ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1493

Multi-Robot Adversarial Coverage

Roi Yehoshua and **Noa Agmon**[1]

Abstract. This work discusses the problem of adversarial coverage, in which one or more robots are required to visit every point of a given area, which contains threats that might stop the robots. The objective of the robots is to cover the target area as quickly as possible, while maximizing the percentage of covered area before they are stopped. This problem has many real-world applications, from performing coverage missions in hazardous fields such as nuclear power plants, to surveillance of enemy forces in the battlefield and field demining. Previous studies of the problem dealt with single-robot coverage. Using a multi-robot team for the coverage has clear advantages in terms of both coverage time and robustness: even if one robot is totally damaged, others may take over its coverage subtask. Hence, in this paper we describe a multi-robot coverage algorithm for adversarial environments that tries to maximize the percentage of covered area before the team is stopped, while minimizing the coverage time. We analytically show that the algorithm is robust, in that as long as a single robot is able to move, the coverage will be completed. We also establish theoretical bounds on the minimum covered area guaranteed by the algorithm and on the coverage time. Lastly, we evaluate the effectiveness of the algorithm in an extensive set of environments and settings.

1 Introduction

Coverage path planning is one of the fundamental problems in robotics. The goal of coverage path planning is to find a sequence of world locations which allows the robot(s) to visit every part of the target area while optimizing some criteria, usually minimizing travel cost while avoiding obstacles. This problem has many real-world applications, from automatic floor cleaning [2] and coating in supermarkets [4], to field demining [14] and surveillance by unmanned aerial vehicles (UAVs) [7].

In a recently introduced version of the problem, *adversarial coverage* (e.g., [20]), the robot has to cover the given terrain without being stopped by an adversary. Each point in the area is associated with a probability of the robot being stopped at that point. The objective of the robot is to cover the entire target area (including the threat points) as quickly as possible while minimizing the probability that it will be stopped before completing the coverage. This problem is a generalization of the original problem of coverage in neutral environments (without adversarial presence), where risks do not exist, thus are not accounted for, and the only goal is to minimize coverage time [13], [5], [8].

Previous work of adversarial coverage dealt with the single-robot version of the problem. There are obvious advantages in using multiple robots in the adversarial coverage task. Using multiple robots clearly decreases the time to complete the task due to workload division, as seen in the original multi-robot coverage problem [8], [1]. Additionally, using multiple robots improves robustness, as failure of some members of the robot team can be compensated by others. Therefore, in this paper we extend adversarial coverage to multi-robot systems. We focus on coverage using a map of the work-area (known as offline coverage [6]).

First, we formally define the multi-robot version of adversarial coverage, and the problem of finding the safest coverage path for a multi-robot team. In this paper we are mainly concerned about the survivability of the team and not the coverage time. Nevertheless, the algorithm we propose also tries to minimize the coverage time, as long as the robots' safety is not compromised. Clearly, there is a tradeoff between the two objectives of survivability and coverage time: trying to minimize the risk to the robots along their coverage paths typically means making some redundant steps, which in turn can make their coverage paths longer, and thus increase the risks involved, as well as the coverage time.

Second, we describe an efficient distributed multi-robot adversarial coverage algorithm, that tries to maximize the survivability of the team while optimizing the coverage time. The algorithm is based on decomposition of the target area into connected areas of safe and dangerous locations, and prioritization of the coverage such that coverage of safer areas comes before coverage of more dangerous ones. Our method also utilizes graph partitioning techniques in cases where more than one robot is assigned to the coverage of a given area, in order to speed up its coverage. We provide a theoretical bound on the expected coverage percentage guaranteed by the algorithm, and also analyze the best-case and worst-case completion times for the algorithm.

Finally, we evaluate our method in an extensive set of experiments. The results show that adding more robots to the coverage task can significantly increase the percentage of the area covered as well as reduce the coverage time.

2 Related Work

The problem of single and multi-robot coverage has been extensively discussed in the robotic literature (see Galceran and Carreras [6] for a recent survey). Most approaches to multi-robot coverage extend single-robot ideas to multiple robots by using a strategy to divide the workload. Hazon and Kaminka [8] generalized the STC method to multi-robot teams in the family of Multi Robot Spanning Tree Coverage (MSTC) algorithms. Their solution, along with decreasing the total coverage time, achieved robustness in the sense that as long as one robot works properly, the coverage of the terrain is guaranteed. They have also shown that in multi-robot teams redundancy might be necessary for more efficiency. Agmon et al. [1] proposed a spanning

[1] Bar Ilan University, Israel, email: yehoshr1@cs.biu.ac.il, agmon@cs.biu.ac.il

tree construction algorithm that provides efficient paths in terms of distance, and can be used as a basis for MSTC.

Rekleitis et al. [16] presented a collection of algorithms for the coverage planning problem using a team of mobile robots on an unknown environment, based on an exact cellular decomposition. To achieve coverage in line-of-sight-only communications, the robots take two roles: some members, called explorers, cover the boundaries of the current target cell, while the other members, called coverers, perform simple back-and-forth motions to cover the remainder of the cell. For task/cell allocation among the robots, a greedy auction mechanism is used.

Other approaches to multi-robot coverage found in the literature include methods inspired by biological behaviors found in nature. For example, Luo and Yang [12] presented a biologically inspired neural network approach for coverage tasks to multi-robot scenarios where the robots see each other as moving obstacles. Wagner and Bruckstein [17] explored the problem of room cleaning by a group of robots, and proposed a robust algorithm for complete coverage of a terrain. In their case, the robots have limited capabilities and communicate with each other mainly using pheromones.

The offline single-robot adversarial coverage problem was formally defined in [21], in which we proposed a simple heuristic algorithm for generating a coverage path aiming at minimizing a cost composed of both the survivability of the robot and the coverage path length. The heuristic algorithm worked only for obstacle-free areas, and without any guarantees. In a follow-up paper [22] we have addressed a more specific version of the problem, namely, finding the safest coverage path. There we suggested two heuristic algorithms: STAC, a spanning-tree based coverage algorithm, and GAC, which follows a greedy approach. We have shown that while STAC tends to achieve higher expected coverage, GAC produces shorter coverage paths with lower accumulated risk. In [18] we have built a more sophisticated model of the adversary, in which it can choose the best locations of the threat points, such that the probability of stopping the covering robot is maximized. Lastly, the online single-robot version of the problem was presented in [19].

