Complete and Robust Cooperative Robot Area Coverage with Limited Range

Pooyan Fazli, Alireza Davoodi, Philippe Pasquier, and Alan K. Mackworth

***Abstract*— We address the problem of multi-robot area coverage and present a new approach in the case where the map of the area and its static obstacles are known and the robots have a limited visibility range. The proposed method starts by locating a set of static guards on the map of the target area and then builds a graph called *Reduced-CDT*, a new environment representation method based on the *Constrained Delaunay Triangulation (CDT)*. *Multi-Prim's* is used to decompose the graph into a forest of *partial spanning trees (PSTs)*. Each *PST* is then modified through a mechanism called *Constrained Spanning Tour (CST)* to build a cycle which is then assigned to an individual robot. Subsequently, robots start navigating the cycles and consequently cover the whole area. We show that the proposed approach is complete and robust with respect to robot failure.**

I. INTRODUCTION

Multi-robot area coverage is receiving considerable attention due to its applicability in different scenarios such as search and rescue operations, planetary exploration, intruder detection, environment monitoring, floor cleaning and so on. In this task a team of robots is cooperatively trying to observe or sweep an entire area, possibly containing obstacles, with their sensors or actuators. The goal is to build an efficient path for each robot which jointly ensure that every single point in the environment can be seen or swept by at least one of the robots while performing the task. If there is a need to detect some resources in the environment, area coverage guarantees finding all the resources in the target area.

There is confusion in the literature regarding the terms *Coverage* and *Exploration*. To clarify the problem definition, we note that in exploration, we have an unknown environment and a team of robots is trying to build a map of the area together [22], [4]. In a coverage problem, the map of the environment may be known or unknown and the goal of the team is to jointly observe/sweep the whole area with their sensors or physical actuators. Building a map of the environment is not the ultimate aim of the coverage mission.

Another similar class of problems is *Boundary Coverage* in which, the aim is to inspect all of the obstacles' boundaries by a team of robots instead of complete coverage of the area [21].

Several research communities including robotics/agents [10], [8], [2], sensor networks [3], [14] and computational geometry [5] work on different variations of the area coverage problem. In computational geometry, this problem stems from the *Art Gallery* problem [17] and its variation for mobile guards called the *Watchman Route* problem [7], [18]. In the *Art Gallery* problem, the goal is to find a minimum number of static guards (control points) which jointly can cover a work space under different restrictions. On the other hand, in the *Watchman Route* problem the objective is to compute routes watchmen should take to guard an entire area given only the map of the environment. Most research done on the above problem definitions deal with simple polygons without obstacles, unconstrained guard visibility, and single watchman scenarios.

From a robotics point of view, in a taxonomy presented by Choset [8], the proposed approaches for area coverage are divided into *offline* methods, in which the map of the environment is known, and *online* methods, in which the map of the environment is unknown. Choset [8] further divides the approaches for area coverage based on the methods they employ for decomposing the area: *Exact Cellular Decomposition*, and *Approximate Cellular Decomposition*.

Previous research on multi-robot area coverage is mostly focused on approaches using the *Approximate Cellular Decomposition* (e.g. grid-based methods) [1], [23], [12]. These methods have limitations since they do not consider the structure of the environment and as a result are unable to handle partially occluded cells or cover areas close to the boundaries in continuous spaces. In contrast, methods based on the *Exact Cellular Decomposition* (e.g. graph-based methods) which employ structures such as the *reeb graph* for environment representation do not suffer those restrictions [20]. However, while traversing the graph guarantees covering the whole environment in continuous spaces, it might include many redundant movements.

A. Contributions

This paper addresses the problems mentioned above regarding dealing with the structure of the environments and also robots' limitations in the real world in a coverage scenario. First, the proposed approach guarantees completeness, meaning every accessible point in the environment will be visited in a finite time. This alleviates issues raised by methods based on *Approximate Cellular Decomposition* ignoring partially occluded cells or areas close to the boundaries. Second, it minimizes redundant movements of the robots in the workspace. Third, it supports robustness by handling individual robot failure and finally the proposed approach is designed in a way to overcome the restrictive constraint imposed by the robots' limited visibility range as well.

