Decentralized Multi-Robot Coordination for Risky Interventions

Daniela Doroftei, Eric Colon

Royal Military Academy, Department of Mechanical Engineering (MSTA)

Av. de la Renaissance 30, 1000 Brussels

{daniela.doroftei, eric.colon}@rma.ac.be

1. Introduction

Teams of robotic agents can provide a valuable asset in a military context where hostile conditions can create a hazardous working environment. A possible application for these multi-agent robotic systems consists of the surveillance of an urban environment. In this case, a team of robots must scan a city for the presence of enemy forces and take the appropriate action when enemy forces are detected. The deployment of a team of robotic agents described above requires a careful consideration of the autonomous control aspect of the multi-agent team, considering both the autonomous control of each individual robot as the global intelligent operation of the team of robots. Indeed, in order to fulfill the required tasks, the individual robots must be able to operate totally autonomous, and this in possibly hostile terrain. Each robot must seek to fulfill the part of the global task to which it was dedicated, while at the same time, the global control strategy must seek to fulfill the global task within the given constraints (time, energy…). This research subject has been studied before by a number of researchers [1][2][3][5]. In this paper, a novel control strategy is presented for multi-robot coordination. An important aspect of the presented control architecture is that it is formulated in a decentralized context. This means that the robots cannot rely on traditional global path planning algorithms for navigation. The presented approach casts the multi-robot control problem as a behavior-based control problem. In the behavior-based spirit a complex control problem is divided into a set of simpler control problems that collectively solve the original complex control problem [4]. To do this, it is thus necessary to address the problem of coordination of the activities of the behaviors so to satisfy the initial complex system's control objectives. This problem is known as the action selection or behavior fusion problem. The paper describes how each behavior was designed and how the behavior fusion problem was solved. The behavior-based control paradigm was chosen, because it is inherently decentralized and because it thus provides a natural and elegant way to combine the different subtasks and capabilities of each individual robot and because – unlike more traditional sense-model-plan-act approaches – it scales very well when applied to a large number of robots.

2. Global Strategy

Following the requirements of the urban surveillance scenario, the robots must scan a city area for the presence of enemy forces (intruders). The urban setting, considered for this scenario, poses an important difficulty for robot navigation, as can be understood from analyzing *Figure 1*, which shows a top view of the city block used for the simulation of this scenario.

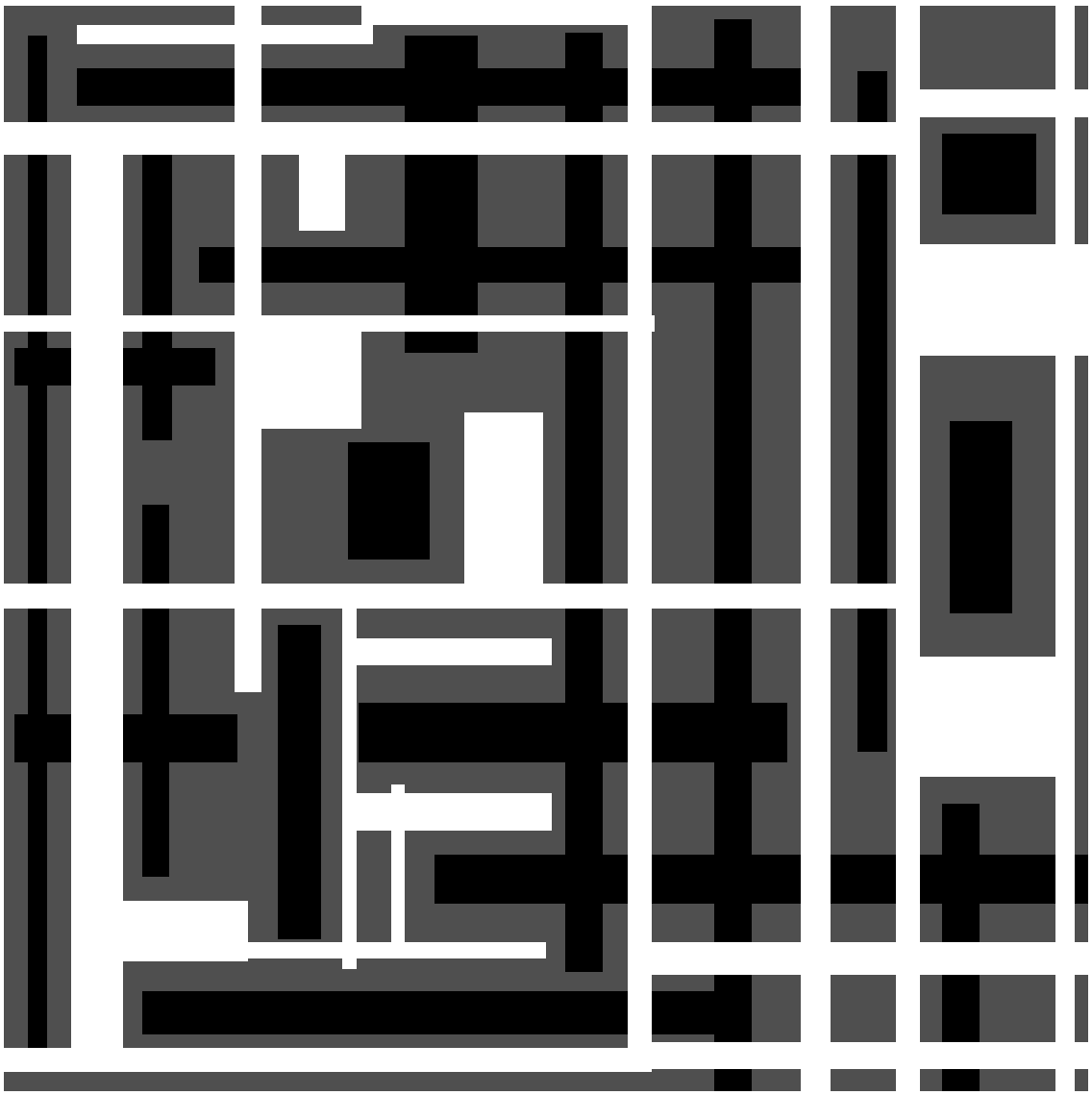


Figure 1: Top View Map of the Simulated Urban zone.
White areas represent streets; Black areas represent buildings; Gray areas correspond to traversable non-street areas (grass)

