in the graph). The robot does not localize itself, nor has a map of the environment (the structure of the graph corresponding to the environment is not known to the robot, nor does it construct it on the fly).
2) We assume the number of markers available for drop-off is unlimited; in [9], [10] a limited number of markers is used.
3) We assume that each marker being dropped off is capable of simple computation and communication; in [9], [10], the markers are passive - they neither

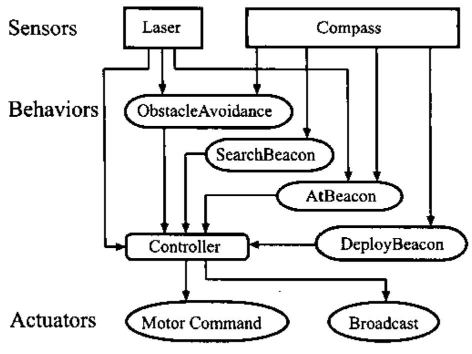


Fig. 1. System Architecture showing Robot Behaviors

compute nor communicate.

4) We do not assume that markers need to be retrieved; in [9], [10] retrieval and reuse of markers by the robot is implied.

The markers we use, act as support infrastructure which the mobile robot uses to solve the coverage problem efficiently. The robot explores the environment, and based on certain *local criteria*, drops a marker into the environment, from time to time. Each marker is equipped with a small processor and a radio of limited range. Our algorithm performs the coverage task successfully using only local sensing and local interactions between the robot and markers.

We compare the performance of our algorithm with two theoretical approaches - a random walk (RW), and a depth-first search (DFS). We also compare our algorithm and the limited marker-based graph algorithm given in [9]. Our algorithm performs significantly better than [9] and is close to the performance of DFS (known to be optimal when the graph is given). Our algorithm significantly outperforms a random walk of the environment, as expected. We report on trials with a simulator which show similar results.

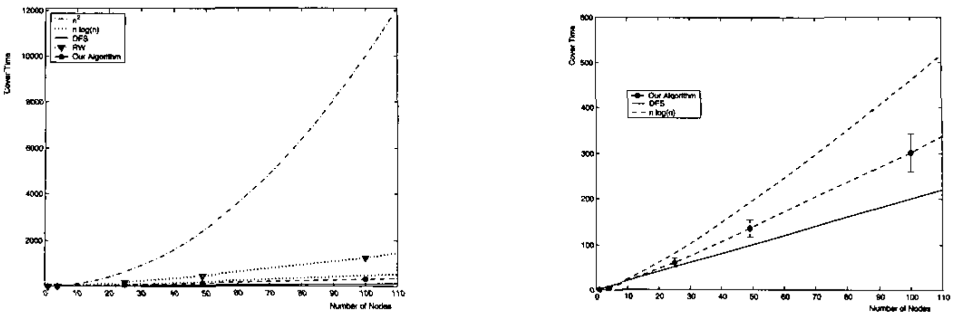
II. SYSTEM ARCHITECTURE

Our algorithm uses two entities: the markers and the mobile robot. The task of each marker is to recommend a locally preferred direction of exploration for the robot within its communication range. Thus each marker acts as a local signpost telling the robot which direction to go next. However, the robot treats this information as a recommendation, and combines this advice with local range sensing to make a decision about which direction to actually pursue.

Each marker has a state associated with the four cardinal directions (South, East, North, West). The choice of four directions is arbitrary. It implies that the marker is equipped with a 2 bit compass. For each direction, the marker maintains a state and a counter. A state can be either `OPEN` or `EXPLORED`, signifying whether the particular direction was explored by the robot previously. A counter C is associated with each direction; it stores the time since that particular direction was last explored. When the robot is in the vicinity of a marker, the marker emits a suggested direction the robot should take. This implies that the robot's compass and the marker's compass agree locally on their measurement of direction. Given the coarse coding of direction we have chosen, this is not a problem in realistic settings. The algorithm used by the markers to compute the suggested direction is simple. All `OPEN` directions are recommended first (in order from South to West), followed by the `EXPLORED` directions with largest last update value (largest value of C).