In the related patrol problem ([15], [3]), a multi-robot team needs to patrol around a closed area with the existence of an adversary attempting to penetrate into the area. The patrol problem resembles the coverage problem in the sense that both require the robot or group of robots to visit all points in the given terrain. However, while coverage seeks to minimize the number of visits to each point (ideally, visiting it only once), patrolling seeks to maximize it (while still visiting all points).

3 Problem Formulation

A team of k robots $\mathcal{R} = \{R_1, ..., R_k\}$ needs to cover a given area. The area contains threats that may stop the robots, as well as obstacles. In contrast to obstacles which the robots cannot go through, threat locations are places that the robots must visit, but might be stopped at. The robots are given a map of the environment in advance. We assume that communication between the robots is available without any restrictions. Each robot autonomously covers the area it is assigned, keeping track of all the covered and uncovered space by communicating with the other robots (see section 4.2 for the communication requirements).

Furthermore, we assume that the given area can be decomposed into a regular grid with n cells. Let us denote this grid by G. G contains two types of cells: free cells and cells that are occupied by obstacles. Some of the free cells contain threats. Each free cell i is associated with a threat probability p_i, which measures the likelihood that a threat in that cell will stop a robot visiting it. We define *safe* cells as cells in which the threat probability is $p_i = 0$, while *dangerous* cells are those in which $p_i > 0$. The robots can move in the four basic directions (North, South, East, West), and can locate themselves within the work-area to a specific cell.

Figure 1 shows an example world map of size 20×20, with 8 robots located at the upper-left corner of the area. Obstacles are represented by black cells, safe cells are colored white, and dangerous cells are represented by 5 different shades of magenta. Darker shades represent higher values of p_i (more dangerous areas).

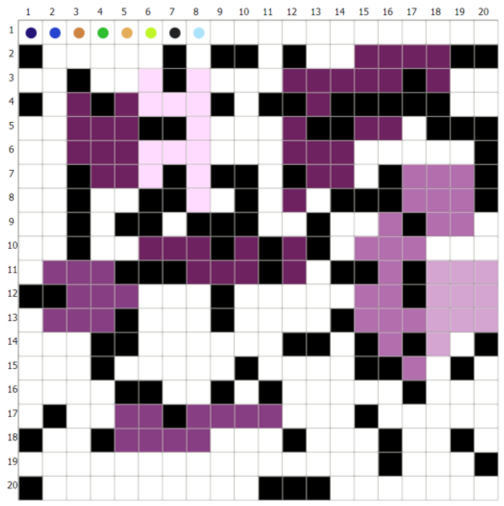

Figure 1. A sample map with 8 robots located at the upper-left corner of the environment. Darker shades represent more dangerous areas.

We assume that robots that have been stopped by threats do not block live robots, i.e. other robots can move through their cells, and that the threats remain there. An example for a real-world scenario in which this assumption holds is when the covering robots are UAVs, and the threats are constant in time (e.g., an anti-drone weapon aimed at a particular location).

The survivability measure can be defined in two levels: the survivability of each single robot, and the survivability of the team. We say that a robot survives the coverage, if it manages to finish its coverage task unharmed. To formally define the survivability of a robot R_i, we first denote the coverage path that it follows by $P_i = (c_1^i, c_2^i, ..., c_{n_i}^i)$, where c_j^i is the cell that robot i is located at in time step j. Thus, the probability that a robot R_i survives the coverage is:

$$Surv(R_i) = \prod_{j=1}^{n_i} (1 - p_j^i) \tag{1}$$

where p_j^i denotes the probability that the robot is stopped by a threat in cell c_j^i.

We say that a team of robots survives the coverage, if at least one of the robots in the team survives until all cells in the area are visited (coverage completed). Thus, if it takes t time steps to cover the area, the probability that a team of k robots $\mathcal{R} = \{R_1, ..., R_k\}$ is able to cover this area is:

$$Surv(\mathcal{R}) = \prod_{j=1}^{t} \left[1 - \prod_{i=1}^{k} p_j^i\right] \tag{2}$$

Following these definitions, we can now define the Multi-Robot Safe Adversarial Coverage Problem (MRSACP) as follows:

Definition 1 *Multi-Robot Safe Adversarial Coverage Problem (MRSACP): Given a team of k robots and a grid representation of a world that contains obstacles and threat points, find a coverage path of the grid that maximizes the survivability of the team.*

The $\mathcal{NP}$-hardness of MRSACP follows directly from the $\mathcal{NP}$-hardness of the single-robot safest coverage path problem [22].

4 Multi-Robot Adversarial Coverage Algorithm

The Multi-Robot Adversarial Coverage algorithm (MRAC, described in Algorithm 1) uses a layered-based approach. It first tries to cover all the safe cells in the target area as efficiently as possible, using the given k robots. Then it covers the dangerous areas from the least dangerous ones to the most dangerous ones. This way the algorithm tries to maximize the coverage percentage before the entire group of robots is stopped.

Before describing the algorithm in detail, let us introduce the following definitions:

Definition 2 *A **connected area** is a connected subset of cells in the grid that belong to the same threat level.*

We also use the terms *safe areas* and *dangerous areas* to refer to connected areas that are composed of only safe cells or only dangerous cells. Figure 2 shows an example of a grid containing two safe areas and five dangerous areas that belong to two different threat levels. The two low-threat-level areas are outlined by yellow lines and the three high-threat-level areas are outlined by blue lines.

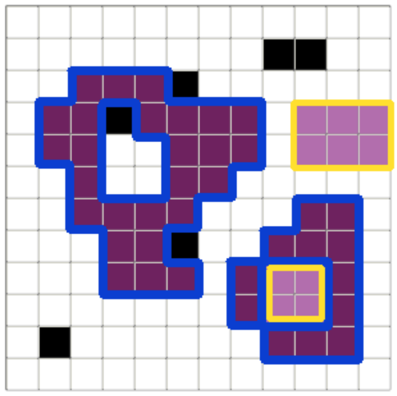

Figure 2. An example of a grid containing two safe areas and five dangerous areas that belong to two different threat levels.