II. MULTI-ROBOT AREA COVERAGE

We present a cooperative approach to covering a known environment using an arbitrary number of robots. For this

P. Fazli and A. K. Mackworth are with the Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada {pooyanf, mack@cs.ubc.ca}

A. Davoodi and P. Pasquier are with the School of Interacive Arts and Technology, Simon Fraser University, Vancouver, B.C., Canada {ada48, pasquier@sfu.ca}

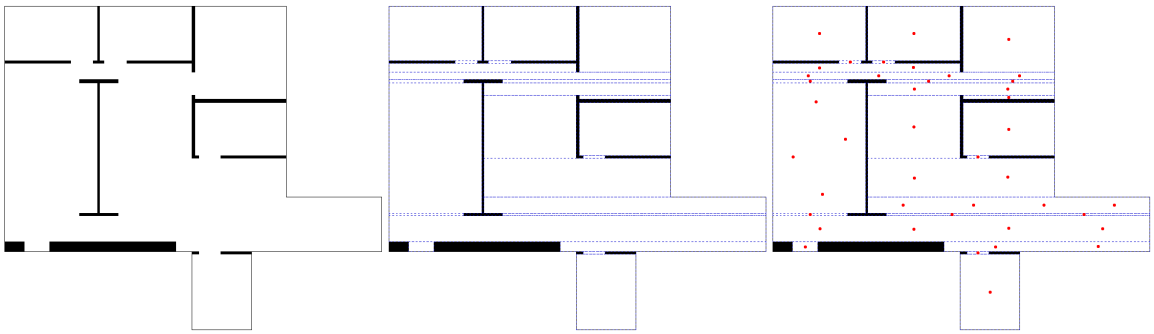

Fig. 1. From Left to Right: (a) Sample Environment (b) Trapezoidal Decomposition (c) Computed Guards

purpose, we make the following assumptions.

Assumption 1. *The environment is a known, 2D, simple polygon containing static polygonal obstacles.*

Assumption 2. *The robots are homogeneous, have the same speed, and can move in all directions.*

Assumption 3. *The robots are assumed to have* 360° *field of view and a predefined circular limit of visibility range.*

Assumption 4. *The vision sensors are ideal without noise.*

Our coverage method is composed of four main steps:

1) First, it determines the locations of static guards required to visually cover a given 2D environment, considering the limited visibility range of the robots' cameras.
2) Then, it builds the *Reduced-CDT*, a graph-based representation of the environment.
3) An algorithm called *Multi-Prim's* is introduced to partition the graph and construct a forest consisting of as many *partial spanning trees* as there are covering robots.
4) Afterward, a new method called *Constrained Spanning Tour* is used to build a cycle on each resultant tree of the forest, and finally, each cycle is allocated to an individual robot in the team.

A. Locating Guards with Limited Visibility

In our problem definition, we assume robots equipped with panoramic cameras with a 360° field of view. However, the cameras' visibility range is limited. The proposed approach initially locates a set of guards required to visually cover an entire area [13]. These static guards are control points that can jointly cover the whole environment while satisfying the visibility constraint of the robots. To this end, the algorithm decomposes the initial target area, a 2D, simple, polygon with static obstacles, into a collection of convex polygons (Trapezoidal Decomposition). Then, a divide and conquer method is applied to successively divide each of the resulting convex polygons into smaller convex sub-polygons until each of them can be visually covered by one guard. Figure 1 illustrates sequential steps of computing the static guards for a sample environment in Player/Stage [11].

B. Environment Representation

In this paper, we investigate a graph structure for environment representation based on the *Constrained Delaunay Triangulation (CDT)*. Given the set of obstacles and their corresponding endpoints, the algorithm first uses the method explained in section II-A to create the set of static guards. The *CDT* is then built on the obstacles and the computed guards.

Then, we introduce a mechanism to reduce the graph so as to minimize the distance each robot has to traverse which consequently improves the efficiency of the whole approach.

1) Constrained Delaunay Triangulation: The *Delaunay Triangulation* of a set of points in the Euclidean plane is a triangulation such that circumcircle of any triangle in the triangulation does not contain vertices other than the three that define it [15]. The *Delaunay Triangulation* corresponds to the dual graph of the *Voronoi tessellation*. Figure 2(a) illustrates the *Delaunay Triangulation* on the sample environment.

The *Constrained Delaunay Triangulation (CDT)* is a variant of the standard *Delaunay Triangulation* in which a set of pre-specified edges (in our case, edges of the obstacles) must lie in the triangulation [6]. A *Constrained Delaunay Triangulation (CDT)* is not truly a *Delaunay Triangulation*. Some of its triangles might not be Delaunay, but they are all constrained Delaunay. Figure 2(b) illustrates the *Constrained Delaunay Triangulation* on the sample environment.