The mobile robot is programmed using a behavior-based approach [11]. Arbitration [12] is used for behavior coordination. Priorities are assigned to every behavior *a priori*. As shown in Figure 1, there are four behaviors in the system: `ObstacleAvoidance`, `AtBeacon`, `DeployBeacon` and `SearchBeacon`. In addition to priority, every behavior has an activation level, which decides, given the sensory input, whether the behavior should be in an active or passive state (1 or 0 respectively). Each behavior computes the product of its activation level and corresponding priority and sends the result to the Controller, which picks the maximum value, and assigns the corresponding behavior to command the Motor Controller for the next cycle.

The robot remembers the identification of the marker it heard most recently. If, during motion, a new marker is heard, (i.e. the robot moved to the communication zone of a different marker), `AtBeacon` is triggered. This behavior analyzes the data messages received from the current marker and orients the robot along the suggested direction. In addition, the robot sends an update message to the marker telling it to mark the direction from which the robot approached the beacon as `EXPLORED`. This ensures that the direction of recent approach will not be recommended soon. We term this the *last-neighbor-update*. After the robot has been oriented in a new direction, it checks its range sensor for obstacles. If the scan does not return any obstacles, the robot proceeds in the suggested direction, while sending an update beacon message (upon receiving this message the current marker updates the state of corresponding direction to `EXPLORED` and resets the corresponding C value). If, however, the suggested direction is obstructed (something is in the way), `AtBeacon` sends a broadcast message updating the marker with this information and requests a new suggested direction. `ObstacleAvoidance` is triggered if an obstacle is detected in front of the robot, in which case an avoidance maneuver takes place. `SearchBeacon` is triggered after `AtBeacon` chooses and positions the robot in a certain direction. The task of `SearchBeacon` is to travel a predetermined distance. `DeployBeacon` is



(a) Comparison of *Cover Time* n^2 curve, $n \ln n$ curve, DFS, RW and our algorithm

(b) A comparison between DFS and our algorithm. This graph is a magnified view of (a)

Fig. 2. Comparison of Graph Coverage Algorithms. The $n \ln n$ curve is shown for reference.

triggered if the robot does not receive a suggestion (i.e. a recommended direction to traverse) from any beacon after a certain timeout value. In this case the robot deploys a new beacon into the environment. Note that the algorithm and the architectures for the robot and marker are similar to those presented in [5]. The major difference is the addition of the *last-neighbor-update* rule, which reduces redundant suggestions and hence, improves coverage performance.

III. THE GRAPH MODEL

For purposes of analysis, we consider an open environment (with no obstacles). In this case, given our marker deployment strategy described in the previous section, we can model the steady state spatial configuration of the markers as a regular square lattice. In the general case the deployed nodes would form a graph $G = (V, E)$, where V is a set of vertices (e.g. non overlapping areas of the environment) and E is a set of edges which connect areas of the environment. The *cover time* [13], is the time it takes a robot to visit every node in the graph. The problem of coverage on the graph is to minimize the average *cover time*, considering every element of V as a starting point.

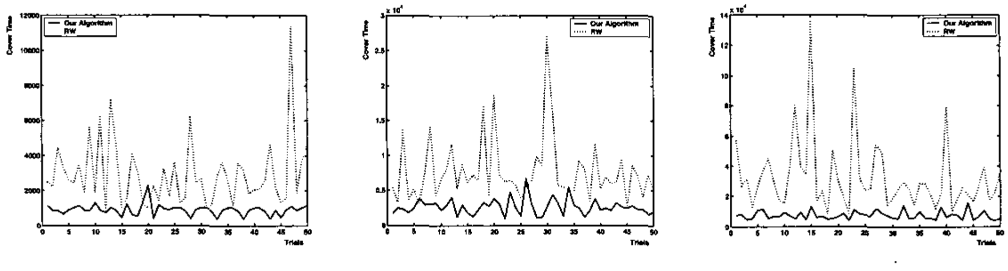
In the most simple case where the environment is unknown, and localization cannot be used, and there are no *markers* available, the problem of coverage can be solved by random walk (RW). It has been shown [13] that the *cover time* of a random walk on a regular graph of n nodes is bounded below by $n \ln n$ and above by $2n^2$. If we assume that passive markers can be used, and the graph $G = (V, E)$ is known (a topological map is available) and the robot has *markers* of three independent colors, then the problem of coverage can be solved optimally by applying DFS which is linear in n. DFS assumes that all resources are available - markers, map, localization and perfect navigation.