In addition, we denote the distinct threat probabilities that exist in the grid by $p_0, ..., p_l (p_0 = 0)$ (l is finite, since the number of cells in the grid is finite), and define *threat level* i as the group of all the cells that contain threats with probability p_i of stopping the robot.

The main steps of the multi-robot coverage algorithm are:

1. Split the target area into layers according to the threat levels, i.e. layer i contains all the cells that belong to threat level i.
2. For each threat level i, denote by $A_1^i, A_2^i, ..., A_{n_i}^i$ the connected areas that belong to level i.
3. Each robot is assigned to a safe area $A \in \{A_1^0, ..., A_{n_0}^0\}$, which has the safest path from its current location. Areas with more than one robot assigned to them are split between their assigned robots.
4. For the coverage of each area, we use a modified version of GAC (Greedy Adversarial Coverage), the state-of-the-art solution to the single-robot adversarial coverage problem [20].
5. When a robot R_i completes its current coverage task, it is allocated an uncovered area from the safest threat level which has not been completed yet. If more than one such area exists, the area with the minimum-risk path from the robot's current location is chosen. If all areas have been allocated, then the robot is added to an area that is already being covered by another robot. This area is split into two, where each robot is assigned to the sub-area with the safest path from its current location.
6. If a robot is hit by a threat during the coverage of its allocated area, then this area is returned to the pool of unassigned areas.
7. If all the robots have completed their area coverage, and there are no uncovered cells, then the robots declare the environment covered.

The algorithm consists of two main phases: the first phase (steps 1–3) is executed prior to the coverage and computes the initial allocations of areas to the robots. This computation can be performed either offline or online by one of the robots, and then its results can be transmitted to all the other robots in the team. The second phase (steps 4–7) is executed independently by each of the robots in a distributed manner during the coverage process itself.

4.1 Multi-level Graph Partitioning

There are several places in the algorithm where we need to use a graph partitioning scheme, in order to divide a given area between several robots. More specifically, we need to apply graph partitioning in the initial allocation of areas to robots, and at the end of the coverage when all the areas have been allocated and there are some idle robots that can help their teammates finish their coverage task.

The problem of partitioning a graph into k equally-sized components is known to be $\mathcal{NP}$-Hard [10]. Therefore, practical solutions are based on heuristics. Here we use a multi-level graph partitioning algorithm [9], that consists of three main phases: coarsening, partitioning, and uncoarsening (Figure 3). In the coarsening phase, a sequence of smaller graphs, each with fewer vertices is obtained by collapsing vertices and edges into single vertices of the next level, which are called multi-nodes. Then, in the partitioning phase, the coarse graph obtained is partitioned. Lastly, in the uncoarsening phase, the partitioning is refined while the original graph is restored.

4.2 Data Structures

The algorithm maintains a list of connected areas, denoted by $\mathcal{A}$. Each area can be in one of the following states:

1. $S_{unassigned}$ - unassigned to any robot
2. $S_{assigned}$ - assigned to a robot and not completely covered yet
3. $S_{covered}$ - covered

An area that has been assigned to a robot can change its state to either being covered (if the robot has successfully finished covering it) or to unassigned (if the robot has been stopped by a threat during its coverage). An area that has been completely covered cannot change its state.

For each area A, we maintain the following fields:

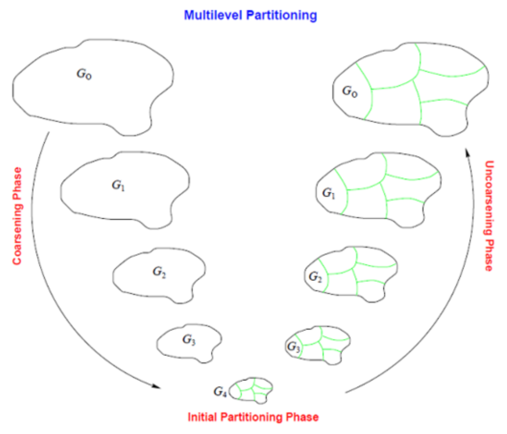

Figure 3. The three phases of multilevel k-way graph partitioning. [9]

- $A.level$ - the threat level this area belongs to
- $A.cells$ - the group of cells that belong to this area
- $A.state$ - the current state of the area
- $A.initial\_robots$ - the group of robots initially assigned to this area. If the group contains more than one robot, then this area would need to split between them.
- $A.robot$ - the robot currently assigned to this area

Each robot in the team can be in one of the following states:

1. S_{idle} - waiting for a task assignment. This is the initial state of each robot.
2. $S_{traveling}$ - traveling to the next area that the robot has to cover.
3. $S_{covering}$ - in the process of covering a given area.
4. S_{done} - the robot has finished covering its allocated area and there are no more areas to be covered.
5. S_{dead} - the robot has been hit by a threat.

The transitions between the robot's states are described in section 4.4. For each robot R, we maintain the following fields:

- $R.state$ - the current state of the robot
- $R.area$ - the current area the robot is assigned to
- $R.location$ - the robot's location
- $R.path$ - the path the robot is currently following. This path can be either the transition path to the robot's next allocated area or the coverage path of the area it is currently covering.

In addition to the global map, we assume that the following data structures are shared (and synchronized) between the robots:

1. The status of each cell in the map (visited or not).
2. The list of connected areas and their states.
3. The states of all the robots, and the areas that they are assigned to.

4.3 Initial Allocation

Algorithm 1 is run prior to the coverage itself. It is responsible for pre-processing of the map and the initial allocation of coverage tasks to the robots.

The procedure Allocate_Areas_To_Robots allocates the initial areas to the robots. The idea is to initially cover all the safe areas as fast as possible, using all the robots available, before moving to the dangerous areas. The procedure first computes the safest paths of each robot to every safe area. Then, it assigns each robot to the safe area with the safest path from its current location, if this area is not too dense with robots. We define a *dense* area as an area whose number of cells is less than d times the number of robots assigned to it, i.e., each robot has less than d cells to cover on average (in experiments we have found that $d = 4$ gave the best results). If the area chosen for the robot is already too dense with robots, then the next safest non-dense area will be allocated to it. After the allocation of areas for all the robots, every area that is assigned to more than one robot is split between the robots using the graph partitioning method described in section 4.1.