2) Graph Reduction: The aim of graph reduction is to improve efficiency by minimizing the average or total time taken for the robots to traverse the graph. Algorithm 1 describes the steps of the construction of a reduced graph (*Reduced-CDT)* on a given environment. The input of the algorithm is the *CDT* made on the map of the area.

The method starts by using the *Floyd-Warshall* algorithm to find the set $MD = \{(c_{ij}, v_i, v_j) | v_i, v_j \in V_{cdt}\}$ of minimum distances, c_{ij}, and the set $SP = \{(r_{ij}, v_i, v_j) | v_i, v_j \in V_{cdt}\}$ of shortest paths, r_{ij}, between any pair of vertices v_i and v_j of the input graph *(line 2)*.

The minimum value of all the minimum distances in *MD* is then selected provided that both the endpoints of the corresponding shortest path in *SP* belong to the set of static guards, *SG*, computed in section II-A *(line 3)*. The chosen path *(line 4)*, including all its vertices and edges, forms the initial component called *Connected Component*, G_{cc} *(line 5)*.

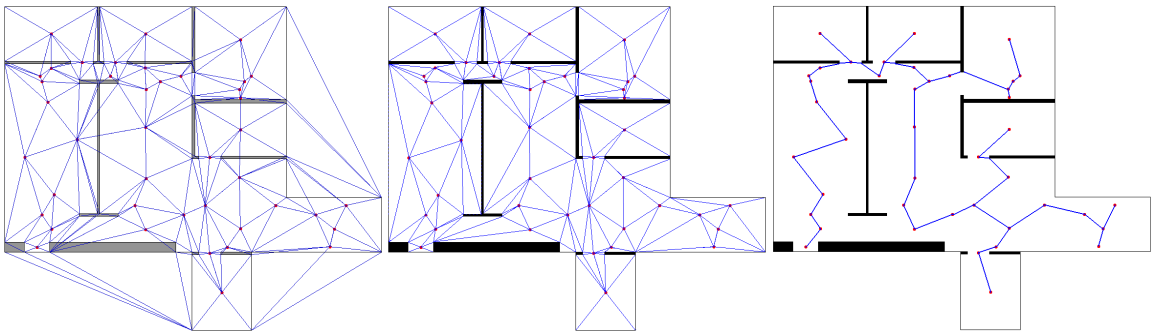

Fig. 2. From Left to Right: (a) Delaunay Triangulation (b) Constrained Delaunay Triangulation (c) Reduced-CDT

Algorithm 1 Graph Reduction

Input:
Graph $G_{cdt}(V_{cdt}, E_{cdt})$, where $V_{cdt} = SG \bigcup P$ // CDT
$SG = \{g_1, g_2, ..., g_m\}$ // Set of Static Guards
$P = \{p_1, p_2, ..., p_n\}$ // Endpoints of Obstacles
Ouput:
Graph $G_{r\_cdt}(V_{r\_cdt}, E_{r\_cdt})$ where $V_{r\_cdt} = SG \bigcup \widetilde{P}$, $\widetilde{P} \subset P$

1: set $V_{r\_cdt} = \phi$ and $E_{r\_cdt} = \phi$
2: $(MD, SP) = FloydWarshall(G_{cdt})$
3: $(i, j) = \arg\min_{(i,j)} \{c_{ij} | (c_{ij}, v_i, v_j) \in MD \text{ and } v_i, v_j \in SG\}$
4: $r_{ij} = GetTheCorrespondingShortestPath(i, j)$
5: $G_{cc}(V_{cc}, E_{cc}) = InitialConnectedComponent(r_{ij})$
6: **while** $\neg$*all the guards added* **do**
7: $g = FindTheClosestGuardTo(G_{cc})$
8: $Expand(G_{cc}, g)$
9: **end while**
10: **return** $G_{r\_cdt}(V_{r\_cdt}, E_{r\_cdt})$, where $V_{r\_cdt} = V_{cc}$ and $E_{r\_cdt} = E_{cc}$

Next, among all the guards that have not yet been added to the component, the algorithm finds the closest guard to the current component *(line 7)*, merging the corresponding shortest path with it *(line 8)*. Following the same process, the algorithm keeps expanding the *Connected Component* until there are no more guards to be added to the current component *(line 6)*. The resultant *Connected Component* is the final reduced graph G_{r-cdt} *(line 10)*.