It is immediately evident that the complexity of the terrain makes it really hard to perform intelligent navigation from one point to another without the knowledge of any map data and use of global path planning algorithms, as is the case in this decentralized context. Furthermore, the position and movement of the intruder robots is a priori unknown and they can only be detected when they are within the sensor range of one of the team members. In the simulation, the intruder positions are initialized randomly and the intruders also follow a randomized movement strategy, which makes it hard for the team members to detect them.

Another question which can be raised is how to react to enemy forces which are detected by a member of the robotic team. The scenario description foresees several levels of response:

1. Report the detected intruder to the base station and take no further actions.
2. Report the detected intruder to the base station and start tracking its movements, without following the intruder.
3. Report the detected intruder to the base station and start tracking its movements, while following the intruder from a distance.
4. Report the detected intruder to the base station and start tracking its movements, while advancing towards the intruder for an interception.

An automated expert system chooses between one of these 4 intruder handling modalities, depending on the *hostility grade* and the type of the intruder. For intruders representing a minor security risk, only a report should be send. Intruders classified as more dangerous must be tracked and – if they represent an even higher security risk – followed too. All these robot actions seek to passively decrease the security risk situation by handling the security breach represented by the intruder, by passive observation only. For intruders of the last class, representing the most severe security risk, the robot must actively seek to neutralize the security risk, by moving towards the intruder position, such that the adequate action can be taken. The implementation of this automated expert system is not consisered in the multi robot simulation and control architecture presented in this text; here it is considered that the intruder handling modality is chosen a priori.

To deal with the issues mentioned above, the control strategy proposed for this scenario considers 4 behaviors:

1. Intruder Searching: Seek for the presence of enemy forces within the field of view of the robot and maximize the terrain coverage such that as much terrain is viewed (cleared) as quickly as possible;
2. Intruder Tracking: Report the detected intruder to the base station and start tracking its movements, without following the intruder;
3. Intruder Following: Report the detected intruder to the base station and start tracking its movements, while following the intruder from a distance;
4. Intruder Interception: Report the detected intruder to the base station and start tracking its movements, while advancing towards the intruder for an interception.

In the following section, each of these behaviors will be discussed more in detail. The behaviors mentioned above are mutually exclusive, meaning that only 1 behavior is active (in 1 robot) at any given time. This means that the action selection or behavior fusion is not an issue in this case, as there are no multiple behavior outputs to be combined. In the initial state, all robots are in the first - intruder searching - behavior, as no intruders have been found yet.

3. Behavior Design

3.1. Intruder Searching

In intruder searching mode, the robots should seek for the presence of enemy forces in the entire designated area. This is a complicated task, as the robots have no prior knowledge whatsoever about the nature of this environment. As such, it is impossible to rely on classic path planning approaches. It is also impossible to calculate a globally optimal multi-robot coverage strategy.

Traditional approaches towards multi-robot coverage rely on the solution of the so-called “multi-agent travelling salesman”. The travelling salesman problem is a combinatorial optimization problem where the goal is to find the shortest path linking a number of cities/points of interest. In the case of intruder searching the places to be visited are related to the sensor range of the robot. The visited places are chosen such that the detection range of enemy forces is maximized as quickly as possible. In a multi-agent context, the task of optimizing the global detection range is subdivided between a number of agents, each solving a subpart of the travelling salesman problem for the designated area.

As stated above, these traditional approaches are not applicable for the presented decentralized scenario, as the robots have no global environmental model. To deal with these issues, the robots employ a local information maximization approach. The basis for this approach is the construction of a local coverage map. With each scan of its enemy detection sensors, each robot stores the returned sensor data in such a local coverage model, by storing the value ”1” in all cells which have been “viewed” by the robot sensors.

$$CoverageMap(i,j) = \begin{cases} 1 & if\ \ Visible(i,j) \\ unchanged & if\ \ !Visible(i,j) \end{cases}$$

The visibility model employed here takes into consideration possible occlusions due to obstacles, as shown on *Figure 2*, which shows on the right side an integrated view of 10 local coverage maps for all of the 10 simulated robots indicated in the urban area, as shown on the left figure.