Algorithm 1 Multi_Robot_Adversarial_Coverage(G, $\mathcal{R}$, d)

```
input: $G$ - a grid representing the target area, $\mathcal{R}$ - group of $k$ robots, $d$ - maximal area density
output: $\mathcal{A}$ - list of connected areas
1: $\mathcal{A} \leftarrow \emptyset$
2: Group the cells in $G$ into $l+1$ threat levels, $T_0, ..., T_l$
3: for each threat level $i$, $0 \le i \le l$ do
4:     Build the graph $H_i$ induced from the cells in $T_i$
5:     Find the connected components (areas) of $H_i$ using DFS
6:     Let $A_1, ..., A_k$ be the connected areas of $H_i$
7:     for each area $A_j$, $1 \le j \le k$ do
8:         $A_j.state \leftarrow S_{unassigned}$
9:         $A_j.level \leftarrow i$
10:        Add $A_j$ to $\mathcal{A}$
11: Allocate_Areas_To_Robots($\mathcal{A}$, $\mathcal{R}$, $d$)
```

```
1: procedure ALLOCATE_AREAS_TO_ROBOTS($\mathcal{A}$, $\mathcal{R}$, $d$)
   input: $\mathcal{A}$ - list of connected areas, $\mathcal{R}$ - the group of robots, $d$ - maximal area density
2:     $\mathcal{S} \leftarrow \{A | A \in \mathcal{A} \wedge A.level = 0\}$        ▷ the safe areas
3:     for each robot $R \in \mathcal{R}$ do
4:         {Find safest paths to all safe areas}
5:         for each area $A \in \mathcal{S}$ do
6:             $P_A \leftarrow$ Find_Safest_Path_To_Area($A$, $R.location$)
7:         {Find safest non-dense area}
8:         Let $A_1, ..., A_k$ be the areas sorted by $P_A.cost$
9:         $i \leftarrow 1$
10:        while $i \le k$ and $R.area = null$ do
11:            if $|A.initial\_robots| \cdot d \le |A.cells|$ then ▷ check if area is not too dense with robots
12:                Add $R$ to $A.initial\_robots$
13:            else
14:                $i \leftarrow i + 1$
15:        {Split the areas that are allocated to multiple robots}
16:        for each area $A \in \mathcal{S}$ do
17:            if $|A.initial\_robots| > 1$ then
18:                Split_Area_Between_Robots($A$)
19:            else if $|A.initial\_robots| = 1$ then
20:                $R \leftarrow A.initial\_robots[0]$
21:                $R.path \leftarrow P_A$
22:                Assign_Area_To_Robot($A$, $R$)
```

The procedure Find_Safest_Path_To_Area [2] searches for the safest path from the current location of the robot to any of the cells that belong to the given area. For that purpose, it runs Dijkstra's shortest paths algorithm on the graph induced from the grid's cells, using the following edge weights:

$$w_{ij} = \begin{cases} p_j/p_{min} & \text{if cell } j \text{ contains a threat} \\ 1/n & \text{otherwise} \end{cases} \qquad (3)$$

This weight function ensures that $w_{ij} \ge 1$ for edges that target a dangerous cell, while $w_{ij} = 1/n$ (where n is the grid size) for edges

[2] The pseudocode of some of the procedures is omitted due to space constraints. The full pseudocode of these procedures can be found at http://goo.gl/mfYn2Y

that target a safe cell. This way, the cost of visiting one dangerous cell is greater than the cost of visiting all the safe cells in the grid. Thus, only when there are two equally safe paths, the robot will prefer the shorter one.

The procedure Assign_Area_To_Robot updates the data structures to indicate that an area A has been assigned to robot R.

The procedure Split_Area_Between_Robots splits the given area into k connected sub-areas using the multi-level graph partitioning algorithm. Then it assigns each robot to the sub-area with the safest path from its current location. Since each sub-area can be allocated to only one robot, we use the Hungarian algorithm [11] for deciding the optimal assignment.

```
1: procedure SPLIT_AREA_BETWEEN_ROBOTS($A, \mathcal{R}$)
   input: $A$ - a connected area, $\mathcal{R}$ - the group of robots globals: $\mathcal{A}$ - list of connected areas
2:     Build the graph $G$ induced from the area $A$'s cells
3:     $k \leftarrow |\mathcal{R}|$
4:     Partition_Graph($G$, $k$)
5:     Let $G_1, G_2, ...G_k$ be the subgraphs created from the partition
6:     for $i \leftarrow 1$ to $k$ do
7:         Create a subarea $A_i$
8:         $A_i.cells \leftarrow$ the nodes in $G_i$
9:         $A_i.level \leftarrow A.level$
10:        Add $A_i$ to $\mathcal{A}$
11:        {Compute the safest path of each robot to the sub-area}
12:        for $j \leftarrow 1$ to $k$ do
13:            $P_{ij} \leftarrow$ Find_Safest_Path_To_Area($A_i, R_j.location$)
14:            $C_{ij} \leftarrow P_{ij}.cost$                    ▷ the cost matrix
15:    $O \leftarrow$ Hungarian_Method($C$)      ▷ $O$ is the optimal assignment of robots to sub-areas
16:    $O_i \leftarrow$ optimal assignment of robot $i$
17:    for $i \leftarrow 1$ to $k$ do
18:        $R_i.path \leftarrow P_{O_i,i}$
19:        Assign_Area_To_Robot($O_i$, $R_i$)
20:    Remove $A$ from $\mathcal{A}$
```

4.4 The Coverage Algorithm

We now describe the algorithm that is executed by each robot independently, after the initial allocations have been performed. Algorithm 2 describes the action taken by each robot during one time step.

The robot typically starts in the state $S_{traveling}$, unless there were not enough areas to allocate to all the robots in the initial phase (e.g., there was only one area with 20 cells and there are 10 robots). In this case, the robot starts in S_{idle} and waits for a task assignment. When the robot arrives at the area that it needs to cover, its state changes to $S_{covering}$. If along the way to its designated area, the robot completes a coverage of another area (e.g., an area that consists of only one cell that resides on the connecting path between two larger areas), then this area's state is changed to $S_{completed}$, and is removed from the pool of available areas for coverage. Only when the robot arrives at its designated area, the coverage path of this area is computed. This is because by the time the robot gets to this area, some of its cells may have already been visited along the connecting paths of other robots.