Figure 2(c) illustrates the *Reduced-CDT* computed on the *CDT* from figure 2(b).

C. Multi-Prim's Algorithm

The *Multi-Prim's* algorithm [9] extends the *Prim's* algorithm [19] used to build the *minimum spanning tree* of a weighted graph. This extension is motivated by the fact that multiple robots are involved in the environment to accomplish the task. The proposed approach has a weighted graph (*i.e. Reduced-CDT*) as an input and outputs a forest of $|R|$ *partial spanning trees*, where $|R|$ is the number of robots in the team. These trees are created incrementally from the initial location of the robots.

The *Multi-Prim's* algorithm starts by determining the starting points. A corresponding starting point for a robot is a visible reduced graph vertex closest to the robot. In some situations, the algorithm might lead to the same starting points for different robots.

Having determined all the starting points, the *Multi-Prim's* algorithm initiates as many trees as there are covering robots. At this stage, each tree only contains two vertices, (*i.e.* the robot and the corresponding starting point), and the edge between them. Subsequently, robots try to sequentially expand their own trees (one guard at a time) until all the guards of the reduced graph are visited at least once. The guards are visited in a way that satisfies the following constraints:

1) Find the nearest immediate (that is, ignoring the endpoints of the obstacles) guard, add it and the corresponding path to the tree provided that it does not create a cycle. In case of a tie, choose the guard which maximizes the sum of the distances from the guards most recently selected by the other robots.
2) Do not add a guard which has already been visited by any other robot, unless there is no other unvisited immediate guard.
3) When all the guards of the graph are visited by at least one robot, remove as many as possible of common vertices shared by the trees of the robots. To this end, by starting from the most recently selected guard, discard the guards and their corresponding paths from the robot's tree if they had been visited sooner by any other robot. Continue this process until it reaches the most recently selected guard which hasn't been visited sooner by other robots.

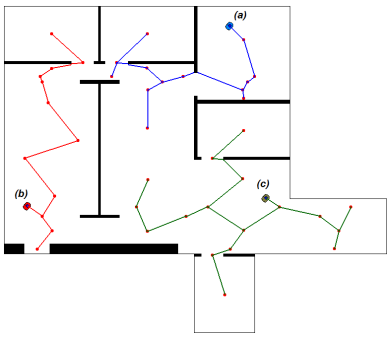

Fig. 3. Multi-Prim's Algorithm

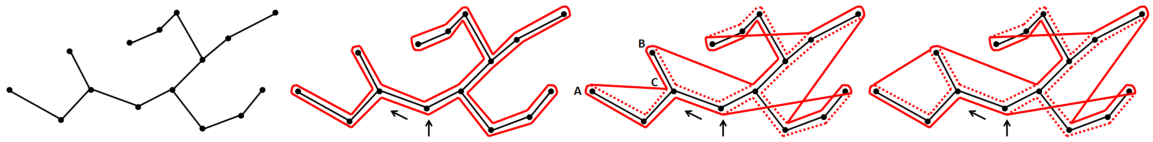

Fig. 4. From Left to Right: (a) A Sample Tree (b) Double-MST (c) Revised-DMST (d) CST

Figure 3 illustrates the result of decomposing the reduced graph among three robots on the sample environment.

D. Constrained Spanning Tour

The next step is to construct a cycle on each *partial spanning tree* resulting from the *Multi-Prim's* algorithm. To this end, we introduce an algorithm called *Constrained Spanning Tour (CST)* which is an improved variant of the *Double-Minimum Spanning Tree (Double-MST)* algorithm. *Double-MST* takes a tree as an input and returns a cycle whose length is twice the length of the tree. In this algorithm, every edge of the tree is visited twice. Figure 4(a) and 4(b) illustrate running the algorithm on a sample tree. A revision can be made to the algorithm in order to form a cycle less than or equal in size to the one generated by the *Double-MST*. Starting from an arbitrary initial point, as indicated by the arrow in Figure 4(c), the revised algorithm called *Revised-DMST* traverses the vertices in the same way as the *Double-MST* algorithm does, but whenever it reaches a vertex visited before, it discards it, proceeding to the next vertex along the cycle to find an unvisited one, making a shortcut edge to it. This process continues until it returns back to the starting point. Figure 4(c) illustrates the result of the algorithm.

