Securing Hostile Terrain with a Robot Team

Kristel Verbiest, Eric Colon
Royal Military Academy, Department of Mechanical Engineering
Av. de la Renaissance 30, 1000 Brussels
{kristel.verbiest,eric.colon}@rma.ac.be

Abstract
In some more complicated tasks the use of multiple robots can be advantageous, as there are some tasks that are just not achievable by a single robot. Teams of cooperating robots can accomplish more and faster than individual robots, as is the case in the intruder detection problem. Here, two off-line algorithms are presented for intruder detection in a known environment. In the first approach, the objective is to perform surveillance of a known terrain with multiple robots. The goal is to create a path for every robot, such that the union of all paths generates a full coverage of the terrain and the total coverage time is minimized. Terrain coverage techniques are used to generate just these paths. The second approach will let the robot team perform a security sweep of a known area to detect mobile adversaries. The robots will move from end to end of the map in line formation, preventing enemies to slip through. Securing an area by performing a sweep is a complex task: the coordination between the robots must be tight not to break the sweep line and no space can be left undetected.

Introduction

In certain situations, such as surveillance, transport…, the use of multiple robots can be beneficial, as some complicated tasks are just not achievable by a single robot. The problem studied here is that of intruder detection by multiple robots in a known environment. The robots are required to complete their mission without colliding with each other or the environment, preferably taking the shortest possible paths. The robot team is assumed homogeneous, with 360° FOV and limited sensor ranges. Two algorithms are presented as possible solutions to the intruder detection problem in part 1 and part 2 respectively.
In part 1, the objective is to perform surveillance of a known terrain with multiple robots in order to detect intruders. A team of robots each equipped with a sensor, are required to jointly sweep the entire given terrain while minimizing the total coverage time. Some other real-world applications include lawn mowing, cleaning, chemical spill clean-up, harvesting, search-and-rescue, intrusion detection and mine clearing…
In part 2 a multi-robot line sweep algorithm is described. The objective here is to perform a sweep to detect mobile adversaries in the terrain. The robots will move from end to end of the map in line formation, preventing enemies to slip through. Security sweep requires multiple robots, as most maps cannot be swept by a single robot. If the perimeter stays intact all intruders will be detected once the sweep has completed. Securing an area by performing a sweep is a complex task: it requires tight coordination between the robots not to break the sweep line. As an underlying basis the A* algorithm is used to plan the paths. Applications can be found in security and safety domains. In both approaches an approximate cellular decomposition is used to model the environment.

1. Multi-Robot Surveillance

The problem considered here is that of multi-robot surveillance of a known terrain with probability of adversaries attempting to penetrate the area. The surveillance problem requires that the every location in the area is visited repeatedly in order to monitor change in the state. Here this problem is tackled by using terrain coverage techniques. A team of robots each equipped with a sensor are required to jointly sweep the entire given terrain while minimizing the total coverage time. In other words the goals is to find a coverage path (sequence of waypoints) for every robot, such that the union of all paths generates a full coverage of the terrain and the total coverage time is minimized. Some other real-world applications include lawn mowing, cleaning, chemical spill clean-up, harvesting, search-and-rescue, intrusion detection and mine clearing…

1.1 Existing Multi-Robot Coverage algorithms

While single-robot coverage algorithms have received a lot of attention, there are currently many fewer algorithms for the multi-robot coverage problem. An overview is given in [1]. A method, underlying several coverage algorithms, suggests the use of spanning trees as base for creating coverage paths. The single-robot coverage problem is solved essentially optimally by Spanning Tree Coverage (STC), a polynomial time coverage algorithm that decomposes terrain into cells, computes a spanning tree of the resulting graph, and makes the robot circumnavigate it [2]. Naturally, multiple robots will expedite this process, as they can cover in parallel. The multi-robot coverage problem is to compute a trajectory for each robot so that the cover time is minimized. Spanning Tree Coverage (STC) was generalized to Multi-Robot Spanning Tree Coverage (MSTC) in [3].
Other strategies are given in [3], [4], and [5]. In [3] algorithms for multi-robot coverage are presented, that are complete and robust in the face of robot failures. The algorithms are based on spanning tree coverage of approximate cell decomposition. In [4] a new multi-robot coverage algorithm, called Multi-Robot Forest Coverage (MFC), is presented. MFC is a polynomial-time multi-robot coverage heuristic for finding a tree cover with trees of balanced weights, one for each robot. [5] deals with constructing spanning trees for offline coverage that reduces the total coverage time obtained by algorithms based upon it.