When the robot finishes its coverage task, it is assigned a new area to cover from the pool of unassigned areas and its state changes back to $S_{traveling}$. If there are no more unassigned areas, it joins another covering robot to help it finish its coverage task. If there are no more areas that can be shared, then the robot's state changes to S_{done} and it waits until one of the areas becomes unassigned (e.g., when another robot is stopped). If the robot is stopped during the coverage of its assigned area or on its way to it, the robot's state changes to S_{dead} and its allocated area returns to the pool of unassigned areas.

Algorithm 2 Robot_Action(R)

```
1: switch $R.state$ do
2:     case $S_{traveling}$
3:         $c \leftarrow$ next cell on $R.path$
4:         Mark $c$ as visited
5:         $A \leftarrow$ the area that contains $c$
6:         if all cells in $A$ are visited then
7:             $A.state \leftarrow S_{completed}$
8:         if robot was hit by a threat in $c$ then
9:             $R.state \leftarrow S_{dead}$
10:            Reallocate_Area($R.area$)
11:        else if $c$ is the last cell on $R.path$ then
12:            $R.path \leftarrow$ Area_Coverage($R.area, R.location$)
13:            $R.state \leftarrow S_{covering}$
14:    case $S_{covering}$
15:        $c \leftarrow$ next cell on $R.path$
16:        Mark $c$ as visited
17:        if robot was hit by a threat in $c$ then
18:            $R.state \leftarrow S_{dead}$
19:            Reallocate_Area($R.area$)
20:        else if $c$ is the last cell on $R.path$ then
21:            $R.area.state \leftarrow S_{completed}$
22:            Allocate_Next_Area($R$)
23:    case $S_{done}, S_{idle}$
24:        if an unassigned area exists in $\mathcal{A}$ then
25:            Allocate_Next_Area($R$)
26:    case $S_{dead}$
27:        $R.area.state \leftarrow S_{unassigned}$
```

The procedure Allocate_Next_Area allocates a new area to cover for a robot that has completed its coverage task. If there are any unassigned areas, the robot is assigned to the safest unassigned area with the safest path from its current location. If all the areas are already assigned, the robot joins another covering robot to help it finish its coverage task.

The procedure Find_Area_To_Share tries to find for a given idle robot an assigned area that it can help finish covering. If the path from the given robot's location to the designated area is longer than the number of unvisited cells in that area, then there is no point of sending the robot there, since by the time the robot arrives there, its coverage will have been completed. If an area that can be shared has been found, its connected unvisited parts are defined as new areas and added to the pool of unassigned areas instead of the original area. If there is only one such part (e.g., its assigned robot has not started covering it), then it is split into two balanced parts. Finally, both the assigned robot and the idle robot are (re-)assigned to the sub-areas with the safest paths from their current location.

The procedure Assign_Robot_To_Safest_Area finds the safest path from the robot's location to each of the given areas and assigns the robot to the area with the safest path.

For the coverage of each area, our algorithm is based on the Greedy Adversarial Coverage (GAC), the state-of-the-art solution to the single-robot adversarial coverage problem described in [20]. GAC follows a greedy approach, where in each step it leads the robot to the safest nearest cell to its current location which has not been covered yet.

We have modified the original coverage algorithm to take into account cells that have already been visited in the target area (see Algorithm 3). By the time the designated area is allocated to the robot and the robot reaches this area, other robots may already have visited some cells in this area on the way to their own designated areas. Thus, in order to avoid repeated coverage of these cells, we have changed the algorithm to cover only unvisited cells in the given area. When transitioning between unvisited cells in the target area, the robot is allowed to visit cells that belong to other areas (if they make the connecting path safer).

Algorithm 3 Area_Coverage(A, s)

```
input: an area $A$, and a starting cell $s$
output: a coverage path $P$ that covers all unvisited cells in $A$
1: $P \leftarrow \emptyset$
2: Build the graph $H$ induced from $A$'s cells with the weight function $w$ from eq. (3)
3: Add the starting cell $s$ to $P$
4: Mark $s$ as visited
5: while there are unvisited cells in $A$ do
6:     Run Dijkstra's shortest paths algorithm on $H$ from $s$
7:     $v \leftarrow$ an unvisited node in $A$ with minimum weighted distance from $s$
8:     Add the path $s \rightsquigarrow v$ to $P$
9:     Mark $v$ as visited
10:    $s \leftarrow v$
11: return $P$
```

Finally, the procedure Reallocate_Area is used to reallocate an area whose coverage was stopped in the middle, because its assigned robot was hit by a threat. The procedure finds all the unvisited parts of the given area and creates new unassigned areas from them. Then these areas are added to the list of connected areas instead of the given area. The next idle robot will be assigned to one of these sub-areas in its next cycle (lines 24–25 in algorithm 2).

5 Analysis of the MRAC algorithm

We now analyze the MRAC algorithm. We first prove that the algorithm is complete, i.e., that it generates coverage paths that together cover all the accessible cells in the given area.

Theorem 1 *(Completeness) Algorithm MRAC generates paths for the robots that together cover every cell accessible from their starting locations.*

Proof. MRAC partitions the target area into k connected areas, whose union is equal to the target area. Each of these areas is eventually assigned to one of the robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). The areas are covered by using the GAC algorithm. Previous work has shown that GAC is complete, i.e. that it produces a path that covers all the accessible cells in its given area (Theorem 8 in [20]). Thus, the union of the coverage paths of the k connected areas covers every accessible cell in the target area. □

As key motivation for using multiple robots comes from robustness concerns, we now prove that MRAC is robust to robotic failures.

Theorem 2 *(Robustness) Algorithm MRAC guarantees that the coverage will be completed as long as at least one robot remains active.*

Proof. The target area is divided into a set of connected areas. Each area is eventually assigned to one of the robots. If a robot is stopped while covering its assigned area, the uncovered parts of this area are returned to the pool of unassigned areas (in procedure Reallocate_Area). These sub-areas are eventually assigned to one of the remaining robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). □

We now provide a bound on the minimum expected number of cells that the robots will cover when following the policy generated by the MRAC algorithm. We will use the definition of expected coverage from [20]. Given a coverage path A of a robot R, we denote the sequence of new cells discovered along A by $(b_1, ..., b_n)$, and the number of new cells visited by the robot until it is stopped by C_A. Furthermore, for each cell in the sequence b_i, we will denote the sub-path in A that leads from the origin cell a_1 to it by g_i. Then, under the threat probability function p, the expected number of new cells that robot R visits can be expressed as:

$$E(R) = \sum_{i \in (b_1,...,b_n)} \prod_{j \in g_i} (1 - p_j) \tag{4}$$

Theorem 7 in [20] provides a bound on the minimum expected number of cells that a single robot will be able to cover, given its coverage path A and the threat probability function p.