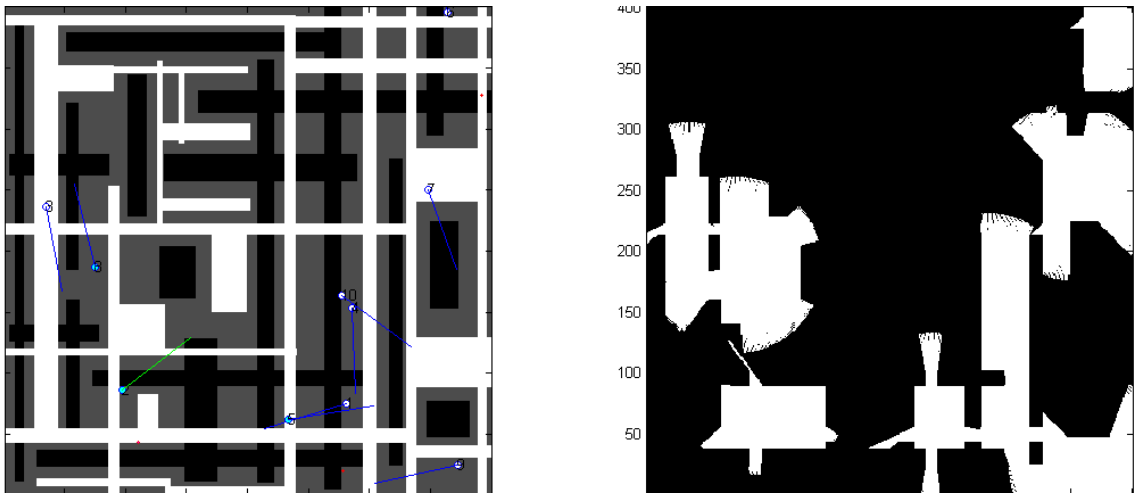


Figure 2: Robot Locations and Integrated Local Coverage Maps

At each iteration of the simulation, the information in the coverage model is “aged” by multiplying all entries of the local coverage map with a value between 0 and 1 (in practice: 0.99). The purpose of this approach is to represent the unreliability of “old” data. Indeed, it is very well possible that there was no intruder present in a

certain cell at time $t=t_k$, but that does not mean that this situation will necessarily stay like this eternally. Intruders can move and can hide in buildings where the robots cannot detect them, such that it is very well possible that at time $t=t_{k+1}$, there will be an intruder present in the same cell. Therefore, the coverage data recorded in the coverage map cannot stay static as well; it must be decreased at each iteration. *Figure 3* shows the effect of the coverage map ageing, by showing the integrated coverage map after a number of iterations. The gray-value areas on this figure represent zones which have been viewed by a robot in the past, but which are currently not in the visibility field of any robot. As such the information which was gathered there some time ago cannot be trusted completely and – as the data becomes older – it becomes more and more likely that a robot will return there to re-check the gathered data.

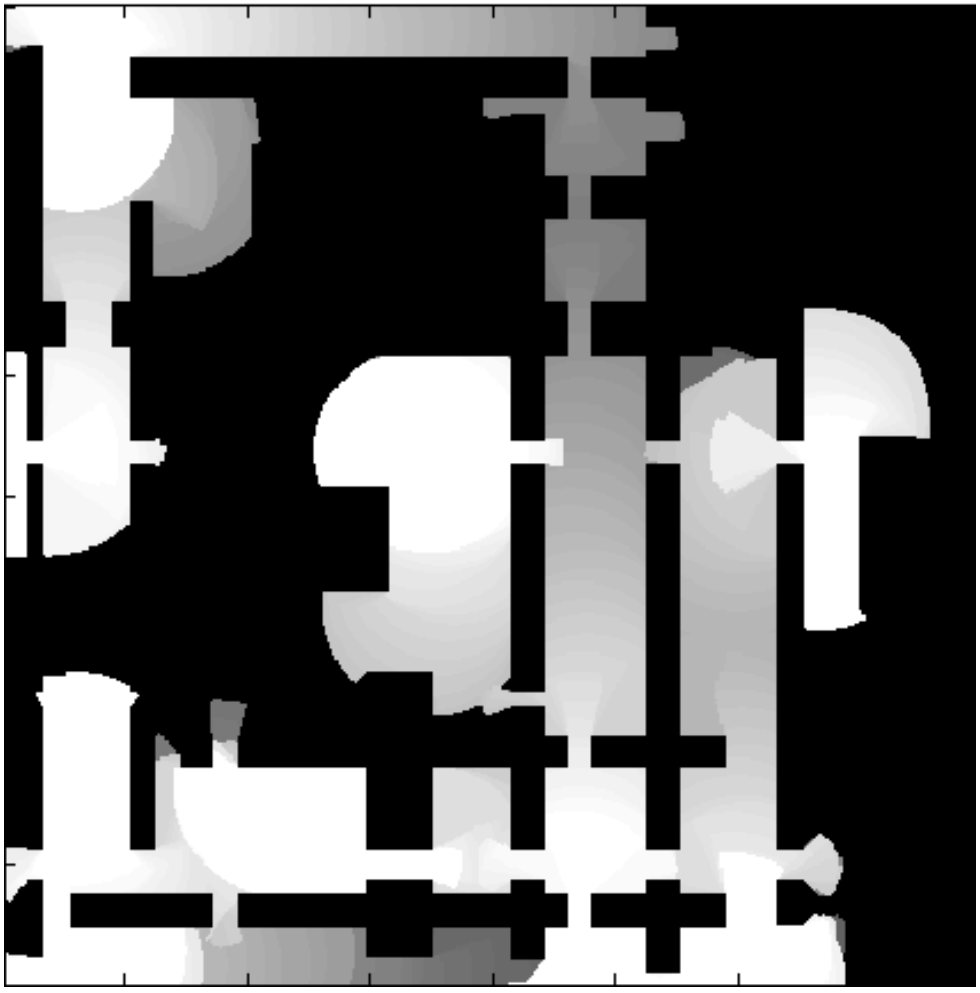


Figure 3: Integrated Coverage Maps after some iterations

For navigation, each robot then analyzes its local coverage map and calculates for each possible move to be made, leading to a new position $\boldsymbol{x}$, the amount of cells already covered, taken into account the visibility model at location $\boldsymbol{x}$:

$$Score_{Coverage}(\mathbf{x}) = \sum_{Visible(\mathbf{x})} CoverageMap^2$$

The new location ***x*** where $Score_{Coverage}(\mathbf{x})$ is minimal is the point where the most new information can be possibly gained when advancing to this location. Otherwise put, the point ***x*** maximizes the information gain and is thus the best place to go to detect intruders.

