The Sentinel Algorithm: Distributed Dynamic Coverage

John Corwin, Ana Cerejo, and Diego Montenegro
{john.corwin,ana.cerejo,diego.montenegro}@yale.edu

Abstract

Sensor networks are predicted to revolutionize the world as they draw us increasingly closer to the Holy Grail of ubiquitous computing. However, sensor networks also introduce new challenges. In our project we will focus on addressing the fundamental issue of coverage. The coverage problem deals with how well a target region is monitored or tracked by sensors. To date work in this field has focused on algorithms for node deployment to statically cover an area. We introduce the concept of dynamic coverage, based on the ideas of exploration and achieving a blanket coverage over time. We then present the Sentinel algorithm for dynamic coverage, which allows a limited number of nodes to efficiently explore and provide coverage of an unknown environment.

I. Introduction

This paper considers the deployment of a mobile sensor network into an unexplored environment. This is of great use in many applications, like search and rescue, manipulation in hazardous environments, and others. Our algorithm, inspired by the work of Francesco Bullo [1], is an emergent process which guarantees that when a group of nodes are deployed in an unknown region, they will be able to explore and track every point in a monitorable region over time. Moreover, when possible, these nodes will create a static coverage of the environment. In contrast, in [1], exploration is of secondary importance to static coverage of the environment. Even when there exist sufficient nodes to entirely cover a monitorable area, large portions of the map may be unexplored at the convergence of the algorithm. The limitation of this approach is that it can neither guarantee that all areas of an environment will be discovered nor guarantee that they will be monitored by a deployed network.

Our algorithm is emergent, because the combination of each node's information is what gives us the global view of the environment. This shared information also allows each node to compute its new goals which allow it to reach new unexplored regions or regions that have not been explored for some time. For these goals, each node creates an optimal path that does not hinder other nodes. In other words, they act as if they were synchronized.

Coverage can be considered as the measure of quality of service of a sensor network. There are three basic types of coverage:

1) Blanket coverage, where the objective is to achieve a static arrangement of nodes that maximizes the total detection area.
2) Barrier coverage, where the objective is to minimize the probability of undetected penetration through the barrier.
3) Sweep coverage, which is essentially equivalent to a moving barrier.

Our paper focuses on the first kind. We try to maximize coverage by deploying dynamic nodes that gather information about the environment, and if it is determined that the environment can be covered statically, the nodes arrange themselves to do so. If not, a dynamic coverage is achieved by the moving nodes.

Our paper is designed as follows: we start by discussing some related work that has been done in both static coverage and collaborative multi-robot exploration. We proceed to describe our algorithm: the model, simulation environment, single-node case, multi-node case and switching to static coverage. We then show several simulations, and discuss how our protocol outperforms the previous algorithms. Finally, we look at some open problems and future work in the field.

II. Related Work

A. Collaborative Exploration

In recent years, significant research efforts have focused on both motion planning and coordination problems for multi-vehicle systems. Below we discuss two approaches to the problems.

Simmons et al. [4] present a technique for coordinating multiple robots in their task of exploring and mapping unknown indoor environments. Their approach consists of distributing the computation among the individual robots and asynchronously integrating their results by performing some global computation over the data. A central mapper is in charge of integrating the individual maps to create a consistent global map. Similarly, each individual robot constructs "bids", which describe their estimates of the expected information gain and costs of traveling to various locations. The central authority receives the bids and assigns tasks in an attempt to maximize overall utility, while trying to minimize overlap in coverage by the robots. The problem with this approach is the need of a central authority to assign tasks to each robot and perform global computations, a problem that our design does not confront.

Burgard et al. [5] consider the problem of collaborative exploration of an unknown environment by multiple robots. The proposed technique uses a map which is built based on the data sensed by the individual robots. During the path selection phase of the algorithm, the utility of unexplored positions are considered along with travel costs. The utility of any unexplored position is reduced as soon as any robot chooses a target position in its visibility range. By trading off the utility and costs of unexplored positions, the proposed approach achieves the coordination of the robot group during exploration. Our algorithm relates to this algorithm in the sense that regions are also given utility values, and the higher utility, the more likely a sensor will move toward the target coverage area.