Let l be the number of dangerous threat levels and the threat probabilities of these levels be $p_0, ..., p_l$ $(p_0 = 0)$. Let $A_{i,1}, ..., A_{i,k_i}$ be the connected areas of threat level i, arranged in the order of their visit by the robot R. Let $|A_{i,j}|$ be the size of area $A_{i,j}$ and $m_{i,j}$ the number of cell visits needed to cover this area $(m_{i,j} \geq |A_{i,j}|)$. Let $C_{i,j}$ be the set of cells on the connecting path between two consecutive areas $A_{i,j}$ and $A_{i,j+1}$, and C_{i,k_i} be the set of cells on the connecting path between the last area of threat level i and the first area of threat level $i + 1$. Denote by $P(C_{i,j})$ the probability to traverse the cells in $C_{i,j}$ without being hit by a threat. Then the expected number of cells robot R will be able to cover before it is stopped is at least:

$$E(R) \geq |A_{0,0}| + \sum_{i=0}^{l} \sum_{j=1}^{k_i} \left[\prod_{x=0}^{i-1} \left[(1-p_x)^{\sum_{y=1}^{k_x} m_{x,y}} \prod_{y=1}^{k_x} P(C_{x,y}) \right] \cdot (1-p_i)^{\sum_{y=1}^{j} m_{i,y}} \prod_{y=1}^{j-1} P(C_{i,y}) \right] |A_{i,j}| \tag{5}$$

We now use this theorem to provide a lower bound on the expected coverage that can be attained by the entire robotic team.

Theorem 3 *Let $\mathcal{R}$ be a group of k robots $\mathcal{R} = \{R_1, ..., R_k\}$. Let l be the number of dangerous threat levels. Let $\mathcal{A}$ be the group of connected areas and let C be the set of cells on the connecting path between two areas. Then the expected number of cells the team $\mathcal{R}$ will be able to cover before all robots in $\mathcal{R}$ are stopped is at least:*

$$E(\mathcal{R}) \geq \sum_{i=1}^{k} E(R_i) - |C| \tag{6}$$

Proof. Since different robots in the team cover different areas (when one area is assigned to more than one robot, it is split between them), the expected number of cells that will be covered by the entire team is equal to the total number of cells that will be covered by each robot individually minus the number of cells on the connecting paths between areas, since those might be visited multiple times by different robots. □

Notice that the ideal expected coverage is attained when the robots visit the cells precisely in increasing order of their threat probabilities, i.e., when they first visit all the safe cells, then all the cells with threat probability p_1, etc. Thus, if we denote the cells that belong to threat level i by $c_{i,1}, ..., c_{i,n_i}$, then the ideal expected coverage is:

$$\sum_{i=0}^{l}\sum_{j=1}^{n_i}\left[\prod_{x=0}^{i-1}(1-p_x)^{n_x}\right]\cdot(1-p_i)^{j-1} \qquad (7)$$

In most environments the ideal expected coverage cannot be attained, since typically some of the cells that belong to a given threat level are disconnected, and the robots have to visit cells that belong to a higher threat level or revisit threat points that have already been covered in order to move between the disconnected cells. In these cases, the cell visits cannot precisely follow the order of the threat levels.

By comparing equations (6) and (7), we can see that the gap between the ideal expected coverage and the expected coverage achieved by MRAC depends on the quality of the coverage algorithm of each connected area (which determines the gap between n_x and $\sum_{y=1}^{k_x} m_{x,y}$). Finding an optimal solution to the coverage problem is $\mathcal{NP}$-Hard [20]. However, MRAC uses for the coverage of each area the algorithm GAC, which creates a close-to-optimal coverage paths [20]. In fact, MRAC can utilize any single-robot coverage algorithm for the internal coverage of each area.

The next theorem provides a bound on the coverage time that can be attained by the robots following the policy generated by MRAC.

Theorem 4 *The worst-case coverage time for k robots is equal to that of a single robot. The best-case coverage time for k robots is approximately $1/k$ the coverage time of a single robot.*

Proof. The worst-case scenario is where all the robots start at locations that are all inside dangerous areas, and thus they will be stopped by threats in the beginning of their coverage paths. In such case, using more than one robot will not improve the total coverage time. The best-case scenario is when the target area consists of only one contiguous safe area (or a few large safe areas with connecting paths that have very low risk). In such scenario, the area is split into k almost equally-sized components that are covered concurrently by the k different robots. In such scenario, the coverage time of MRAC highly depends on the quality of the graph partitioning algorithm, which has no theoretical guarantee, however, in practice, it almost always generated equally-sized sub-graphs. The coverage time in this case also depends on the initial locations of the robots, e.g., if the robots start the coverage at the same location, it will take them some time to move to their assigned areas. □

6 Experimental Results

We have fully implemented the MRAC algorithm and evaluated it in various types of simulated environments with varying parameters, such as map size, number of robots, number of threat levels, distribution of threats, number of obstacles, number of threats, etc. We provide here the most interesting results from our experiments.

First, we demonstrate the algorithm's results on a specific grid map of size 20×20 (Figure 4). In this example, the maps contained 10 contiguous threat areas which were divided into 5 different threat levels with the following threat probabilities: $4\%, 8\%, 12\%, 16\%$, and 20%. We ran the algorithm on this map with two team sizes of 4 robots and 10 robots. All the robots start in the upper-left corner of the area (as shown in Figure 1). The figure shows the locations of the robots after the coverage has finished, and the total number of times each cell in the grid has been visited by the robots. Using 10 robots instead of 4 robots has increased the coverage percentage from 80.33% to 97.33% and reduced the coverage time from 311 to 244 time steps. As can be seen in both maps, more dangerous areas are less frequently visited by the robots.