However, the information maximization cannot be the only movement criterion within the intruder searching behavior. There are other constraints to be taken into account such as:

- Resistance to turning: A mobile robot – like any mobile object – prefers to continue its current status. It is inefficient to make a robot change directions at every iteration; therefore it should be preferred not to change the turning angle too much.
- Terrain traversability: Different types of terrain feature different degrees of traversability. Mobile robots prefer to drive on streets and cannot drive through obstacles (buildings, road blocks). Between these extremes, all intermediate levels of traversability are possible: grass is easily traversable, while mud is harder.

These constraints can be integrated in the intruder searching behavior by defining a $Score_{Turning}(\mathbf{x})$ and a $Score_{Traversability}(\mathbf{x})$. For the resistance to turning, it is straightforward to express this score as the angular difference between the current robot orientation and the envisaged orientation:

$$Score_{Turning}(\mathbf{x}) = AngleDifference(\theta_{robot}, \theta(\mathbf{x}))$$

For the expression of the terrain traversability constraint, the map data is employed. From the example urban map used in this simulation, as shown on *Figure 1*, it is clear that obstacles are marked as black areas (numerical value 0) and streets are labeled white (numerical value 1). All intermediate values, corresponding to gray areas, are possible. As such the traversability properties of the terrain are expressed on the map and a movement constraint can be directly derived from these values:

$$Score_{Traversability}(\mathbf{x}) = MapValue(\mathbf{x})$$

Finally, the total score for each possible new location is calculated by computing a weighted sum of all the different constraints explained above:

$$Score(\mathbf{x}) = w_{Coverage} Score_{Coverage}(\mathbf{x}) + w_{Turning} Score_{Turning}(\mathbf{x}) + w_{Traversability} Score_{Traversability}(\mathbf{x})$$

The different weights express the importance which is given to each of the different constraints. This depends largely on the capabilities of the different robots. For example, for simple road vehicles, high values will be required for $w_{Turning}$ to keep the robot from turning too much and for $w_{Traversability}$ to force the robot to stay on the street. For vehicles with more off-road capabilities, it is possible to lower the value of $w_{Traversability}$. The minimization of the total score over all possible new locations to go for a robot then decides on the robot motion, for the intruder searching behavior.

When an intruder is detected by a robot, the intruder searching behavior is abandoned by that robot and the appropriate other behavior (intruder tracking, following or interception, depending on the automated expert system) is triggered. The intruder handling behavior is initially triggered only for the robot who detected the

intruder, but depending on the intruder handling strategy, the robot can ask for assistance of other robots in the area to help with the intruder handling. If the intruder handling strategy consists only of tracking the intruder, then it is not necessary to call for help from other robots: indeed, if these other robots do not see the intruder, then a switch of behavior has no purpose in this case. In the other 2 cases (intruder following and intruder interception), the robot who detects the intruder will alert a number of robots in the area to help with the intruder handling task. The robots to be alerted are selected on the basis of 2 considerations:

1. The robots must be close, such that they can arrive quickly to help
2. The robots must not already have another intruder handling task at hand

Following this strategy, the robot detecting the intruder thus selects the closest robots which are free from any other intruder handling duty and sets their behavioral state accordingly.

3.2. Intruder Tracking

When executing the intruder tracking behavior, the robot must keep the intruder in view, but not take any pursuit action. When this situation occurs, the robot simply orients itself into the direction of the detected intruder and then stops it motors. When the contact with the intruder is lost, the robot will revert to the intruder searching behavior.

3.3. Intruder Following and Intruder Interception

A robot in the intruder following state moves into the direction of the intruder, but keeps a certain security distance d_{sec} in order not to be detected itself. To achieve this, the distance to the intruder is calculated for each possible new position ***x*** of the robot:

$$Score_{GoToIntruder}(\mathbf{x}) = abs(\sqrt{(x - x_{\text{int}\,ruder})^2 + (y - y_{\text{int}\,ruder})^2} - d_{Sec})$$

This score provides a constraint, which is integrated with the score for turning and for terrain traversability to form a new global score for the intruder following behavior:

$$Score(\mathbf{x}) = w_{GoToIntruder} Score_{GoToIntruder}(\mathbf{x}) + w_{Turning} Score_{Turning}(\mathbf{x}) + w_{Traversability} Score_{Traversability}(\mathbf{x})$$

By minimizing this global score, the robot advances towards the intruder, yet keeps a security distance. However, this approach towards intruder following does not guarantee a successful pursuit of the intruder in all cases. Indeed, if obstacles are present within the trajectory from the robot to the intruder, then the robot is not capable of intelligently avoiding these obstacles, as it has no global environmental model and thus no idea of the specific nature of the obstacle.

The intruder interception behavior can be seen as a form of the intruder following behavior, where the security distance is equal to zero. The implementation of the intruder interception behavior is thus just a question of applying the intruder following behavior with a security distance $d_{\text{sec}} = 0$.