B. Static Coverage

Howard et al. [2] address the mobile sensor network deployment problem by using potential fields. Their goal is to have sensors spread into an unknown environment from an initial compact configuration in a manner that maximizes the total area covered by the network. The potential-field-based approach assumes that each node in the network has sensing, communication, computation, and locomotion capabilities. Each node is treated as a virtual particle which is subject to virtual forces, repulsive and viscous friction forces. The repulsive forces repel the nodes from each other and from obstacles, and enable nodes to quickly spread from their initial configuration. The viscous friction force ensures that the network will eventually reach a state of static equilibrium, thereby enabling nodes to conserve energy. Blanket coverage is an emergent property of this algorithm.

Bullo et al. propose an asynchronous, distributed, and convergent algorithm to address the coverage optimization problem. Their approach requires that each node compute its Voronoi region as well as the centroid of the Voronoi cell, and to then move toward the centroid. A Voronoi region Vf associated with a feature f of a polyhedron P is a set of points exterior to P which are closer to that feature than to any other. If a point p on object PA lies inside the Voronoi region of fB on object PB, then fB is a closest feature to the point P and vice versa for a Voronoi region of fA. This approach further requires that nodes have sensing and communication capabilities. Although the algorithm is guaranteed to converge, the state to which it converges is not guaranteed to be the optimal blanket cover achievable by the network's nodes. Unlike our algorithm, this approach does not guarantee that all points in a region will be covered.

Bullo et al. state that their algorithm is adaptive, distributed, asynchronous, and verifiably correct.

1) Adaptive: since they provide the network with the ability to address changing environments, sensing task, and network topology.
2) Distributed: in the sense that the behavior of each vehicle depends only on the location of its neighbors. Also, the algorithm does not require a fixed-topology communication graph, i.e., the neighborhood relationships do change as the network evolves.
3) Asynchronous: because the algorithms can be implemented in a network composed of agents evolving at different speeds, with different computation and communication capabilities.
4) Verifiable Asymptotically Correct: because the algorithms guarantee monotonic descent of the cost function encoding sensing task.

The basic outline of the algorithm is as follows:

Each node that is deployed in the environment computes its own Voronoi region and the centroid of this region. An ordinary Voronoi diagram is formed by a set of points in the plane called the generators or generating points. Every point in the plane is identified with the generator which is closest to it by some metric. The common choice is to use the Euclidean distance metric:

$$\|X_1 - X_2\|^2 = (\mathrm{x}_1 - \mathrm{x}_2)^2 + (\mathrm{y}_1 - \mathrm{y}_2)^2$$

where $X_1 = (\mathrm{x}_1, \mathrm{y}_1)$ and $X_2 = (\mathrm{x}_2, \mathrm{y}_2)$ and are any two points in the plane. The set of points in the plane identified with a particular generator form that generator's Voronoi region, and the set of Voronoi regions covers the entire plane. Figure 1 illustrates a set of generating points and their associated Voronoi regions.

For all i, node$_i$ performs the following:

1) Determine Voronoi cell V_i
2) Determine the centroid CV$_i$ of V$_i$
3) Set u$_i$ = sat(CV$_i$ - p$_i$)

