sensors

MDPI

Article

Simultaneous Deployment and Tracking Multi-Robot Strategies with Connectivity Maintenance

Javier Tardós, Rosario Aragues *, Carlos Sagüés and Carlos Rubio

Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, 50018 Zaragoza, Spain; jtardos@outlook.com (J.T.); csagues@unizar.es (C.S.); carlosrr@unizar.es (C.R.)

* Correspondence: raragues@unizar.es; Tel.: +34-976-76-23-37

Received: 30 January 2018; Accepted: 19 March 2018; Published: 20 March 2018

Abstract: Multi-robot teams composed of ground and aerial vehicles have gained attention during the last few years. We present a scenario where both types of robots must monitor the same area from different view points. In this paper, we propose two Lloyd-based tracking strategies to allow the ground robots (agents) to follow the aerial ones (targets), keeping the connectivity between the agents. The first strategy establishes density functions on the environment so that the targets acquire more importance than other zones, while the second one iteratively modifies the virtual limits of the working area depending on the positions of the targets. We consider the connectivity maintenance due to the fact that coverage tasks tend to spread the agents as much as possible, which is addressed by restricting their motions so that they keep the links of a minimum spanning tree of the communication graph. We provide a thorough parametric study of the performance of the proposed strategies under several simulated scenarios. In addition, the methods are implemented and tested using realistic robotic simulation environments and real experiments.

Keywords: multi-robot; coverage; Lloyd; Voronoi; tracking; connectivity maintenance; minimum spanning tree

1. Introduction

The coordination of autonomous robot teams has been a rising topic of interest during the last few decades. Formation control, surveillance or coverage are some of the most commonly-faced topics. Heterogeneous robot teams are gaining attention for monitoring tasks in dynamic environments. For instance, in [1], a method is presented for computing the localization of a group with both ground and aerial robots, using measurements that are not associated with identified robots. Considering the low battery life of the aerial robots, in [2], a group of them is assisted by a team of ground robots so that they can dock and recharge their batteries. Some authors focus on particular applications such as mobile sensory systems for monitoring environmental variables in greenhouses [3], instead of large sensor networks, or sediment sampling in estuarine mudflats [4] with drilling ground vehicles and aerial imagery robots.

We face the problem of coordinating a heterogeneous team composed of both ground and aerial robots so that they cooperate to monitor an environment. This way, the different view points of the sensors can be used to build a richer representation of the 3D scene. In these scenarios, the design of the coordination strategy is challenging, since several restrictions should be simultaneously satisfied. Ground and aerial robots should observe the same area to build the representation of the scene. In order to cover a greater area, agents should be deployed far from each other. However, the communication should be kept so that they can exchange data and perform the computations. In this framework, we propose a strategy in which the aerial robots act as leaders of the formation. Ground robots, also

referred to as agents, execute different coverage-based algorithms to keep the aerial ones, from now on targets, in their sensing range.

We consider two alternatives for the ground robots, both of them built on classical centroidal Voronoi-based deployment ideas, which is also known as the Lloyd method and has been often used for static deployment. The alternatives proposed in this paper face the adaptation of these ideas for tracking tasks. The first strategy consists of building density functions, also known as importance functions, with focus on the aerial robots. As they move, the density functions are adapted accordingly, making the ground robots track them. All of the ground agents need to know where the density functions are. This strategy has been previously used in the literature as it is immediately extracted from the Lloyd algorithm. In the second strategy, we build virtual boundaries around the regions in which the aerial and ground robots are operating. Ground robots coordinate to evenly deploy over the region enclosed by these time-varying virtual boundaries. This strategy has never been used in the literature, as far as we know. It has the benefit that only the agents next to the boundaries need to be informed about them, and the others will react to their neighbors' positions. For both strategies, we ensure that the ground robots remain connected by making them keep the links of a Minimum-distance Spanning Tree (MST) in the communication graph. This MST is recomputed at each iteration, so that agents have more freedom to move. Our method is centralized, although some of its steps admit a parallel execution.

We provide a thorough parametric study of the performance of the proposed strategies under several simulated scenarios built with MATLAB. We also evaluate the performance of the methods using realistic robotic simulation environments based on Gazebo and ROS (Robot Operating System), as well as using an experimental setup with six low-cost differential-drive robots.