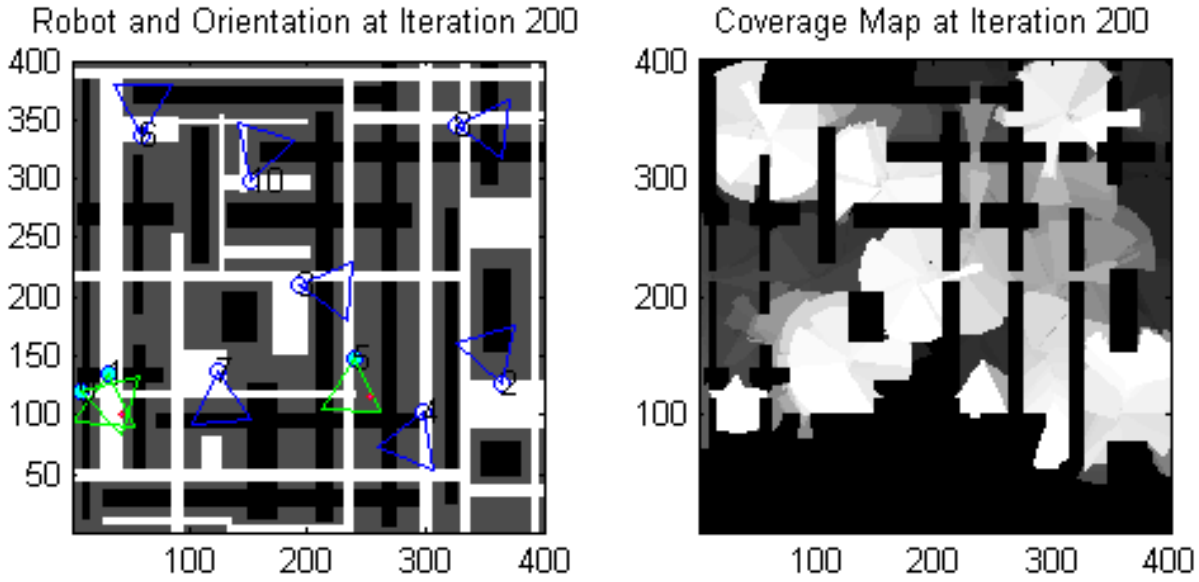
4. Behaviour fusion

As stated above, there is always only 1 high level behavior active at the same time, which makes that there is no real behavior fusion to be performed. In fact, the combination of different behaviors plays at a lower level for this

scenario, as there are the different constraints which need to be combined together. As was shown above in the behavior design section, this is achieved through applying a weighted sum of all individual constraints. This approach is also known as the weighting method and it is the most classical methodology for behavior fusion.

5. Results

Figure 4 shows the user interface of the multi-robot simulator. The top left figure shows the top view map of the urban environment, with the robot and intruder positions marked. The top right figure shows the integrated coverage map of the robots at this time instance in the iteration. It can be seen that a large section of the map has been covered by now, only in the bottom there is still an area marked black, which means it hasn't been visited (lately). The bottom figure shows a textual view of the robot state and the commands to be sent to the robot. At each iteration, the simulator displays a time indication, and for each robot its position, orientation, speed, behavioural state, whether an intruder has been detected and is so, where it has been detected. This information serves as input for the low level robot motion planner to plan a path to the different waypoints.



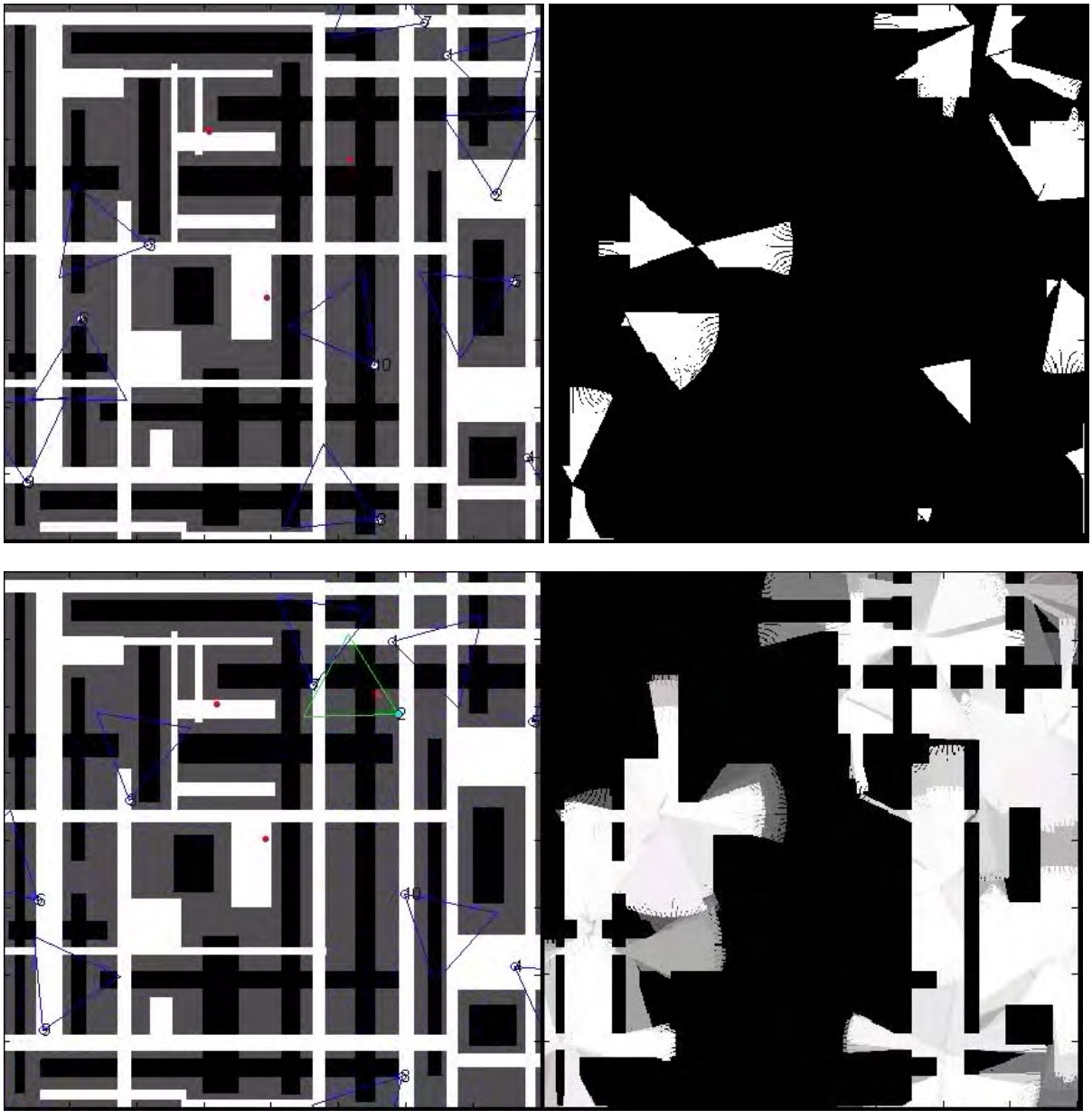
Timestep	Robot Number	X	Y	Orientation	Speed	Behavior	Intruder Detected	Intruder Position
200	1	33	135	-97	0.0	Following	Yes	(42, 101)
200	2	364	127	108	4.0	Searching	No	
200	3	193	210	354	4.0	Searching	No	
200	4	299	104	249	4.0	Searching	No	
200	5	241	147	-95	0.0	Following	Yes	(253, 115)
200	6	61	336	90	4.0	Searching	No	
200	7	125	137	275	4.0	Searching	No	
200	8	325	346	353	4.0	Searching	No	
200	9	9	120	-16	0.0	Following	Yes	(42, 101)
200	10	152	299	74	4.0	Searching	No	