1.2 Minimum Spanning Tree Coverage

To generate patrol paths for the robots, a minimum spanning tree of the map will be calculated. In graph theory a Minimum Spanning Tree (MST) of a weighted graph is subset of edges of minimum total weight which span all the nodes. Among existing algorithms there are the **PRIM** and Kruskal algorithm. The minimum spanning tree can be found in polynomial time.

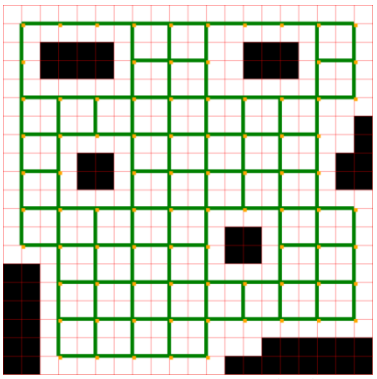

Figure 1a: The graph structure *G = (V, E)* for *R* = 0 of given map.

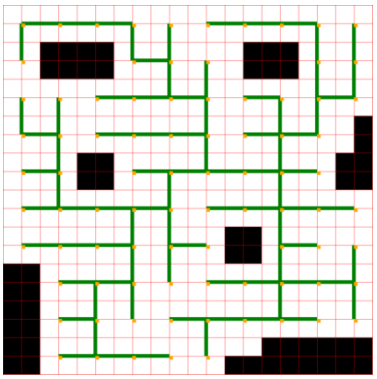

Figure 1b: The minimum spanning tree of the graph in figure 1a.

The population of *N* robots is considered to be identical. Each robot is equipped with a sensor with a field of view (FOV) of 360° and limited sensor range *R*. Sensor ranges are defined here as follows:

R = 0, the robot only covers current cell.
R = 1, the robot covers everything within 1 cell distance of its current position.
R = 2, the robot covers everything within 2 cells distance of its current position.
…

The map of the area is divided into square cells of size 4*R*. The resulting cells that possibly contain obstacles are discarded. A graph structure ***G = (V, E)*** can be obtained from this data. The nodes ***V*** are defined by the centres of the 4*R* sub-cells, displayed in orange in figure 1a. The edges ***E*** are the line segments obtained by connecting the centres of adjacent 4*R* sub-cells, displayed in green in figure 1a. From this graph a spanning tree can be constructed using for instance the **PRIM** algorithm, see figure 1b.

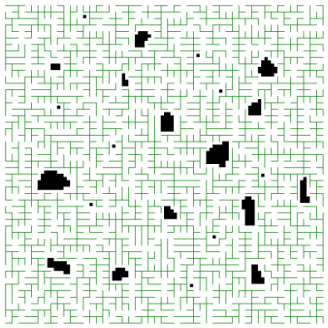

Figure 2a: Minimum spanning tree without preference to horizontal or vertical edges.

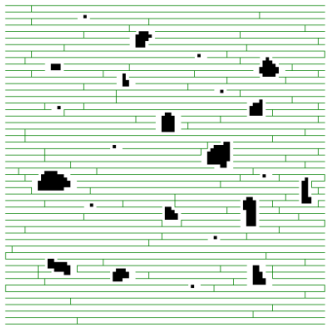

Figure 2b: Minimum spanning tree with preference to horizontal edges.

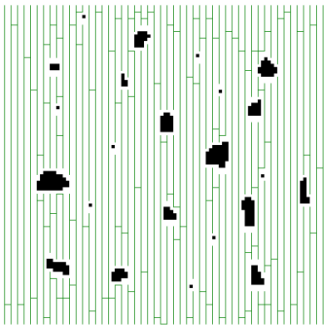

Figure 2c: Minimum spanning tree with preference to vertical edges.

The paths that accompany the spanning tree are obtained by moving along the sides of the tree, turning clockwise at endpoints. The number of robots will determine how long it takes to cover the free space. The resulting path of the spanning tree can be divided by the number of robots, and each robot will be assigned one part, see figure 3a. Different colour zones represent the path for one particular robot to follow. The different robots move to their start positions, and then start moving on their 'coverage paths'. As each cell is visited only ones, no collisions will occur. When the robots are finished they can continue to the next zone to provide repetitive coverage.

There can be alternated between different surveillance routes by changing the spanning tree regularly leading to different surveillance paths. This is only possible when multiple solutions exist, which is mostly the case, especially when the weight of the edges are the same.