To sum up, the main contributions of this paper are: (i) the proposal of methods for solving the problem of tracking a formation of targets or aerial robots with a team of agents or ground robots, covering the same areas as the targets and keeping the network connected; (ii) an algorithm based on virtual boundaries applicable to dynamic setups, which makes agents deploy in an equally-spaced configuration; (iii) we present extensive validations of our methods in different simulation scenarios and experiments under different setups.

This paper is organized as follows. Section 2 gathers a review of related work in coverage, tracking and connectivity maintenance. In Section 3, we present a base coverage algorithm with connectivity maintenance, using the Lloyd method and MST, respectively. Then, in Section 4, we propose two strategies to adapt the coverage algorithm to tracking tasks. These strategies are tested with MATLAB scenarios in Section 5. In Section 6, we carry out some realistic simulations with Gazebo and ROS. In Section 7, we show the results of the real experiments. Finally, a set of conclusions and future work are given in Section 8.

2. Related Work

As previously stated, we consider tasks of environmental monitoring where a team of ground and aerial robots establishes coordination strategies to cover the same planar region with different view points. During their mission, ground agents must simultaneously satisfy several restrictions. They must follow the aerial robots and adapt dynamically to the positions covered by them. They also have to deploy in their coverage task, keeping the links of the communication network connected. Note that in this paper, we will consider that coverage and monitoring tasks are similar and will refer to them indifferently. In this section, we give an overview of the strategies that are the most related to the situation considered in this paper and discuss works on swarm aggregation, formation abstraction, formation control, containment control and leader-follower tracking. We discuss also some applications of Voronoi and Lloyd methods.

Leader-follower methods [5,6] are consensus-based control laws that make a team of followers track the position of the leader. The aim is that robots remain and finish close to each other and to the leader. Swarm aggregation methods [7,8] make robots move as a single entity through an environment,

while avoiding obstacles. Although they do not collide, agents tend to be quite close to each other. In containment control tasks [9–11], there are multiple stationary or dynamic leaders, which are aware of the global objectives of the task. The remaining agents execute distributed local rules to follow the leaders, so that they converge to the convex hull spanned by the leaders. Another approach is formation abstraction [12]. The idea is to make agents remain inside the boundaries of a shape, e.g., polygonal, which can be modified or bent to adapt to the environment. The application is quite interesting, but ensuring collision avoidance or accurate tracking of the abstraction is usually hard to analyze. Here, we consider a similar idea, but associated with the well-known and widely-used Lloyd method. In this context, it is easy to introduce other ideas such as connectivity maintenance and collision avoidance, as well as uniform distribution on the area. There are also other leader-follower behaviors of higher complexity, as [13], where a team of leaders moves so that they induce a set of periodic trajectories to be followed by the dependent robots. None of these methods is appropriate to deal with the problem we consider in this paper, since they may group too many agents, whereas we would like them to spread as much as possible to cover a larger area as they explore the environment. Moreover, some of the strategies described do not have control of the specific positions of the dependent robots.

Coverage and deployment algorithms are better suited for our purposes. The solutions considering Voronoi partitioning or the Lloyd method are the most common. They give rise to an easy to implement method, which is highly appealing. Voronoi partitioning consists of attaching to every robot the set of points of the working area that are closer to them and is very popular for deployment and coverage tasks. In fact, many researches take advantage of Voronoi ideas. Some examples include [14], where agents detect evaders within their Voronoi partition and pursue them. In [15], the goal is to capture targets randomly distributed in an environment. Each agent detects targets inside its Voronoi cell and estimates their position with a Kalman filter. The Lloyd method uses the Voronoi partitions to deploy the robots in the environment, and they are iteratively moved to the centroid of their Voronoi cell, separating them from each other and spreading over the working area. This method is widely used in the literature. The situation where some areas have more importance and robots have unlimited sensing and communication capabilities can be solved [16–18] by iteratively moving the robots so that they reach centroidal Voronoi configurations. Several variations of this method have been presented to consider, e.g., robots using outdated information about the positions of the other team members [19] or agents that can only sporadically send information to a central base station [20]. The works discussing limited sensing and communication capabilities are more realistic and thus have a higher interest. Circular sensing footprints are included in [21], while [22] focuses on how to learn and adjust the regions associated with each robot depending on their sensing and actuating performances. Lloyd methods also allow the inclusion of obstacle-avoidance by the definition of safety regions that ensure there will be no collisions between the agents [23]. These previous Lloyd-based strategies focus on static environments, while we want to make the agents track the mobile aerial robots. Lloyd coverage methods have been also used to track moving targets. The most commonly-used solution is to assign the density functions describing the importance of the environment to the targets and make them change with time. It has been experimentally validated that the Lloyd method works properly in these cases [24,25]. This is one of the strategies we also adopt in our work. However, in the presence of density functions, agents tend to accumulate nearby the targets. For this reason, we explore an alternative strategy that makes the agents spread uniformly along a varying working region.