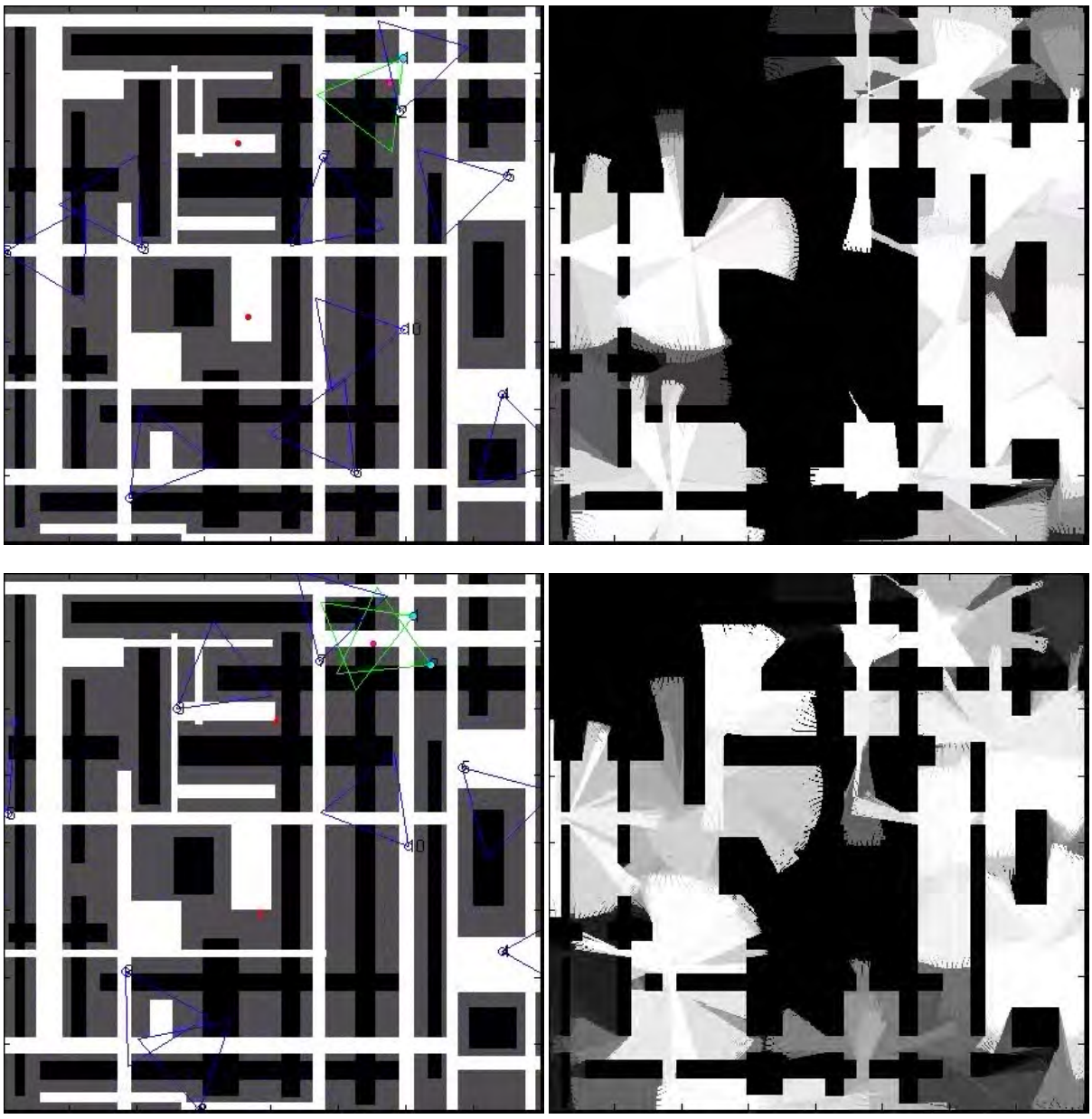
Figure 4: User interface of the multi-robot simulator