Going straight is less demanding on the robot than making many turns while following the prescribed surveillance path. In order to minimize the turns the robot must perform, preference can be given to either horizontal or vertical edges, leading to spanning trees displayed in figures 2a, 2b and 2c. This is done by giving the respective edges a lower edge weight.

Depending on the range of the sensor, different spanning trees will be formed. In figures 3a, 3b and 3c, spanning trees of sensor range 0, 1, and 3 are respectively shown, together with the generated paths for 20 robots in different colours. As can be seen, the higher the sensor range, the shorter the paths. The regions where there is

no robot coverage, inherent to this approach, get larger with increasing sensor range. When the sensor range is greater than 0, its range is displayed in the figures in light green.

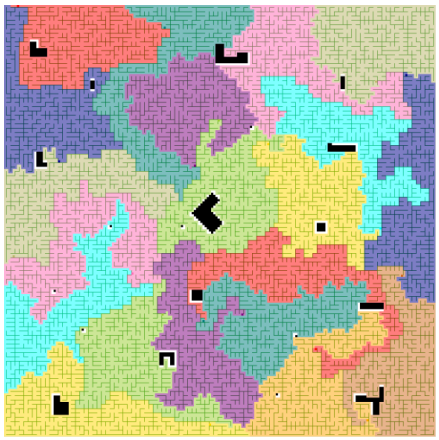

Figure 3a: Map with minimum spanning tree with sensor range 0, and surveillance paths generated for 20 robots.

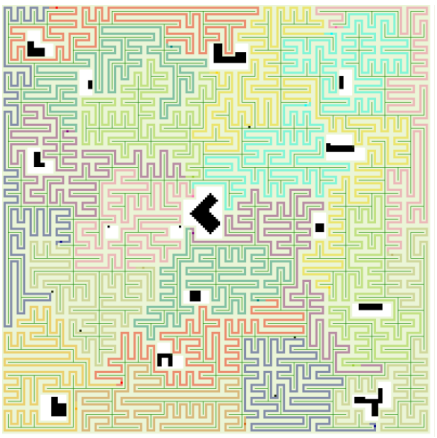

Figure 3b: Map with minimum spanning tree with sensor range 1, and surveillance paths generated for 20 robots.

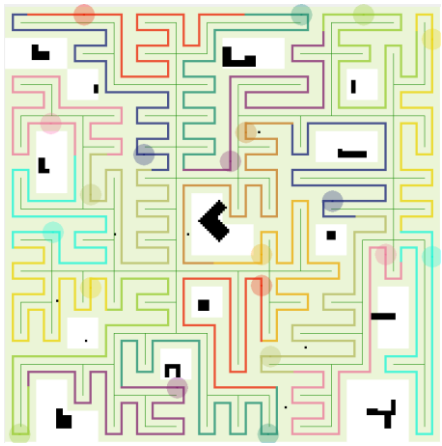

Figure 3c: Map with minimum spanning tree with sensor range 3, and surveillance paths for 20 robots.

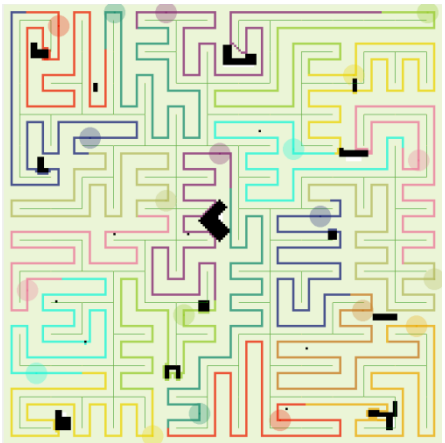

Figure 3d: Map with adjusted minimum spanning tree with sensor range 3, and adjusted surveillance paths for 20 robots.

1.3 Coverage Improvement

As the sensor range increases more and more space will be left unchecked. In this section a variation on the method above is described. It improves results with respect to uncovered areas, with the drawback that some redundancy is introduced.

In the method above, the map of the area is divided into square cells of size 4*R* and the resulting cells that possibly contain obstacles are discarded. Here cells that contain obstacles are retained, as long as they also enclose obstacle free space. A graph structure ***G = (V, E)*** can be obtained from this data. From this graph a spanning tree can again be constructed. Here preference is given to edges that do not lie in obstacle space, thus avoiding edges that cross obstacles. The surveillance paths that result from this spanning tree will traverse obstacle space. To remedy this, at these locations paths will be planned locally using the A* algorithm. Start and goal points will be the point just before the obstacle is entered and the point right after the obstacle is exited. In figure 4a and 4b can be seen how with A* planning the robot can locally deviate from the initial patrol path. In figure 3c and 3d the results for both methods can be compared. For the same map, sensor range and number of robots, the surveillance paths are displayed.