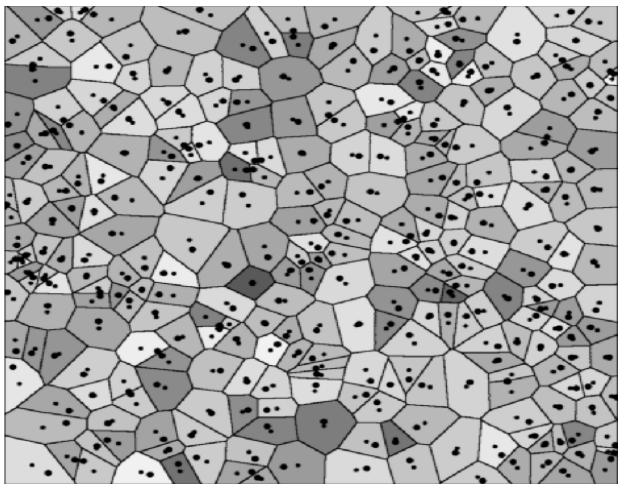

Fig. 1. Voronoi diagram with generators (large dots) and centroids (small dots)[6]

node identifier	A unique integer identifying the node
region boundaries	boundaries of the sensing environment
starting location	starting location in the sensing environment
movement speed	maximum movement speed
sense range	maximum sensing range
communication range	maximum node-to-node communication range
max-steps	number of steps to move along current path before recomputing

Fig. 2. Node initial configuration parameters

III. Sentinel Algorithm

We first describe our model and simulation environment. We then describe the Sentinel algorithm for the case of a single node, and then extend this algorithm to multiple nodes. Finally, we give criteria for switching to a static coverage algorithm when it is appropriate to do so.

A. Model and simulation environment

Our simulation environment consists of a bounded two-dimensional area containing some open space and potentially many obstacles. Nodes are placed at a particular starting location and are given the bounds of the entire area, but the topology of the area is initially unknown. We assume that nodes are able to move in two dimensions at up to a given maximum speed and have a laser or IR range finder to detect obstacles. Nodes also have a *sense range*, the range at which the node can perform its sensing functions, and a *communication range*, specifying the range of communication with other nodes. Sensing and communication are both line-of-sight — nodes can not sense or communicate through obstacles. Points within a node's sensing range that are not otherwise blocked by an obstacle are considered "covered" by that node. Figure III-A shows the parameters for a node's initial configuration.

We chose to implement our algorithm and simulation environment in a discrete (non-continuous) manner. Thus, the entire sensing area is divided into a matrix of uniformly sized regions. If an obstacle covers only part of a discrete region, we consider the entire region to be obscured. We feel that this approach is not only easier to implement in a simulation environment, compared to a geometry-based continuous implementation, but may also be more practical for actual sensor network deployments as well, as it relieves each node of the burden of computing the geometry of an unexplored area. Selecting an appropriate granularity involves a trade-off between accuracy and computational complexity, and is an interesting problem in its own right. Key selection criteria are complexity of obstacles in the sensing environment, computational resources available to each node, and the relative size of targets or features being sensed.

B. Single node algorithm

A node keeps two matrices to represent the state of the outside world: a *topology matrix* and a *utility matrix*. In each matrix, the entry (i, j) contains a piece of information about the $(i, j)^{th}$ region of discretized space.

The topology matrix keeps track of which areas of the map have been explored and the location of obstacles. Each element in the matrix has one of three values:

- REACHABLE – the region has been explored and does not contain an obstacle
- OBSTACLE – the region has been explored and does contain an obstacle
- UNKNOWN – the region has not been explored

Since nodes do not initially have any information about the environment, each entry of the matrix is initialized to UNKNOWN. When a node moves, the node updates its topology matrix with the status of newly explored areas. In our simulation environment, we assume a node can detect obstacles within its coverage range that are not otherwise blocked — a node can sense the presence of walls, but can not sense the area behind a wall.

The node's utility matrix keeps track of the utility or value of covering a particular area in the environment. At each time step in the algorithm, the utility matrix is updated with the following function:

$$U_{i,j} = \begin{cases} 0 & \text{if } (i,j) \text{ is covered} \\ U_{i,j} + 1 & \text{otherwise} \end{cases}$$

This states that areas currently covered have zero utility, while areas that are not covered become increasingly

At each time step, execute the following phases:

Update phase
for each region (i, j) in the environment **do**
  update topology matrix entry $T_{i,j}$ according to the topology update function
  update utility matrix entry $U_{i,j}$ according to the utility update function
end for
if path-step $>$ max-steps
  or we have reached the current target **then**
  compute a new path using the single-node path algorithm
  set path-step $\leftarrow 0$
end if

Movement phase
move one step along the current path
set path-step $\leftarrow$ path-step $+1$

Fig. 3. Single-node sentinel algorithm

valuable. The increasing utility of non-covered areas are a key component of the desired "patrolling" behavior of a node.