However, because of the obstacles in the environment, our coverage mechanism can not apply the *Revised-DMST* algorithm. In order to avoid the obstacles, we introduce another algorithm called *Constrained Spanning Tour (CST)*. This algorithm traverses the vertices of the tree similar to the way the *Revised-DMST* does, with the difference that it uses edges of the original graph (*CDT)* as shortcuts.

The *CST* algorithm uses backtracking to find out the best shortcut. If the shortcut does not belong to the original graph, the next best shortcut will be considered. For instance, in Figure 4(d), suppose there is no edge connecting the vertices *A* and *B* in the original graph. Therefore, the *CST* algorithm can not add the shortcut edge *AB* to the path; instead, it selects the shortcut *AC*, assuming there is such an edge in the *CDT* graph. It keeps following the same way to find the best shortcut. *CST* traverses the tree to return to the initial point. In the worst case the result will be the same as the result of the *Double-MST* algorithm.

Property 1. *Assuming $Len_A(T)$ is the length of the cycle returned by an algorithm A over tree T, we have the following property:*

$$Len_{Revised-DMST}(T) \leq Len_{CST}(T) \leq Len_{Double-MST}(T)$$

Finally, the robots start navigating the cycles which consequently results in full coverage of the target area. *CST* has the benefit that it returns the robots to their initial locations, facilitating tasks like garbage collection and storage.

III. FAULT-TOLERANT MULTI-ROBOT AREA COVERAGE

Robot failure during execution can jeopardize the completion of the area coverage task. By failure, we mean that the robot is not capable of operating and moving anymore, and by fault tolerance, we refer to the ability of the team to respond to individual robot failures that may occur at any time during the mission. Our approach addresses the issue through the concept of *Supportive trees*.

Definition 1. A bridge between two trees of a forest is either a vertex or an edge in common or an edge in the original reduced graph with endpoints each located on one of the trees.

Definition 2. Two trees of a graph are *Mutually Supportive* if there is at least a bridge connecting those two trees.

The algorithm uses the forest, built on the original reduced graph by *Multi-Prim's* algorithm, to find all pairs of *Mutually Supportive* trees.

Lemma 1. *There is at least one Supportive tree for each tree of the forest*

Proof. (Proof by Contradiction) Assume that there is no *supportive* tree for a tree of the forest, meaning that there is no bridge between this tree and any other trees of the forest. This assumption implies that the tree is disconnected within the original reduced graph, which is in contradiction with the graph's connectivity. □

Definition 3. Robots working on two *Mutually Supportive* trees are also *Mutually Supportive*.

Corollary 1. *Each robot has at least one Supportive robot.*

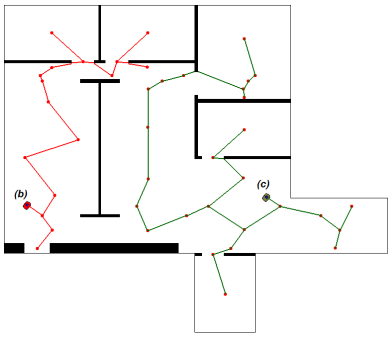

Fig. 5. Fault-Tolerant Multi-Robot Area Coverage

Algorithm 2 Fault-Tolerant Multi-Robot Area Coverage

Input:

A forest of n trees, $T = \{T_1, T_2, \ldots, T_n\}$ where $\bigcup_{i=1}^{n} V_{T_i} = V$ in which V is the set of guards of the reduced graph and V_{T_i} is the set of guards of the tree T_i

$T_F = \{T_{F_1}, T_{F_2}, \ldots, T_{F_m}\}$ // Corresponding Trees of the Failed Robots

Output:

A forest of $n-m$ trees, $T' = \{T'_1, T'_2, \ldots, T'_{n-m}\}$ where $\bigcup_{i=1}^{n-m} V_{T'_i} = V$ in which V is the set of guards of the reduced graph and $V_{T'_i}$ is the set of guards of the tree T'_i

set $ST = \phi$ // Supportive Trees
for all $T_{F_i} \in T_F$ **do**
  for all $T_j \in T$ **do**
    if T_{F_i} *and* T_j *are Mutually Supportive and* $T_j \notin T_F$ **then**
      $ST = ST \bigcup T_j$
    end if
  end for
end for
while $\neg$*all the guards of the graph* G_{r_cdt} *visited* **do**
  for all $T'_i \in ST$ **do**
    find $v \in V$ which is the nearest immediate guard to T'_i in G_{r_cdt} and $\neg$*visited*
    if *There is no such a vertex* v **then**
      find $v \in V$ which is the nearest immediate guard to T'_i in G_{r_cdt}
    end if
    $T'_i.push(v)$
  end for
end while
for all $T'_i \in ST$ **do**
  while $T'_i.top()$ *visited sooner by any other robot* **do**
    $T'_i.pop()$
  end while
end for
return T'

When a robot fails, all the vertices of its assigned tree are released. Then all of its *Supportive* robots expand their trees through the *Multi-Prim's* algorithm to possess the released vertices and to cover the whole environment again.