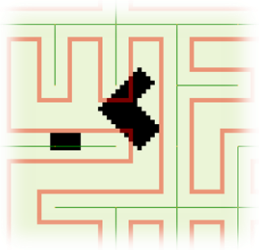

Figure 4a: Initial surveillance path, traversing obstacle space.

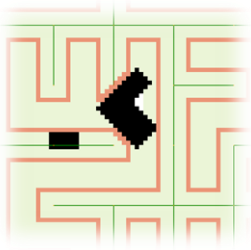

Figure 4b: Surveillance path altered by using A* to locally plan a path to guide the robot around the obstacle.

2. Multi-Robot Line Sweep

In this second part the robot team, with limited sensor range, will have to perform a security sweep of a known area to detect mobile adversaries. It can be considered as a form of the pursuit-evasion problem [9], in which one or more searchers must move through a given environment in such a way as to guarantee detection of any and all evaders, which can move arbitrarily fast. Here the goal is to let the robots move from end to end of the map in line formation, preventing enemies to slip through. Security sweep requires multiple robots, as most maps cannot be swept by a single robot. Securing an area by performing a sweep is a complex task: the timing must be right not to break the sweep line and every space in the area needs to be unchecked. The task cannot be divided into subtasks that each robot can complete; the actions of the robots must be tightly coordinated. At certain points robots will have to secure a part of the map that is already swept while the other robots move forward to sweep following parts. If the line stays intact all intruders will be detected once the sweep is completed. An ideal solution to the security sweep problem would require that mission time and distance traversed would be minimized and intruders are prevented to slip through the sweep line into the previously secured area.

2.1 Existing Sweep Algorithms

Different strategies are possible to sweep an environment. Some are described in [6], [7] and [8]. The approach taken here strongly builds on the existing cooperative A* algorithm [10]. In [6] the algorithm used is inspired by methods used for complete coverage of an area. A strategy is developed for exploration of an unknown environment with a multi-robot system. The communication between robots is restricted to line-of-sight. The map is divided in different strips and the idea is to let a robot formation with zigzag-shape scan strip after strip. When an obstacle is encountered the algorithm tries to guide the robot group around it in a time-optimal way.

In [7] a market-based coordination framework, Hoplites, is presented that addresses tight coordination in multi-robot teams. The Hoplites framework consists of two novel coordination mechanisms: passive and active coordination. Passive coordination quickly produces locally-developed solutions while active coordination produces complex team solutions via negotiation between teammates. Robots use the market to efficiently vet candidate solutions and to choose the coordination mechanism that best matches the current demands of the task. The Hoplites framework can be applied to the security sweep domain. In [3], the continuous area sweeping task is tackled. Here a group of robots must repeatedly visit all points in a fixed area, possibly with non-uniform frequency. The focus of this paper is adaptive and decentralized task assignment in continuous area sweeping problems, with the aim of ensuring stability in environments with dynamic factors, such as robot malfunctions or the addition of new robots to the team.