In [9] the problem of coverage is considered in the context of mapping a graph-like environment with n vertices. Their algorithm explores the environment and constructs a topological map on the fly. The assumptions of the algorithm are that the robot has $k (k < n)$ markers, and perfect localization and navigation within the graph. The *cover time* of their algorithm is bounded by $O(n^2)$. It is important to note that the problem addressed in [9] is more complex than simple coverage, since they build a map while exploring.

We conducted experiments running RW, DFS and our algorithm on graphs with $n = 25, 49$ and 100 nodes. For every experiment each vertex of the graph was tried as the starting point. We conducted 50 experiments per starting point. The average time over all experiments was computed. The results of the experiments are presented in Figure 2. The figure also shows the $n \log n$ curve and the n^2 curve for reference. These experiments lead us to

Conjecture 1: *The asymptotic cover time of our algorithm is less than $O(n \log n)$.*

While our algorithm is designed for coverage, it can be applied to the problem of mapping as well. In order to construct a complete map of a graph G under the assumption that a robot executes our algorithm, a robot has to visit every vertex of the graph and follow every direction of every *marker* (which would guarantee that every edge is traversed and mapped). Suppose our algorithm executes on a graph G once. It is obvious that every vertex is covered and at least one direction per *marker* is marked as explored. Thus, after the first execution of the algorithm, the number of unexplored directions at every vertex is at most $d - 1$, where d is the maximum degree in G. Note, that at a given vertex, while there are unexplored directions, an algorithm will choose one in sequence.



(a) $25m^2$ (b) $49m^2$ (c) $100m^2$

Fig. 3. Simulation results for different environment sizes across 50 trials. Our algorithm consistently outperforms a random walk by an order of magnitude.

Hence, after at most d executions of the algorithm every vertex would be covered and every direction would be marked as explored, implying that every edge is covered as well. Thus, the mapping time MT can be bounded by:

$$MT \leq d \times \max_i(CT_i) \tag{1}$$

where d is the largest vertex degree (4 in our case) and CT_i is the $i - th$ cover time. This leads us to

Conjecture 2: *Our algorithm produces a map of the environment in asymptotic time faster than $O(n \log n)$.*

Let us consider different tradeoffs between the above mentioned techniques. As mentioned earlier, the clear performance boundaries for coverage task are given by RW (upper) and DFS (lower). The more interesting comparisons are between our algorithm and DFS and our algorithm and an algorithm with a limited number of passive markers [9].

Figure 2b shows that the asymptotic performance of our algorithm is similar to DFS. Note that in order to determine the color of neighboring vertices and to navigate perfectly from node to node, DFS assumes that the map of the environment is available and the robot is localized. Our algorithm, on the other hand, does not have access to global information and the robot does not localize itself. The *markers* used in our algorithm are more complicated than those used in DFS, and the cover times are asymptotically somewhat larger than the cover times of DFS.

In [9] the algorithm builds a topological map of the environment and assumes perfect navigation (and thus, localization) on the graph within constructed map. The *markers* are very simple (the only function is to mark the vertex) and the robot cannot differentiate between them. In addition to *markers* the algorithm assumes that there exists a local enumeration of edges. The cover time of this algorithm, however, is bounded by $O(n^2)$. Our algorithm, on the other hand, does not have a map and the robot does not localize itself. Another important difference is that we assume that the number of *markers* available to us is equal to the number of vertices. In addition, the *markers* used in our algorithm are more complex, since they keep a certain state per direction, and are uniquely identifiable. The cover time of our algorithm, however, is conjectured to be less than $n \log n$. Thus, the apparent trade off is using a large number of "smart" *markers* (and no global information or localization) vs. a limited number of simple *markers* (with mapping and partial localization within the graph). The cover time achieved by our algorithm is clearly better. However, if the *markers* are a precious resource, the algorithm described in [9] would be preferred.