The value of a node covering a particular area is equal to the utility of covering the area, discounted by the cost of moving to this area. The following formula gives the discounted value of covering an area (i, j) for a node at position (x, y):

$$V_{i,j} = \frac{U_{i,j}}{path[(x, y), (i, j)]}$$

where $path[(x, y), (i, j)]$ is the minimum length path from region (x, y) to region (i, j).

The node must compute the maximal $V_{i,j}$ while taking path length and obstacles into account. To accomplish this, the node performs a breadth-first search over all regions marked REACHABLE or UNKNOWN in its topology matrix, starting with its current position. The search does not progress through regions marked OBSTACLE. The node then selects the region and associated path of greatest value as its next destination. By treating UNKNOWN regions as potentially being REACHABLE, the node will attempt all paths to reach an unknown area with high utility.

After the maximal $V_{i,j}$ is found, the path used in the computation, $path[(x, y), (i, j)]$, is stored as the node's current path. The last step on this path is region (i, j), which we will refer to as the node's current target. Figure 3 shows the Sentinel algorithm for a single node.

One important property of the Sentinel algorithm is that every reachable area will eventually be covered by a node. This is easily shown in the single-node case. Let (i, j) be any region in the environment that is reachable from the node's starting location. The minimum length path to (i, j) from any other reachable area in the environment is bounded by the total number of regions. However, the utility of (i, j) is unbounded — as long as (i, j) is not covered, $U_{i,j}$ will continue to increase. Thus, $V_{i,j}$ will eventually have the greatest value, and will thus be chosen as the node's next destination.

C. Multi-node algorithm

We now consider multiple nodes simultaneously operating within the same environment. Ideally, nodes should coordinate their efforts in order to more rapidly explore and provide greater simultaneous coverage of the environment. To achieve this goal, we add a communication phase to the single-node algorithm, which occurs at each time step before the update and movement phases take place.

The communication phase proceeds as follows: For each node s with neighbors $n \in N$, s and n exchange compressed representations of their topology and utility matrices, as well as their current paths being traversed.

s then updates its topology matrix with the following function:

$$Ts_{i,j} = \begin{cases} Tn_{i,j} & \text{if } Ts_{i,j} = \text{UNKNOWN and} \\ & \exists_{n \in N} Tn_{i,j} \neq \text{UNKNOWN} \\ Ts_{i,j} & \text{otherwise} \end{cases}$$

s updates its utility matrix as follows:

$$Us_{i,j} = \min[\forall_{n \in N} Un_{i,j} \cup \{Us_{i,j}\}]$$

This merging of topology matrices allows nodes to obtain information about the environment through other nodes' exploration. Merging the utility matrices encapsulates the logic that if a neighbor of node s has covered a particular region, it is equivalent to s covering that region itself.

The coverage area of each sensor node would not be well-used if multiple nodes choose the same region as their next target. Thus, we need a mechanism for nodes to divide the task of providing a dynamic cover. The goal is for nodes to defer to other nodes when selecting the next target.