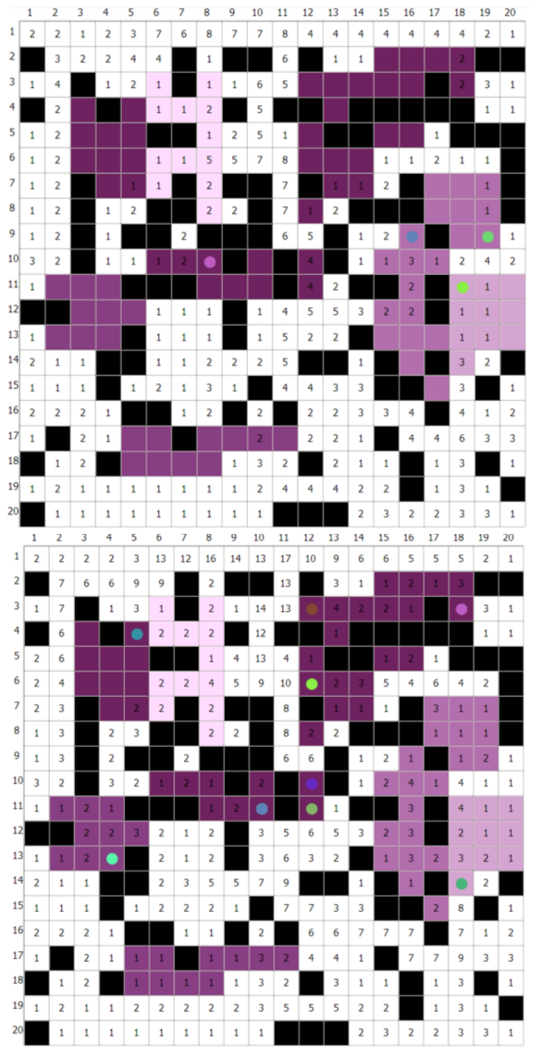

Figure 4. A sample map with the results of MRAC with 4 robots (upper figure) and 10 robots (lower figure). The number in each cell indicates the number of visits in that cell.

Second, we examined the effect of changing the team size on the performance of the algorithm. We used 500 randomly-generated maps of size 20×20, with varying number of robots between 1 and 20. In all the experiments, the obstacles ratio and the threats ratio were both set to 25% of the cells. The maps contain 10 contiguous threat areas which are divided into 5 different threat levels with the following threat probabilities: $4\%, 8\%, 12\%, 16\%$, and 20%. The locations of the obstacles were randomly chosen. Figure 5 shows the percentage of covered area and the coverage time for each team size.

As can be clearly seen in the graph, the coverage percentage increases and the coverage time decreases as we add more robots to the team. Although the area that the robots are able to cover increases as the team grows, they are able to cover it more quickly. The small increase in the coverage time in the transition from one robot to two robots is due to the fact that the area that two robots are able to cover is significantly larger than one robot (62% for two robots vs. 44% for

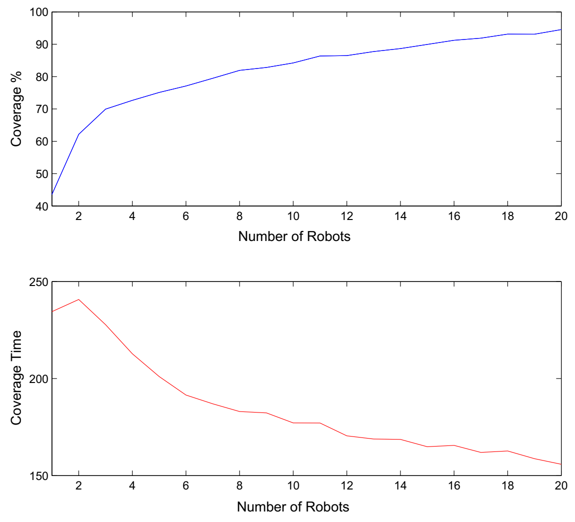

Figure 5. Comparing coverage percentage and coverage time for 1 to 20 robots in environments with contiguous areas of threats.

one robot), thus it takes more time until the two robots are stopped despite the division of the workload between them (e.g., one robot may be stopped much earlier than the second one).

Next, we examined environments where the threats are randomly scattered across the map. We kept all the other map settings as in the previous experiment (i.e., the same ratio of obstacles and threats, and the same number of threat levels). Figure 6 shows the results.

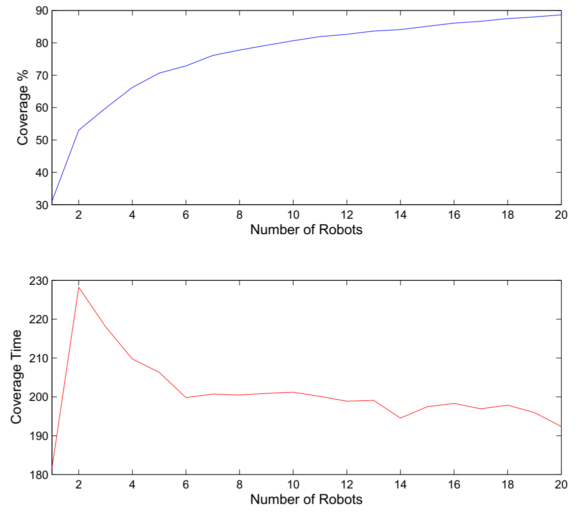

Figure 6. Comparing coverage percentage and coverage time for 1 to 20 robots in environments with randomly-scattered threats.

As previously, the coverage percentage increases and the coverage time decreases as we add more robots to the team. However, the coverage percentage that the team is able to achieve in this type of environments is smaller than in environments with contiguous dangerous areas, and it takes longer time for the team to complete the coverage. When the threats are scattered, the effectiveness of the team is mitigated, since the robots spend much of their time in moving between cells with different threat levels than in covering large areas. As a consequence, locations that reside on connecting paths between cells that belong to different threat levels are repetitively visited by different robots.