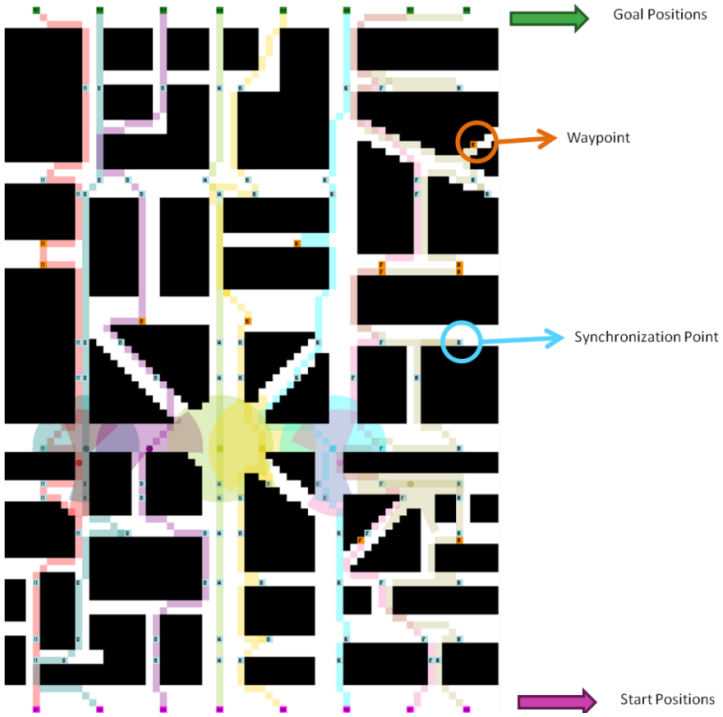

Figure 5: Screenshot showing start and goal positions, waypoint and synchronization points for 8 robots performing sweep. The sensor range is set to *R=5*.

2.2 Set-up

The population of *N* robots is considered to be identical. Each robot is equipped with a sensor with a field of view (FOV) of 360° and limited sensor range R. Sensor ranges are defined here as above. The dimensions of the map are (*Xmax, Ymax).* Each robot is mainly responsible for performing a sweep across imaginary stripes along the Y-axis of width *Xmax/N*. All robots move at the same speed. The robots will synchronize on some horizontal lines

by waiting for each other at specified points; these are synchronization points, shown in blue in figure 5. Waypoints can be placed for specific robots at specific locations to force the robots to pass that position, shown in orange. Start and goal positions are set at both ends of the terrain.

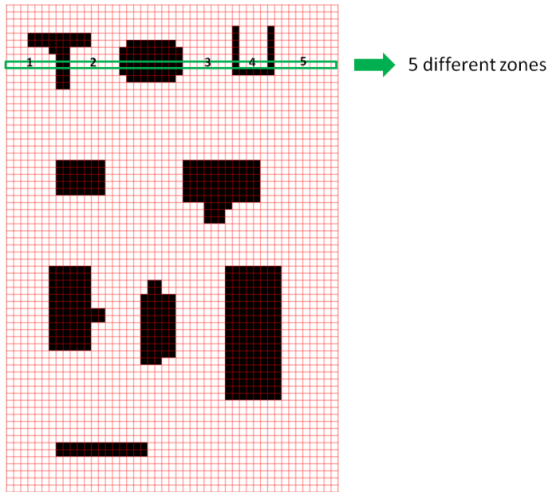

Figure 6: Map showing a line with 5 different zones. In each zone a robot must be present to perform a secure sweep.

2.3 Minimum Robots Required

The number of robots required to perform a secure sweep of the environment will depend on the map and the sensor range of the robots. To determine the minimum number of robots the map will be scanned line by line. In such a line the number of separated white spaces, zones, will be counted. The line with the maximum number of zones will determine the minimum number of robots needed. In figure 5, a line is shown with 5 different zones. As this is a line with maximum number of zones, the minimum number of robots needed to perform a secure sweep is 5.

The minimum number or robots determined in the previous step may not be sufficient. If the zone is greater than the range of the sensor, it may be that more robots must occupy one zone to perform a secure sweep. For instance in figure 6, zone 5 is 9 cells wide. For a robot with sensor range 3, only a 7 cell wide space could be scanned. Therefore in this case, zone 5 would require 2 robots.

2.4 Perimeter Violations

The perimeter is considered broken when it is possible for an intruder to enter to the portion of the terrain that is supposed to be secure. It is assumed that the intruder can move arbitrarily fast and can be arbitrarily small; so even if the sweep line is broken only for a short time span or the gap in the line is small, the previously secured area could be contaminated. Figures 7a and 7b give examples of an intact and broken sweep line according to this definition. The paths the robots will follow are displayed in different colours in the figures shown here; the coloured circles designate the accompanying sensor range of the robots, taking occlusion from obstacles into account.

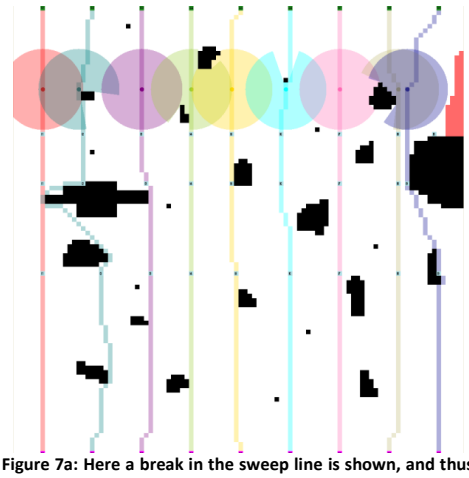

Figure 7a: Here a break in the sweep line is shown, and thus intruders will be able to enter the previously cleared area. The unscanned area in the top right corner is shown in red.