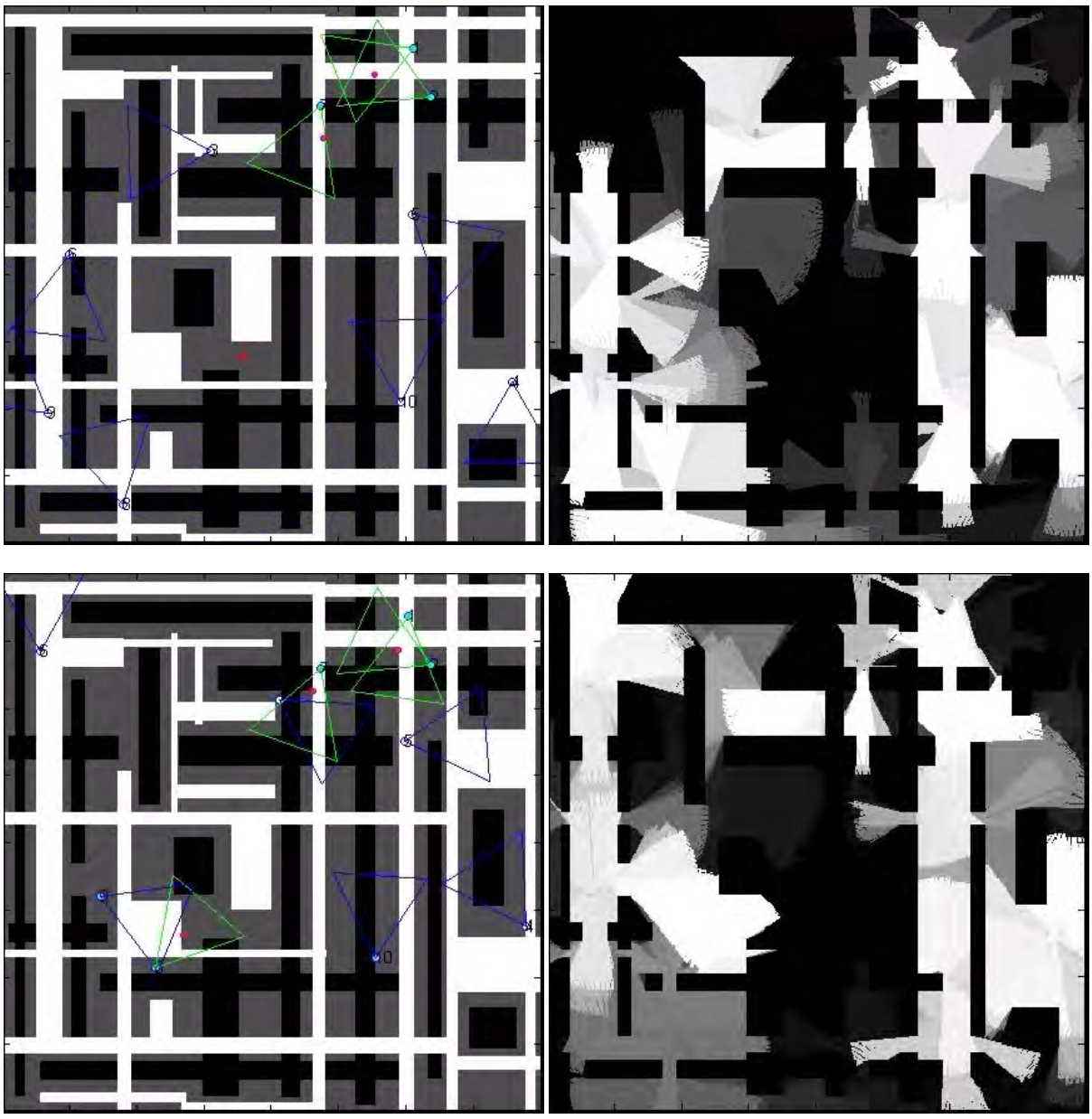
In order to assess the behaviour of the presented multi-robot coordination strategy, it is useful to show the evolution of the urban surveillance simulation step by step. Therefore, Figure 5 shows the different steps of a

simulation, indicating at different time steps the map with robots and intruders (on the left) and the integrated coverage map (on the right).

It can be noticed that in the beginning, no intruder is seen by any robot and that the integrated coverage map is mainly black (indicating that the area is unvisited). At the second time instance, one robot has detected an intruder and has asked for other robots to come and help him to keep this intruder in view. It can be seen in later time instances that other robots have reacted to this call and multiple robots track the intruder. At the fifth and sixth time instance, other robots have detected other intruders, and have called for help from other robots. As such, it can be seen that in the end, each intruder has been localized and is tracked by multiple robots.