To accomplish this, we make use of the node identifiers to assign a priority level to each node, by assigning higher priorities to nodes with lower ID numbers. This gives us a total ordering of the nodes. We then use a modified definition of the node's value function $Vs_{i,j}$. For node s with neighbors N,

$$Vs_{i,j} = \begin{cases} 0 & \text{if } ncovered(s, N, i, j) \\ \frac{Us_{i,j}}{path[(x,y),(i,j)]} & \text{otherwise} \end{cases}$$

where

$$ncovered(s, N, i, j) =$$
$$\exists_{n \in N}\ \exists_{(x,y) \in n.path} \left[\begin{array}{l} (n.id < s.id) \wedge \\ n \text{ covers } (i, j) \text{ when at } (x, y) \end{array} \right]$$

position	node's current position
topology matrix	node's exploration status
utility matrix	node's coverage status
current path	current path to follow
path list	paths of neighboring nodes

Fig. 4. Node state (in addition to configuration parameters)

Each node s executes the following phases at each time step:

Communication phase
for each neighbor n of s **do**
  exchange and merge topology matrices with n
  exchange and merge utility matrices with n
  if n has a lower ID, store n's path
end for

Update phase
for each region (i, j) in the environment **do**
  update topology matrix entry $T_{i,j}$ according to the topology update function
  update utility matrix entry $U_{i,j}$ according to the utility update function
end for
if path-step $>$ max-steps
  or s has reached the current target **then**
  compute a new path using the multi-node path algorithm
  set path-step $\leftarrow 0$
end if

Movement phase
move one step along the current path
set path-step $\leftarrow$ path-step $+1$

Fig. 5. Multi-node sentinel algorithm

Figure 5 shows the sentinel algorithm for multiple nodes.

Having multiple nodes does not affect the guarantee that every reachable point will eventually be covered. Let L be the node with highest priority (lowest ID number). Thus, the test $(n.id < L.id)$ in $ncovered(L, N, i, d)$ will never succeed, so L will not take any other nodes' paths into account when selecting a target. Let (i, j) be a region reachable by L. One of the following must occur:

1) $Us_{i,j}$ will grow large enough such that (i, j) is selected as a target, thus (i, j) will be covered by L, or
2) the value of $Us_{i,j}$ decreases as a result of a merge operation with a neighboring node. Since $Us_{i,j}$ has decreased, it follows that another node has covered (i, j).

D. Switching to static coverage

In many situations, a static cover may be more desirable than a dynamic cover. For example, if nodes have limited energy, a static cover would be more energy-efficient as nodes would not have to move after their positions have converged. However, we only want to switch to a static cover if the nodes can provide a reasonably good static cover of the current environment.

We designate the node with ID 0 as the leader node L. L determines if the network of nodes should switch to a static coverage algorithm by checking to see that the following criteria are met:

1) The reachable environment has been fully explored from L's perspective. This is satisfied if a breadth-first search on L's topology map reveals no regions marked UNKNOWN.
2) L is part of a connected subgraph of nodes G, based on the ability of nodes to communicate with each other, such that the sum of the maximum coverage of each node in G is at least as large as the size of the reachable area.

$$\sum_{n \in G} \pi * (n.\text{coverage})^2 >= \text{reachable-area} * oc$$

where oc is a constant representing the over-coverage factor.

If the above criteria are met, L broadcasts a message that directs all other nodes to switch to a voronoi-region based static coverage algorithm.

IV. Experimental Results

In each of the following demonstrations of the Sentinel algorithm, red (darker) areas represent regions of high utility, and lighter areas represent lower utility. The utility shown is a global merge of each individual node's utility matrix using the merge function from the multi-node sentinel algorithm. Obstacles are drawn in black, and each node's coverage area is shaded with a different color. Note that the utility values shown do not represent the state of any particular node, but are useful as a visualization of the coverage of the entire environment.

Figure 6 shows a simple demonstration of a single node running the Sentinel algorithm in a moderately complex office environment. Each of the six images shows the node position, coverage area, and utility levels at a particular time step while executing the algorithm. The images show the node's coverage over time when read left to right, top to bottom.

We next show a more complex environment where a complete static cover is difficult to achieve without using a very large number of nodes. Figure 7 shows node positions and coverage area of 10, 50, 100, and 200 nodes after convergence of a Voronoi-region based static coverage algorithm.