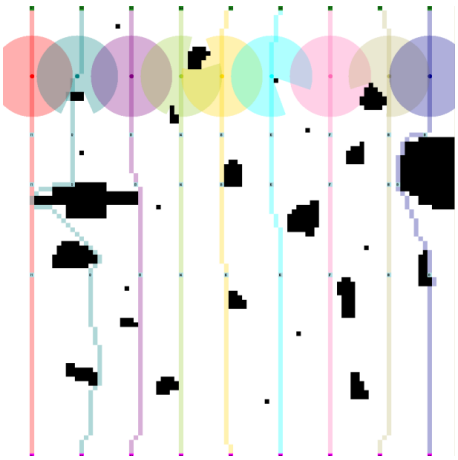

Figure 7b: The sweep line is considered intact; no intruder from the top can enter the cleared area.

2.5 Repairing Perimeter Violations

In the first step the paths for the robots from start to goal are calculated using the cooperative A* algorithm, which will prevent collisions with other robots and obstacles. The second step consists of iteratively updating the paths of the robots to repair existing perimeter violations until the paths constitute a secure sweep. The paths

are updated by adding waypoints for certain robots and synchronization lines where robots will wait for each other in order not to deviate too much from the horizontal sweep line. Where these waypoints and synchronization lines are placed depends on the location of the violation occurrence. Synchronization lines are lines where robots will wait until each robot has arrived on this line before continuing on their paths. In these lines a synchronization point is placed for each robot where it will wait if needed.
The sweep line can be considered as a chain of sensor ranges linking every pair of adjacent robots. Robots at the end of the chain are linked to the border of the map. Between two adjacent robots there can only be one obstacle. Between the border and a robot there can be no obstacle. The sensor range of a robot must either reach an obstacle, the border or the sensor range of another robot. If these conditions are not fulfilled, a violation occurs. The waypoints and synchronization points will be calculated for rows where this violation is present. The current positions of the robots on the row are moved in such a manner that the row is without violation. These new positions will be the waypoints/synchronization points the robots must pass on that row thereby eliminating the former violation. Not all the rows with a violation are repaired at once, as it is often the case that by repairing one row, others are repaired at the same time, as a violation usually spans more than one row. The row where a conflict occurs is repaired by calculating new waypoints and synchronization points according to the strategy shown in figure 8.
The robots will be forced to pass by these newly calculated positions by placing synchronization points in the map. If some of the newly synchronization points coincide on the row, waypoints are used instead. The resulting paths are then again checked for violations. If a violation still is present the above steps are repeated.

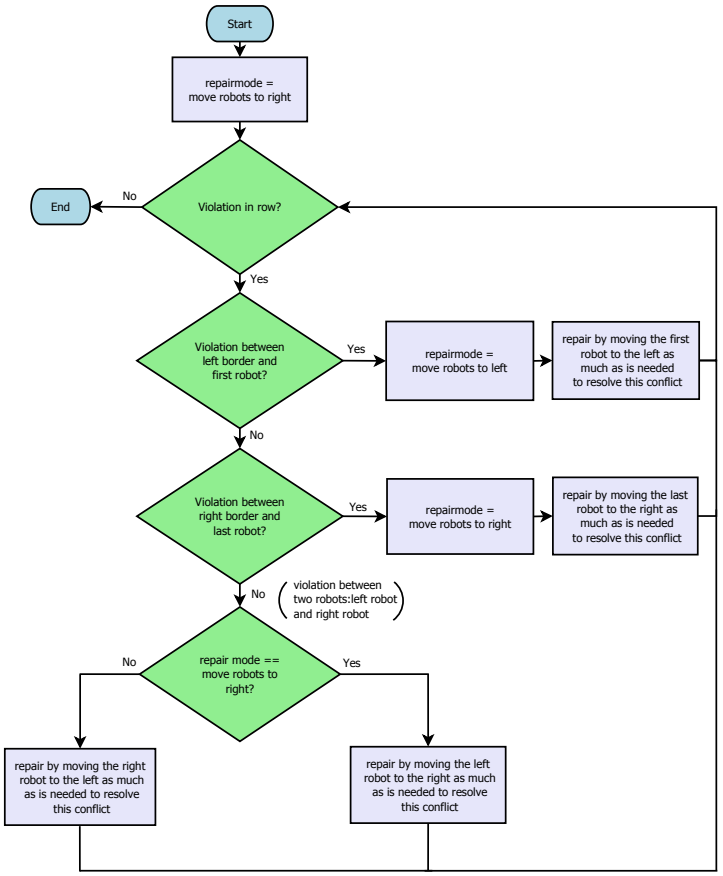

Figure 8: Flowchart showing strategy for calculating new waypoints/synchronization points for row with violation.