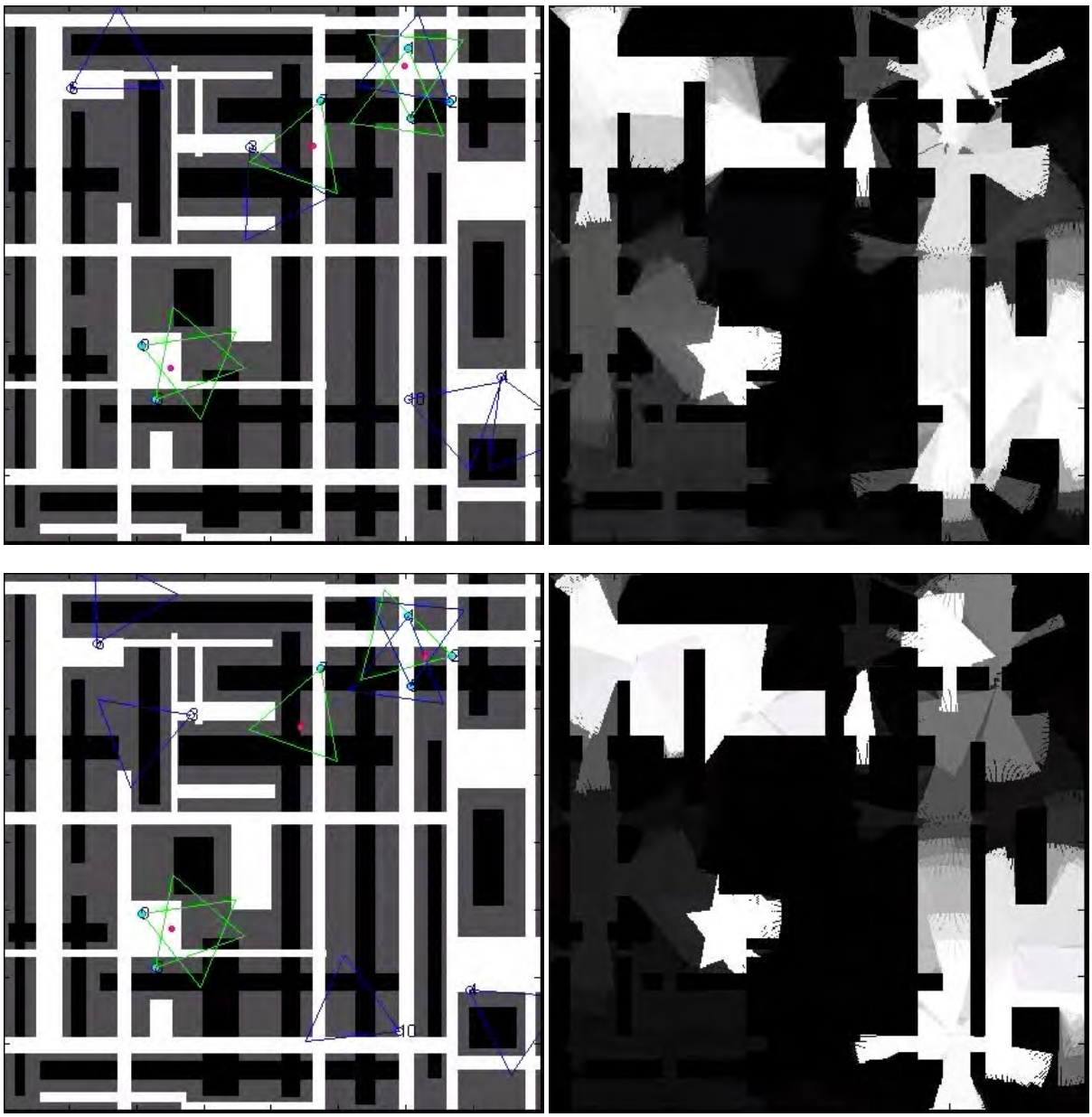


Figure 5: Evolution of the robot map and the integrated coverage map over the course of an urban surveillance experiment

6. Conclusions

In this work, we have presented a decentralized control strategy for multi-robot coordination for urban surveillance. Using the presented multi-agent control architecture, it is possible to make teams of robots execute a well-defined task in a challenging environment. To achieve this, a behavior based framework was implemented. In this context, a novel multi-robot control strategy was proposed, which considers the optimal placement of the robotic team members to provide an optimal field-of-view coverage for intruder detection, through the information maximization approach. The algorithm presented here for multi-robot coordination was shown to achieve good results and to scale well with increasing the number of robots. The implementation of the presented technologies on real-world robots provides interesting applications in the domain of risky interventions.

Acknowledgements

This research was funded by the Networked Multi-Robot Simulation (NMRS) project. The NMRS research project 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).

References

[1] E. Folgado , M. Rincón , J. R. Álvarez and J. Mira, A Multi-robot Surveillance System Simulated in Gazebo, LNCS, Nature Inspired Problem-Solving Methods in Knowledge Engineering, Vol. 4528, page 202-211, 2007.

[2] T. Fong, C. Thorpe, and C. Baur, Multi-Robot Remote Driving With Collaborative Control, IEEE Transactions on Industrial Electronics, Vol. 50, No. 4, August 2003.

[3] K. Madhava Krishna and Henry Hexmoor, Social Control of a Group of Collaborating Multi-robot Multi-target Tracking Agents, IEEE, AIAA 22nd Digital Avionics Systems Conference, Indianapolis, USA, Oct. 12-16, 2003.

[4] P. Pirjanian, Multiple Objective Action Selection & Behavior Fusion using Voting, PhD thesis, Dept. of Medical Informatics and Image Analysis, Aalborg University, Denmark, August 1998.

[5] Y. Zhang, Y. Meng, Dynamic multi-robot task allocation for intruder detection. ICIA '09, pp 1081-1086, Macau, 2009.