A limitation of some multi-robot strategies is that often, there are no guarantees that the network will remain connected as the agents operate. This is an important issue, since robots often must be able to exchange data and communicate with each other in order to coordinate and to successfully fulfill their tasks. In [26], this term is included in the control law governing the global coordination objective. It keeps an accurate estimate of the time-varying Fiedler eigenvector. Depending on the high-level coordination strategy, the connectivity control term will affect its achievement. It is remarkable

that they study these effects both theoretically and experimentally. In [27], there is a control law that enforces connectivity and an additional bounded control term that encapsulates the high-level control law for multi-robot cooperation. In addition, they design a control law that guarantees that the trajectories of the agents remain within a domain. This is done by means of a repulsion vector nearby the boundaries of the domain. A review of other connectivity control methods can be found in [28]. All these control methods often rely on keeping up to date estimates of data that encapsulate global properties.

An alternative to maintain connectivity is to make the agents keep a set of links in the network. The less restrictive method is to build a tree in the graph. Specifically, the most used is the Minimum Spanning Tree (MST), where the links are selected depending on an assigned weight. Note also that as agents move, the weights of the tree branches change, as well, and the last MST becomes obsolete. Therefore, this MST should be periodically recomputed. Several works adopt this method for connectivity maintenance in different situations. A method for continuous-time systems that preserves a set of links is proposed in [29]. They mention that the best structure to be kept would be a spanning tree, but no strategy is discussed for adapting the tree-links as robots move. In [30], a high-level method is presented for managing tree topologies, which can be combined with motion tasks such as coverage, but the connections between the successive trees and the minimum spanning tree of the communication graph have not been established yet. In [31], the authors present a formation control method for unicycle kinematics robots that keeps the connectivity. A triggered method for readjusting the tree at some time instances during deployment tasks is developed in [32]. In [33], the connectivity control takes place in a middle-ware layer to modify the goals of the robots when there is some risk of breaking some constraint. Furthermore, in [34], the goal is to keep the links of a tree, done in this case in a separate layer at constant time intervals. In this paper, we explore a different approach. We make the agents keep the links of the Minimum-distance Spanning Tree (MST) of the communication graph. Therefore, we consider the re-computation of the MST at every time step, as is done in [33,34]. However, in order to alleviate the communication load, a triggering strategy as the one proposed in [31,32] could be used.

As a result, our methods are centralized. However, as acknowledged in [35], some of the assumptions established for graph-based distributed control strategies are difficult to accommodate in realistic systems, and it may be reasonable to relax requirements on distributed methods. Some of the parts of our method are highly parallelizable. For instance, the MST computation could be alleviated by using the distributed algorithm presented in [36]. We will investigate in the future ways to transfer information on density functions and on the virtual boundaries along the robot networks, to get an even more distributed method. Our aim is to get one step closer to a real implementation of the aerial-ground task, so we use a realistic simulation environment, as well as experiments with low-cost differential drive robots.

3. Coverage for Static Deployment

This section describes the basis of the coverage algorithm for a static environment. The ideas described in this section are a compound of the methods used in [16–18,21,32,36,37]. The coverage task is performed using the Lloyd method. It is appropriate to remark here that the algorithm described is implemented only in the ground robots.