Theorem 1. *(Robustness) The approach is robust even if* $|R|-1$ *of the robots fail, or in other words, as long as at least one robot is operating correctly (*$|R|$ *is the number of robots).*

Proof. (Proof by Induction) We want to show that the statement is true for all number of robots i from 1 to $|R|-1$, meaning that if i robots fail, the remaining robots will still be able to cover the whole area.

Induction Base: Let $i = 1$. According to Corollary 1, each robot has at least one *Supportive* robot. Therefore, if a robot fails, then all of its supportive robots expand their trees to possess all the released vertices of the failed robot and consequently cover the whole area again.

Induction Step: Assume if $i = k$ robots fail during the mission, the remaining robots can still cover the whole environment. It must then be shown for $i = k+1$ robots.

$|R|-k$ robots can cover the whole environment, meaning that there are $|R|-k$ trees within the environment which by navigating around them *(CSTs)*, the robots can together cover the whole area. According to Lemma 1, there is at least one *Supportive* tree for each tree of the forest. Therefore, if one more robot fails during the mission, its *Supportive* robots expand their trees to include all the released vertices of the failed robot and consequently cover the whole area again. □

Figure 5 illustrates the map of the environment after robot *(a)* fails.

IV. ANALYSIS OF THE ALGORITHM

The variation of the problem with just one robot operating in an environment without obstacles has an exact polynomial time solution. But, extending the problem to support obstacles in the environment or allowing multiple robots make the corresponding decision problems hard. The hardness proofs use simple reductions from the TSP [7] and partition [16] problems, respectively.

A. Overall Complexity

In summary, the complexity order of the different stages of the proposed algorithm is shown in Table I:

TABLE I
COMPLEXITY OF DIFFERENT STAGES OF THE COVERAGE ALGORITHM

Stages of the Algorithm	Time Complexity
Locating Guards	$O(n^2 \log n^2)$
Environment Representation	$O((n+m)^3)$
Multi-Prim's Algorithm	$O(\|R\|(n'+m)log(n'+m))$
Constrained Spanning Tour	$O(\|R\|(n'+m)^2)$

n: Number of vertices of the obstacles
n': Number of vertices of the obstacles in the reduced graph
m: Number of guards
$|R|$: Number of robots

Since the complexity order of the whole coverage mechanism is dominated by the Environment Representation part, the entire approach is a polynomial time algorithm of complexity of $O((n+m)^3)$

B. Other Properties

Using multiple robots may reduce the coverage time by dividing the task among the robots. The coverage time of the area is determined by the robot traversing the longest distance in the environment. We define the *running time* of an area coverage task as the maximum of the distances each robot has to traverse in the area, $\max_{r \in R}(dist(r))$, where $dist(r)$ is the distance traversed by robot r.

Property 2. *(Worst Case Running Time) The worst case running time of the proposed approach is* $2 \times weight(G)$*, where G is the Reduced-CDT graph and* $weight(G)$ *is the sum of the length of all edges in G.*

Proof. In the worst case, the coverage time for a team of $|R|$ robots is equal to that of a single robot. The worst case scenario happens when all the robots start from initial locations very close to each other and the environment (e.g. a narrow corridor) can be represented by a small and sparse graph (e.g. a chain of vertices, resembling a straight line). □

Theorem 2. *(Completeness) The proposed approach covers every accessible point in the environment in a finite time.*

Proof. As mentioned earlier, the computed static guards can together inspect the whole target area considering the limited visibility constraint of the robots' cameras. In the *Multi-Prim's* step, the robots divide the reduced graph created on the static guards among themselves so that the union of the generated trees includes all the guards. Hence, traversing the cycles *(CSTs)* created on the trees assigned to the robots leads to visiting all the static guards in the area and therefore to full inspection of the environment. □

V. CONCLUSION AND FUTURE WORK

This paper investigates the multi-robot area coverage problem and presents a new approach for covering a known polygonal area cluttered with static obstacles. The robots are assumed to have a limited visibility range. The proposed algorithm is guaranteed to be complete and robust provided that at least one robot operates correctly. Also, this approach can be used in other applications such as border inspection instead of area coverage by only considering the convex regions having common edges with the border.