Lastly, we have examined the effect of changing the threats ratio in the environment on the ability of the robotic team to cover it. Figure 7 shows the coverage percentage and coverage time for various threat ratios between 0% and 35%. In all the experiments we used teams of 8 robots to cover the area and the threats were concentrated in 10 dangerous areas. As expected, adding more threats the environment causes the team to cover smaller percentage of the area. Note that when the threats ratio is more than 15% the coverage time begins to decrease, since the area that needs to be covered gets smaller thus the team is able to cover it more quickly.

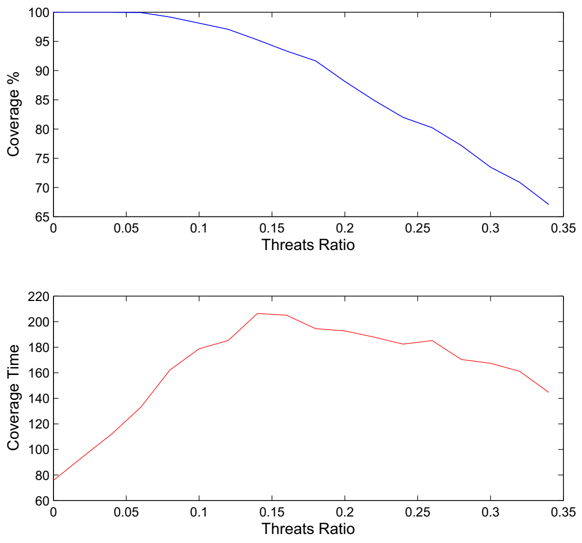

Figure 7. Comparing coverage percentage and coverage time for varying threat ratios between 0 and 0.35.

7 Conclusions and Future Work

In this paper, we described a multi-robot coverage algorithm for adversarial environments that is complete and robust in face of threats harming the robots. Our approach is based on decomposition of the target area into areas of different danger levels, and assignment of the robots to these areas based on the safety of the paths leading to them. In our approach no robot remains idle while there are areas to be covered. We examined the efficiency of the algorithm in terms of both the survivability of the team and the coverage time, and have shown that adding more robots to the team significantly improves both measures. For future work, we would like to extend the multi-robot coverage algorithm to online scenarios, in which the robots are not given a map of the area in advance. In addition, we would like to consider non-stationary environments, where the locations of the threat points may change over time.

REFERENCES

[1] Noa Agmon, Noam Hazon, and Gal A Kaminka, 'Constructing spanning trees for efficient multi-robot coverage', in *IEEE International Conference on Robotics and Automation (ICRA-06)*, pp. 1698–1703, (2006).

[2] J Colegrave and A Branch, 'A case study of autonomous household vacuum cleaner', *AIAA/NASA CIRFFSS*, 107, (1994).

[3] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka, 'Multi-robot area patrol under frequency constraints', *Annals of Mathematics and Artificial Intelligence*, **57**(3-4), 293–320, (2009).

[4] Hermann Endres, Wendelin Feiten, and Gisbert Lawitzky, 'Field test of a navigation system: Autonomous cleaning in supermarkets', in *IEEE International Conference on Robotics and Automation (ICRA-98)*, volume 2, pp. 1779–1781, (1998).

[5] Yoav Gabriely and Elon Rimon, 'Competitive on-line coverage of grid environments by a mobile robot', *Computational Geometry*, **24**(3), 197–224, (2003).

[6] Enric Galceran and Marc Carreras, 'A survey on coverage path planning for robotics', *Robotics and Autonomous Systems*, **61**(12), 1258–1276, (2013).

[7] Anouck R Girard, Adam S Howell, and J Karl Hedrick, 'Border patrol and surveillance missions using multiple unmanned air vehicles', in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pp. 620–625, (2004).

[8] Noam Hazon and Gal A Kaminka, 'On redundancy, efficiency, and robustness in coverage for multiple robots', *Robotics and Autonomous Systems*, **56**(12), 1102–1114, (2008).

[9] George Karypis and Vipin Kumar, 'Multilevel k-way partitioning scheme for irregular graphs', *Journal of Parallel and Distributed computing*, **48**(1), 96–129, (1998).

[10] Robert Krauthgamer, Joseph Seffi Naor, and Roy Schwartz, 'Partitioning graphs into balanced components', in *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 942–949, (2009).

[11] Harold W Kuhn, 'The Hungarian method for the assignment problem', *Naval research logistics quarterly*, **2**(1-2), 83–97, (1955).

[12] Chaomin Luo, Simon X Yang, and Deborah A Stacey, 'Real-time path planning with deadlock avoidance of multiple cleaning robots', in *IEEE International Conference on Robotics and Automation (ICRA-03)*, volume 3, pp. 4080–4085, (2003).

[13] Chaomin Luo, Simon X Yang, Deborah A Stacey, and Jan C Jofriet, 'A solution to vicinity problem of obstacles in complete coverage path planning', in *IEEE International Conference on Robotics and Automation (ICRA-02)*, volume 1, pp. 612–617, (2002).

[14] Jean Daniel Nicoud and Maki K Habib, 'The Pemex-B autonomous demining robot: perception and navigation strategies', in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 'Human Robot Interaction and Cooperative Robots'*, volume 1, pp. 419–424, (1995).

[15] David Portugal and Rui Rocha, 'A survey on multi-robot patrolling algorithms', in *Technological Innovation for Sustainability*, 139–146, Springer, (2011).

[16] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset, 'Efficient boustrophedon multi-robot coverage: an algorithmic approach', *Annals of Mathematics and Artificial Intelligence*, **52**(2-4), 109–142, (2008).

[17] Israel A Wagner, Michael Lindenbaum, and Alfred M Bruckstein, 'Distributed covering by ant-robots using evaporating traces', *IEEE Transactions on Robotics and Automation*, **15**(5), 918–933, (1999).

[18] Roi Yehoshua and Noa Agmon, 'Adversarial modeling in the robotic coverage problem', in *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, pp. 891–899, (2015).

[19] Roi Yehoshua and Noa Agmon, 'Online robotic adversarial coverage', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-15)*, pp. 3830–3835, (2015).

[20] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Robotic adversarial coverage of known environments', *International Journal of Robotics Research*, Advance online publication. doi:10.1177/0278364915625785, (2016).

[21] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Robotic adversarial coverage: Introduction and preliminary results', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-13)*, pp. 6000–6005, (2013).

[22] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Safest path adversarial coverage', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-14)*, pp. 3027–3032, (2014).