The different steps, for different violation rows, are shown for an example map in figures 9a, b, c, d, e and f. Red areas designate unscanned areas, and the blue stripes on the left indicate in which rows a gap in the sweep line occurs. The red circles point out where the violation occurs, the arrow point out the row where the waypoints and or synchronization points are placed.

3. Summary

Here, two off-line algorithms are presented for intruder detection in a known environment by a robot team. The first approach is geared towards continuous surveillance of a known are; the second to performing a one-time sweep, resulting in a guaranteed cleared area. Both algorithms could be extended to work with robots with limited FOV. Or the population could be replaced with a heterogeneous group of robots, with different characteristics: different speeds, different size robots, different sensor (and thus FOV).

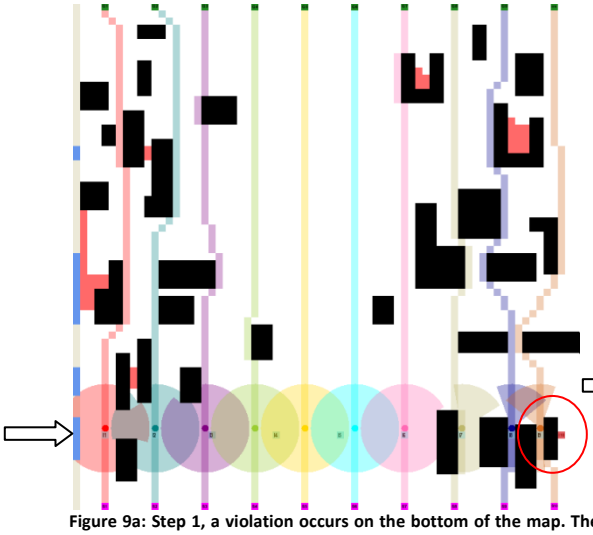

Figure 9a: Step 1, a violation occurs on the bottom of the map. The newly calculated synchronization points, shown here, will resolve this problem, results are shown in step 2, figure 9b. The robots will synchronize at these points.

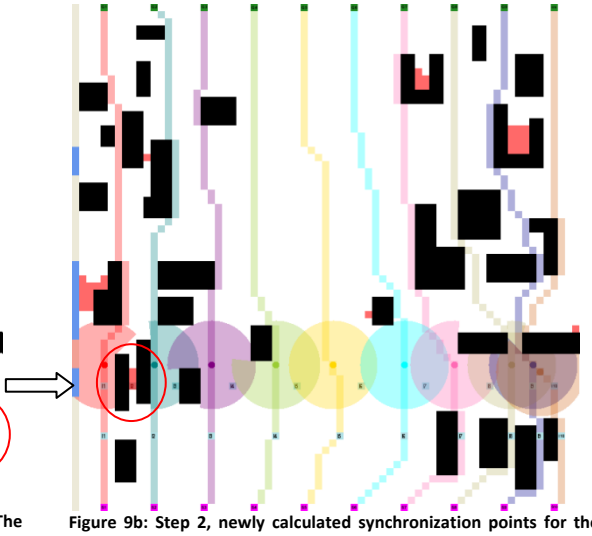

Figure 9b: Step 2, newly calculated synchronization points for the violation are shown, results are shown in step 3, figure 9c. The robots will synchronize at these points.

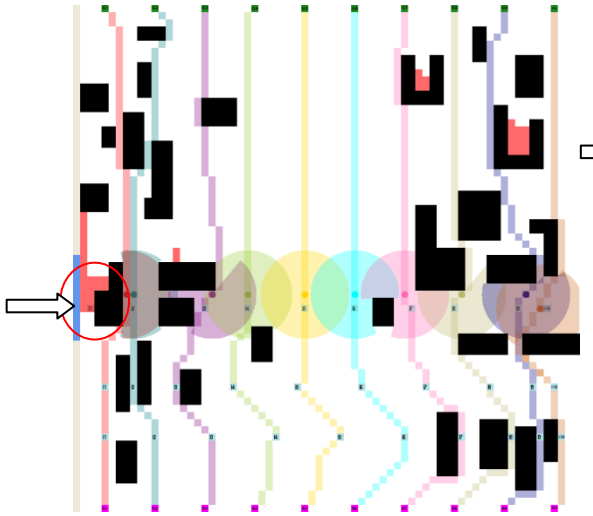

Figure 9c: Step 3, newly calculated synchronization points for the violation are shown, results are shown in step 4, figure 9d. The robots will synchronize at these points.