3.1. Static Deployment

Considering a planar environment given by a convex polygon $Q \in \mathbb{R}^2$, we want to deploy n agents with positions $p_i \in Q$, for $i = \{1, \ldots, n\}$, in order to minimize the distances between them and any point $q \in Q$. This is known in the literature as a locational optimization problem and here is solved

using the Voronoi regions method. Partitioning Q into n regions, $\mathcal{W} = \{W_1, ..., W_n\}$, the coverage goal is accomplished by minimizing the cost function [16]:

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} f(\|q - p_i\|) \phi(q) \, dq, \tag{1}$$

where $f(\|q - p_i\|)$ describes the performance of the sensing devices of the agents and $\phi(q)$ is a distribution density function that denotes the importance of each point. We assume $\mathcal{W}$ as the Voronoi partition, $\mathcal{V} = \{V_1, ..., V_n\}$, where:

$$V_i = \left\{ q \in Q \mid \|q - p_i\| \leq \left\|q - p_j\right\|, \, \forall j \neq i \right\},$$

i.e., the regions defined by attaching to each node, p_i, their closest points, q. Therefore, the cost function (Equation (1)) to minimize is now:

$$\mathcal{H}(P, \mathcal{V}) = \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|) \phi(q) \, dq. \tag{2}$$

The sensors of the agents observe better nearby points, and considering this fact, their performance function is defined as $f(\|q - p_i\|) = \|q - p_i\|^2$. Recalling some basic quantities associated with a region W and a density function ϕ, the mass M_W, center of mass C_W and polar moment of inertia $J_{W,p}$ are defined as:

$$M_W = \int_W \phi(q) \, dq, \qquad C_W = \frac{1}{M_W} \int_W q \, \phi(q) \, dq, \qquad J_{W,p} = \int_W \|q - p\|^2 \phi(q) \, dq.$$

Considering the parallel axis theorem $J_{W,p} = J_{W,C_W} + M_W \|p - C_W\|^2$, the cost function (Equation (3)) and its gradient (Equation (4)) can be calculated as:

$$\mathcal{H}(P) = \sum_{i=1}^{n} J_{V_i, C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \left\|p_i - C_{V_i}\right\|^2 \tag{3}$$

$$\frac{\partial \mathcal{H}}{\partial p_i}(P) = 2 M_{V_i} \left(p_i - C_{V_i}\right), \tag{4}$$

As done in [16]. Thus, in order to deploy the agents so as to optimize the cost function, they iteratively compute their Voronoi regions and move to their center of mass (Equation (5)). This gradient descent method is commonly known as the Lloyd method.

$$C_{V_i} = \left(\int_{V_i} \phi(q) \, dq \right)^{-1} \int_{V_i} q \, \phi(q) \, dq \tag{5}$$

Note that before the calculation of the Voronoi region of each agent, we need to know which agents are their neighbors. The process we follow to solve this problem is the simple criterion proposed in [37], explained next for the node p_1 (see Figure 1). Assuming that we know the position of all the nodes, p_i, the method starts computing the boundaries of the Voronoi region, V_1, by the nearest neighbor, then the second nearest, etc. After incorporating each one of these closest nodes, the algorithm checks whether there is any other node inside a circle $B(p_1, R)$, where R is a changing radius equal to the maximum distance from p_1 to any point of the region E_1 computed until that iteration. This region is the one composed by drawing parallel lines to the Voronoi boundaries through the corresponding already found neighbors. If there are no longer any more agents inside this circle, all the neighbors have been found. In the example shown, p_2, p_3, p_5 and p_6 are Voronoi neighbors of p_1, while p_4 and p_7 are not.

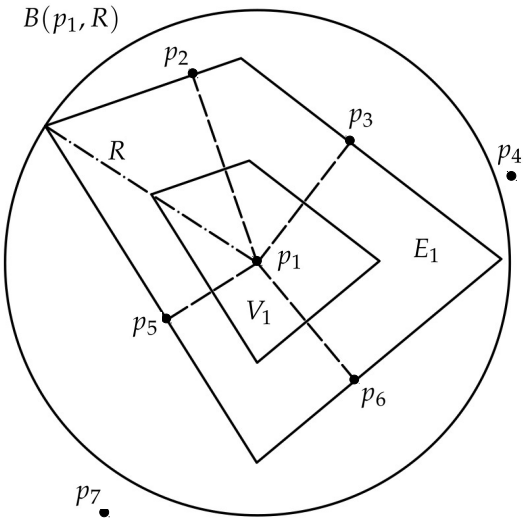

Figure 1. Criterion for selecting the Voronoi neighbors.