IV. SIMULATION EXPERIMENTS

In our experiments we used the Player/Stage [14], [15] simulation engine populated with a simulated Pioneer 2DX mobile robot equipped with two 180° field-of-view planar laser range finders positioned back-to-back (equivalent to a 2D omnidirectional laser range finder), wireless communication and a set of markers. The implementation of the algorithm in simulation proceeds as follows. The robot explores the environment in a direction suggested by the last heard marker until it does not receive a message from any of the markers. In this case a new marker is deployed into the environment. The robot continues to explore as long as there are OPEN directions left. If all regions are EXPLORED, then the robot picks the direction which was least recently explored. As discussed in Section II, decisions of which direction to explore next are made by the markers. The robots, however, may alter those decisions if real world observations (through laser data analysis) of obstacles are made. In addition, the *last-neighbor-update* rule prevents the robot from going back in the direction from which it came recently. Values for communication radius and range of the laser were set to 1500mm in the simulations. Figure 3 shows the *cover*

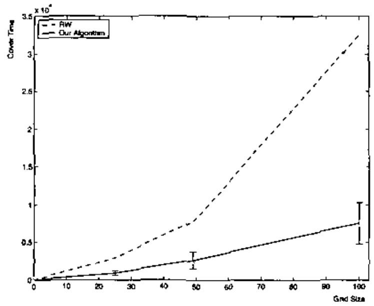


Fig. 4. Average cover times for three different grid sizes in simulation. Environment sizes are $25m^2$, $49m^2$ and $100m^2$.

times for the RW and our algorithm on environments of different sizes: $25m^2$, $49m^2$ and $100m^2$. For every grid size 50 experiments were conducted for both algorithms. The experiments show that our algorithm outperforms RW and is more stable. In addition, Figure 4 shows the average *cover times* for three different grid sizes. Note the direct correspondence between the results obtained on the graph world (Figure 3) and the results of the simulation. Both support our conjecture that the cover time of our algorithm is less than $O(n \log n)$.

V. DISCUSSION

We presented an algorithm for the *dynamic* coverage problem using a robot. We examined the trade offs that should be considered in choosing one algorithm over the other to solve this problem. The bounds for the coverage task are given by random walk (the robot has no information and explores randomly) and depth first search (a map of the environment is available in the form of a graph) which solves the problem optimally.

The data shown in Figure 2, suggest strongly that *our algorithm asymptotically outperforms the k marker* algorithm presented in [9]. This is due to two reasons. In [9] it is assumed that the number of *markers* is limited and that the localization is perfect within the topological map. Because of the limitation imposed on the number of *markers*, their approach assumes that the robot is capable of not only detecting the *markers*, but also of retrieving them. Our algorithm, on the contrary, assumes that the number of *markers* is not limited[1]. In addition we restrict our solution to the case where neither a map nor localization is used.

In addition, it is shown in [9] that if the number k of available *markers* reduces, the *cover time* increases rapidly. Therefore, in dynamic environments the performance of the algorithm decreases drastically even if one *marker* is destroyed. Whereas in our algorithm such a problem does not exist, since a new *marker* will be deployed in place of the destroyed one automatically.

[1]This assumption might be reasonable in the light of modern technology [16]

Fig. 6. Screen shots of a preliminary experiment with a robot and markers

We verified the performance of our algorithm and its asymptotic behavior in simulation. There exists a direct correspondence between the results obtained from the theoretical analysis (coverage on the graph) and the data from simulation experiments. Note also, that even though the lattice grid was considered as a graph environment for the theoretical analysis, in practice the network of deployed markers is not required to be a perfect grid. Figure 5 shows a series of screen shots taken from one of the trials of the simulation in the $49m^2$ environment. Note also that the performance of our algorithm is not affected, since it does not rely on localization or mapping.

VI. CONCLUSION AND FUTURE WORK

The theoretical analysis on graphs and verification in simulation shows that trade offs in the assumptions can affect *cover time* significantly. Simple algorithms like RW or DFS can be used for coverage, but only in the extreme cases as described above. In case, where mapping and localization are not available, but the number of available *markers* is unlimited, our algorithm appears to outperform others.