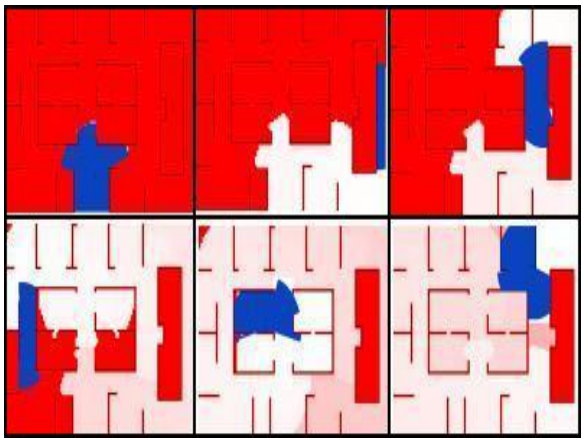

Fig. 6. Sample run of the Sentinel algorithm with a single node.

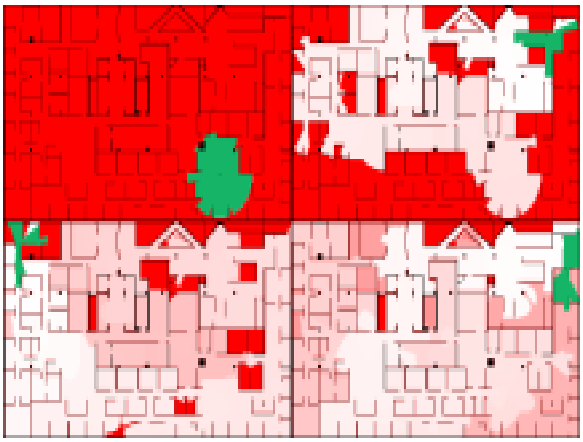

Fig. 8. Single-node Sentinel algorithm, complex environment.

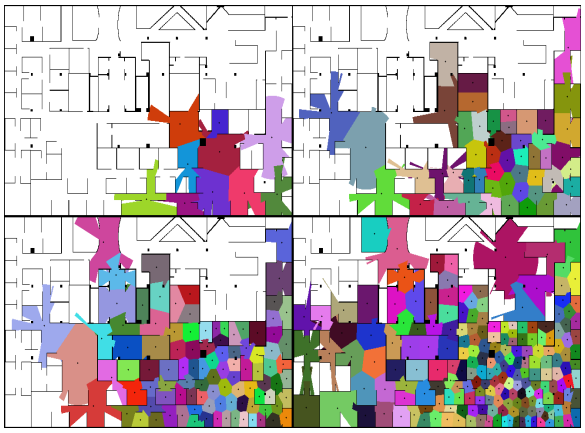

Fig. 7. Placement and coverage of 10, 50, 100, and 200 nodes using a Voronoi-region based static coverage algorithm.

Fig. 9. Sentinel algorithm, 10 nodes, complex environment.

We now show how the Sentinel algorithm handles the same environment. Figures 8 and 9 show the single-node Sentinel algorithm with a single node and the multi-node Sentinel algorithm with 10 nodes, respectively.

Next, we demonstrate switching from dynamic coverage to static coverage. Figure 10 shows the simulation of 25 nodes in a moderately-complex environment. The nodes provide a dynamic cover using the multi-node Sentinel algorithm until the reachable environment has been fully explored and the node graph becomes connected based on the ability of nodes to communicate with each other. The nodes then switch to Voronoi-region based static coverage algorithm, which after convergence is able to almost completely cover the reachable environment.

Finally, we show that the collaboration between nodes when determining paths allows for fairly efficient exploration. Figure 11 shows explored area over time with 1, 3, 5, and 10 nodes in the moderately-complex office environment.