3.2. Limited Sensing

The sensors carried by the agents have a limited range, so that they cannot correctly observe objects outside a circle with center p_i and radius, their sensing radius s. Thus, instead of the performance sensing model $f(\|q - p_i\|) = \|q - p_i\|^2$, we have a sensing area limited by s [32].

$$f(\|q - p_i\|) = \begin{cases} \|q - p_i\|^2, & \text{if } \|q - p_i\| < s, \\ s^2, & \text{otherwise.} \end{cases}$$

Therefore, the coverage region of each agent is the intersection between the sensing area of each agent, $B(p_i, s)$, and its corresponding Voronoi partition, V_i. As a result, the cost function associated with this limitation is:

$$\mathcal{H}_s(P) = \sum_{i=1}^{n} \int_{V_i \cap B(p_i,s)} \|q - p\|^2 \phi(q)\, dq. \tag{6}$$

The gradient of $\mathcal{H}_s$ in this case is:

$$\frac{\partial \mathcal{H}}{\partial p_i}(P) = 2M_{V_i \cap B(p_i,s)} \left(p_i - C_{V_i \cap B(p_i,s)} \right), \tag{7}$$

i.e., the gradient descent moves to the centroid, but of the region defined as the intersection between the Voronoi region and the sensing area.

Referring to Proposition 2.7 in [21], a local minimum of this limited cost function (Equation (6)), $P^* = (p_1^*, \ldots, p_n^*)$, is a local minimum of the cost function without limited range interactions (Equation (2)) if $Q \subset \cup_{i \in \{1,\ldots,n\}} B(p_i, s)$. Therefore, this approximation obtains the local optimum solution of the locational problem considering only the area covered by the sensing radius of the agents.

3.3. Limited Communication

For the communication purposes, we consider an r-limited Delaunay graph, as defined in [21], with a communication radius r. In order to be able to run the methods proposed in this section in a distributed fashion, the r-limited Delaunay graph requires an $r \geq 2s$ communication radius, knowing the positions of the agents at a distance of at least $2s$. If our communication radius is smaller, we need to consider an altered cost function in which we will reduce the sensing region. Suppose s^* is our original sensing radius and s the reduced one, so that $2s \leq r$. Following the steps in Proposition 2.5 [21], we can relate their corresponding limited cost functions $\mathcal{H}_{s^*}$ and $\mathcal{H}_s$. Let $f(x) = x^2$ be our performance

function, where $f(0) = 0$. Since we have a limited sensing problem, the performance function f_{s^*} is defined as $f_{s^*}(x) = f(x)$ for $x < s^*$ and $f_{s^*}(x) = f(s^*)$ for $x \geq s^*$. Reducing the sensing radius for our distributed purposes, for $s \in [0, s^*]$, we define the performance function f_s given by $f_s(x) = f(x)$ for $x < s$, and $f_s(x) = f(s)$ for $x \geq s$. Considering that $s^* \geq s$, we also define a constant $\beta = \frac{f(s^*)}{f(s)} \geq 1$, and setting $b = f(s^*)$, then $f_{s^*}(x) \leq b$, $\forall\, x \in [0, s^*]$.

By construction, $f_s(x) \leq f_{s^*}(x)$, $\forall\, x \in [0, s^*]$, so we can conclude that $\mathcal{H}_s(P) \leq \mathcal{H}_{s^*}(P)$. Now, consider the function $\tilde{f}(x) = \beta f_s(x)$. Note that $\tilde{f}(x) = \beta f(x) \geq f_{s^*}(x)$ for $x < s$, and $\tilde{f}(x) = \beta f(s) = b = f_{s^*}(x)$ for $x \geq s$. Therefore, we also can conclude that $\beta\mathcal{H}_s(P) \geq \mathcal{H}_{s^*}(P)$, and then, for all $P \in \{\cup_{i=1}^{n} B(p_i, s^*)\}^n$,

$$\beta\mathcal{H}_s(P) \geq \mathcal{H}_{s^*}(P) \geq \mathcal{H}_s(P) > 0 \tag{8}$$

Thus, as we can see, reducing the sensing radius so that this part of the method can be run on one-hop neighbors gives rise to another cost function, with specific relations to the original one.