There are numerous challenging future research directions for this problem. Some are as follows:

1) **Heterogeneity:** In a coverage scenario, heterogeneity can be defined in different aspects, such as different movement or sensing capabilities of the robots, and so on.
2) **Open Systems:** The current approach is robust to robot failure during the mission but what if a new robot joins the team in the middle of the execution?
3) **Priority:** In some applications, parts of the target area should be visited or covered sooner than others due to different priorities.
4) **Robustness:** In this paper, we investigate robot failure. There are other robustness criteria that need to be dealt with in the real world, such as robot action failure, communication failure, message loss, and such.
5) **Communication:** The robots could have a limited range of communication, meaning a message sent by a robot is transmitted only to robots within a certain distance from that robot.
6) **Dynamic Environments:** The robot team should have the ability to change its behavior over time in response to a dynamic environment (e.g. dynamic obstacles or an environment changing in shape or size), to either improve performance or to prevent unnecessary degradation in performance.

REFERENCES

[1] N. Agmon, N. Hazon, and G. A. Kaminka, "The giving tree: constructing trees for efficient offline and online multi-robot coverage," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 143–168, 2008.

[2] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping tasks," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2006*, May 2006, pp. 1724–1729.

[3] I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[4] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2000*, vol. 1, 2000, pp. 476–481.

[5] S. Carlsson, B. J. Nilsson, and S. C. Ntafos, "Optimum guard covers and m-watchmen routes for restricted polygons," *International Journal of Computational Geometry and Applications*, vol. 3, no. 1, pp. 85–105, 1993.

[6] L. P. Chew, "Constrained delaunay triangulations," in *Proceedings of the Symposium on Computational Geometry*, 1987, pp. 215–222.

[7] W. Chin and S. Ntafos, "Optimum watchman routes," in *Proceedings of the Second Annual Symposium on Computational Geometry, SCG '86*. New York, NY, USA: ACM, 1986, pp. 24–33.

[8] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[9] A. Davoodi, P. Fazli, P. Pasquier, and A. K. Mackworth, "On multi-robot area coverage," in *Proceedings of the 7th Japan Conference on Computational Geometry and Graphs, JCCGG 2009*, 2009, pp. 75–76.

[10] P. Fazli, A. Davoodi, P. Pasquier, and A. K. Mackworth, "Multi-robot area coverage with limited visibility," in *Proceedings of The 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010*, 2010.

[11] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th International Conference on Advanced Robotics*, 2003, pp. 317–323.

[12] N. Hazon and G. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2005*, April 2005, pp. 735–741.

[13] G. D. Kazazakis and A. A. Argyros, "Fast positioning of limited visibility guards for inspection of 2d workspaces," in *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2002, pp. 2843–2848.

[14] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: maximizing information while minimizing communication cost," in *Proceedings of the 5th International Conference on Information Processing In Sensor Networks, IPSN 2006*, 2006, pp. 2–10.

[15] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a delaunay triangulation," *International Journal of Computer Information Science*, vol. 9, no. 3, pp. 219–242, 1980.

[16] J. Mitchell and E. Wynters, "Watchman routes for multiple guards," in *Proceedings of the 3rd Canadian Conference on Computational Geometry*, 1991, pp. 126–129.

[17] J. O'Rourke, *Art gallery theorems and algorithms*. New York, NY, USA: Oxford University Press, 1987.

[18] E. Packer, "Computing multiple watchman routes," in *7th International Workshop on Experimental Algorithms, WEA*, ser. Lecture Notes in Computer Science, C. C. McGeoch, Ed., vol. 5038. Springer, 2008, pp. 114–128.

[19] A. K. Ravindra, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[20] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: an algorithmic approach," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 109–142, 2008.

[21] K. Williams and J. Burdick, "Multi-robot boundary coverage with plan revision," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2006*, 2006, pp. 1716–1723.

[22] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the Second International Conference on Autonomous Agents, AGENTS 1998*, 1998, pp. 47–53.

[23] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005*, 2005, pp. 3852–3857.