Videos of the above demos are available online at `http://www.pantheon.yale.edu/~jjc72/sentinel/`

V. Future work

Modifying several of the parameters and update functions within the Sentinel algorithm could produce interesting behavior. One such case would be having an alternate method for nodes to update their utility matrices. The current implementation adds 1 to each entry that does not represent a region that is currently covered. Better

Fig. 10. 25 nodes in a moderately-complex environment. The first three images show the nodes running the Sentinel algorithm. Once the environment has been explored, the nodes switch to a Voronoi-region based static coverage algorithm, shown in the bottom three images. Lines between two nodes show when nodes are able to communicate with each other.

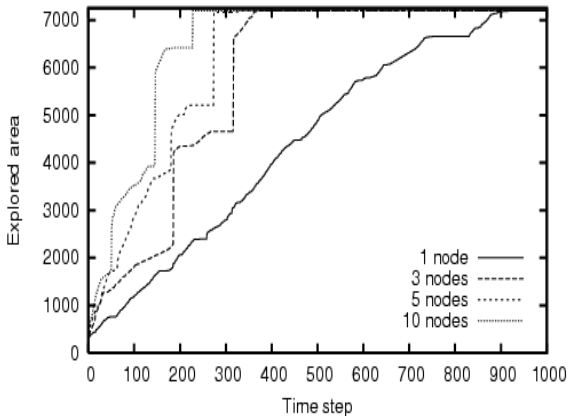

Fig. 11. Total exploration over time with 1, 3, 5, and 10 nodes

coverage could be achieved by instead using an exponential function of the prior utility value. Furthermore, the update function could operate differently on different regions. For example, the update function could be used to indicate that certain regions are especially important by using a greater increment on those regions. Likewise, the behavior of using different path functions could be explored, for example computing the value of an entire path instead of just the value at the end point, as done in [5].

The algorithm presented is a distributed, synchronous algorithm, where computation occurs at fixed time steps. We believe the algorithm could be extended to work with nodes operating asynchronously. The difficulty lies in merging the utility matrices between nodes that are at different time steps.

Another extension of our algorithm would be to handle node failure and unreliable sensor information.

Finally, an analytical analysis of using discrete algorithms and simulation environments in this setting could give an expression for the error generated by the descretizing process. Such an error analysis would be useful in determining an appropriate granularity to divide the sensing environment.

VI. Conclusion

We have discussed different types of coverage in mobile sensor networks, an we introduced the Sentinel algorithm which implements a dynamic cover. The Sentinel algorithm is a distributed algorithm that guarantees every reachable region in an environment will eventually be covered. We described the sentinel algorithm for single nodes and multiple nodes, and gave criteria for switching to a static coverage algorithm when it is appropriate to do so.

Our experiments within our simulation environment have shown the Sentinel algorithm achieves its goals of allowing a limited number of nodes to efficiently explore and provide coverage of an unknown area. We have also shown that collaboration among nodes allows for more efficient exploration as the number of nodes increases.

VII. Acknowledgements

We would like to thank Richard Yang, David Goldenberg, and Stephen Morse for their valuable feedback and advice on our work. We would also like to thank Francesco Bullo for giving an inspiring talk on emergent behavior and coverage.

References

[1] Jorge Cortes, Sonia Martinez, Timur Karatas, Francesco Bullo *"Coverage Control for Mobile Sensor Networks"*, IEEE transactions on Robotics and Automation, June 16, 2003.

[2] Andrew Howard, Maja J Mataric and Gaurav Sukhatme *"Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem"*, In Proceedings of the 6th International Symposium on Distributed Autonomus Robotics Systems, June 2002

[3] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman and Sundeep Pattern *"Distributed Online Localization in Sensor Networks Using a Moving Target"*,
[4] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrum and Hakan Younes *"Coordination for Multi-Robot Exploration and Mapping"*, American Association for Artificial Intelligence, 2000
[5] Wolfram Burgard, Dieter Fox, Mark Moors, Reid Simmons and Sebastian Thrun *"Collaborative Multi-Robot Exploration"*, In Proceedings of the IEEE International Conference on Robotics and Automation, 2000
[6] Adrian Secord, Voronoi Diagrams *"http://stippling.org/npar2002/html/stipples-node2.html"*