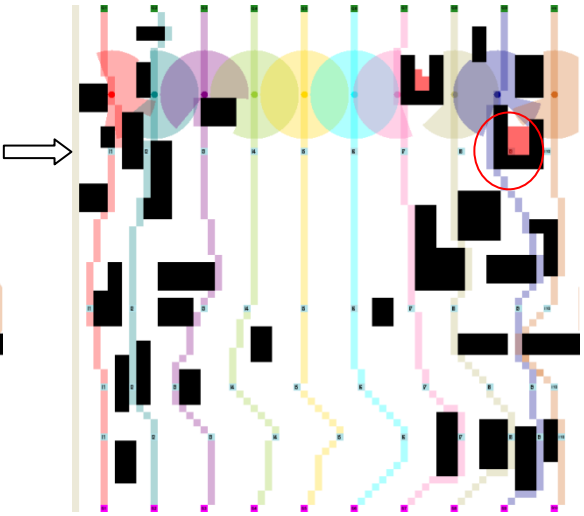

Figure 9d: Step 4, newly calculated synchronization points for the violation are shown, results are shown in step 5, figure 9e. The robots will synchronize at these points.

Figure 9e: Step 5, newly calculated synchronization points for the violation are shown, results are shown in step 6, figure 9f. The robots will synchronize at these points.

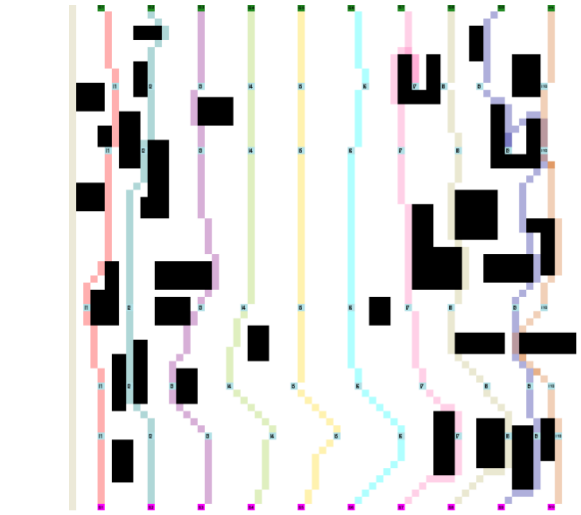

Figure 9f: Step 6, final result after iteratively updating the robot paths. The resulting paths perform a secure sweep.

4. Acknowledgements

The research project Networked Multi-Robot Simulation (NMRS) is funded by the European Defence Agency (EDA). Besides the Royal Military Academy, partners involved in this project are Diehl BGT Defence (Germany - project leader), Otomelara (Italy), Sener (Spain) and FGAN (Germany).

5. References

[1] H. Choset. Coverage for robotics. A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113.126, 2001.

[2] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77.98, 2001.

[3] N. Hazon and G. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage. In *Proceedings of the International Conference on Robotics and Automation*, 2005.

[4] Xiaoming Zheng, Sonal Jain, Sven Koenig, David Kempe. Multi-Robot Forest Coverage. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005. (IROS 2005).

[5] Noa Agmon, Noam Hazon and Gal A. Kaminka. Constructing Spanning Trees for efficient Multi-Robot Coverage.

[6] Jonathan Rogge and Dirk Aeyels. Novel Strategy for Exploration with Multiple Robots.

[7] Nidhi Kalra, Dave Ferguson, and Anthony Stentz. Hoplites: A Market-Based Framework for Planned Tight Coordination in Multirobot Teams.

[8] Mazda Ahmadi and Peter Stone. A Multi-Robot System for Continuous Area Sweeping Tasks.

[9] T.D. Parsons. Pursuit-evasion in a graph. In Y. Alani and D.R. Lick, editors, *Theory and Application of Graphs*, pages 426-441. Springer-Verlag, Berlin 1976.

[10] David Silver, "*Cooperative Pathfinding*".