Currently we are conducting thorough real-robot experiments to validate our algorithm. Figure 6 shows snapshots from an early trial with a physical robot deploying markers.

In future work, we will investigate a formal bound on the performance of our algorithm. In addition, we plan to exploit the deployed markers for other behaviors. One example is recovery, when after being deployed, every robot uses the network to return to "home base". We have made some progress towards this goal [17]. The propagation of information through the network of markers could also dramatically increase performance of the coverage algorithm itself (e.g. by dynamically adjusting the marker drop-off distance, optimally guide the robot towards less explored parts of the environment, etc.).

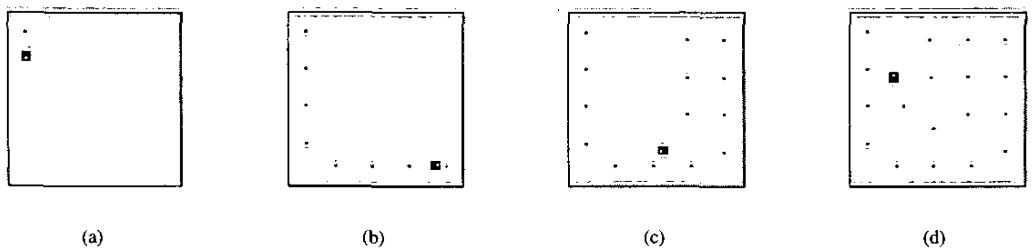


Fig. 5. Deployment of markers in a representative simulation trial. Note that due to noise added in simulation, the deployed nodes do not form a perfectly square lattice.

VII. ACKNOWLEDGMENTS

This work is supported in part by DARPA grant DABT63-99-1-0015 under the Mobile Autonomous Robot Software (MARS) program and NSF grants ANI-0082498, IIS-0133947, and EIA-0121141.

VIII. REFERENCES

[1] D. W. Gage, "Command control for many-robot systems," in *the Nineteenth Annual AUVS Technical Symposium*, Huntsville, Alabama, USA, 1992, pp. 22–24.

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

[3] 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. of 6th International Symposium on Distributed Autonomous Robotic Systems*, Fukuoka, Japan, 2002, pp. 299–308.

[4] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems*, Fukuoka, Japan, 2002, pp. 373–382.

[5] M. A. Batalin and G. S. Sukhatme, "Sensor coverage using mobile robots and stationary nodes," in *SPIE2002*, 2002, vol. 4868, pp. 269–276.

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

[7] A. Zelinsky, "A mobile robot exploration algorithm," in *IEEE Transactions on Robotics and Automation*, 1992, vol. 8, pp. 707–717.

[8] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun, "Collaborative multirobot exploration," in *Proc. of IEEE International Conferenceon Robotics and Automation (ICRA)*, 2000, vol. 1, pp. 476–481.

[9] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," in *IEEE Transactions on Robotics and Automation, 7-6*, 1991.

[10] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," in *Annual ACM Symposium on Theory of Computing (STOC '98)*, 1998.

[11] M. J. Mataric, "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents*, vol. 9, no. 2-3, pp. 323–336, 1997.

[12] P. Pirjanian, "Behavior coordination mechanisms - state-of-the-art," Tech. Rep. IRIS-99-375, Institute for Robotics and Intelligent Systems, University of Southern California, October 1999.

[13] L. Lovasz, *Random Walks on Graphs: A Survey*, vol. 2 of *Combinatorics, Paul Erdos is Eighty*, pp. 1–46, Keszthely, Hungary, 1993.

[14] B. P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, and M.J. Mataric, "Most valuable player: A robot device server for distributed control," in *IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS)*, Wailea, Hawaii, 2001.

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

[16] K. S. J. Pister, J. M. Kahn, and B. E. Boser, "Smart dust: Wireless networks of millimeter-scale sensor nodes," *Electronics Research Laboratory Research Summary*, 1999.

[17] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," in *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, Palo Alto, 2003.