3.4. Particular Closed-Form Expressions for Constant Density Functions

When the density function, $\phi(q)$, is defined as a constant, the centers of mass of the coverage regions of each agent coincide with their geometric centroids. Assuming that the coverage regions, W_i, are convex polygons with N_i vertexes labeled as $\{(x_0, y_0), ..., (x_{N_i-1}, y_{N_i-1})\}$, the geometric centroid is calculated as [16]:

$$\begin{aligned} C_{W_i,x} &= \frac{1}{6M_{W_i}} \sum_{k=0}^{N_i-1} (x_k + x_{k+1})(x_k y_{k+1} - x_{k+1} y_k) \\ C_{W_i,y} &= \frac{1}{6M_{W_i}} \sum_{k=0}^{N_i-1} (y_k + y_{k+1})(x_k y_{k+1} - x_{k+1} y_k), \end{aligned} \tag{9}$$

where:

$$M_{W_i} = \frac{1}{2} \sum_{k=0}^{N_i-1} (x_k y_{k+1} - x_{k+1} y_k). \tag{10}$$

4. Simultaneous Deployment and Tracking with Connectivity Maintenance

We introduce an explanation of the connectivity maintenance method, and we give details on the control law. After that, we explain our two alternative methods for simultaneous deployment and tracking.

4.1. Connectivity Maintenance

Regarding the connectivity maintenance problem, we use a Minimum Spanning Tree (MST) [31–34]. This method joins the n agents with $n-1$ links. The nodes are connected in a communication tree that minimizes the sum of the lengths of the links in order to allow the maximum motion freedom. The MST must be recalculated at each step of the simulation. Otherwise, the links would heavily restrict the motion as agents move along. With all these considerations, the process that the algorithm runs at each iteration to calculate the MST in a centralized version is shown in Algorithm 1. A decentralized version can be found in [36], and a triggered strategy is shown in [32].

As mentioned above, the MST is used to ensure that the network remains connected as agents move. Observe Algorithm 2, which includes the main statements in one iteration of the Lloyd algorithm with connectivity maintenance. At every iteration k, each agent i computes the centroid $C_{W_i}(k)$ of its Voronoi region and computes its next position as:

$$p_i(k+1) = p_i(k) + u_i(k), \qquad u_i(k) = K(C_{W_i}(k) - p_i(k)). \tag{11}$$

If there are no restrictions on the agent motion, then $K = 1$, and the next agent position $p_i(k+1)$ is its current centroid, $p_i(k+1) = C_{W_i}(k)$.

Algorithm 1 MST calculation.

Distances$(i,j) = \|p_i - p_j\|$
Communicables$(i,j) =$ Distances$(i,j) \leq r$
MST_Graph $= zeros(nxn)$
MST_Nodes $= 1$
while $length$(MST_Nodes) $\leq (n-1)$ **do**
  for all $i \in$ MST_Nodes **do**
    Candidates $= find$ (Communicables(MST_Nodes$(i),:)$)
    for all $j \in$ Candidates **do**
      if $\exists$ MST_Nodes $=$ Candidates(j) **then**
        MST_Graph(MST_Nodes(i), Candidates$(j)) = 0$
        MST_Graph(Candidates(j), MST_Nodes$(i)) = 0$
      else
        Curr_Distance $=$ Distances(MST_Nodes(i), Candidates(j))
        if Curr_Distance $<$ Best_Distance **then**
          Best_Distance $=$ Curr_Distance
          Best_Candidate $=$ Candidates(j)
          Best_Source $=$ MST_Nodes(i)
        end if
      end if
    end for
  end for
  MST_Nodes $= append$(Best_Candidate)
  MST_Graph(Best_Source, Best_Candidate) $= 1$
  MST_Graph(Best_Candidate, Best_Source) $= 1$
end while

In order to keep the connectivity of the whole system, each agent must be within the communication radius r of its MST-neighbors. However, the motion of the ground robots towards the centroids of their coverage regions may break the communication links, if no additional restrictions are included. At every step, the algorithm has to limit the motion of each pair of nodes (i,j) that are linked in the MST. When next positions $p_i(k+1), p_j(k+1)$ are restricted to a circle with the center at $(p_i(k) + p_j(k))/2$ and radius r, their distance remains within r [38]. Figure 2 shows the maximum movement (red wide line) available in the worst case, where agents need to move in opposite directions.

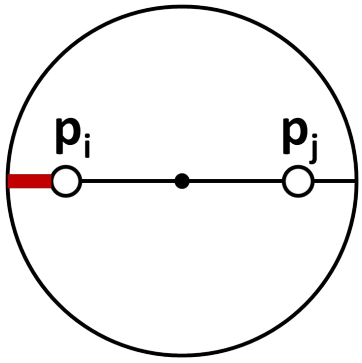

Figure 2. Motion restrictions for linked agents.

Every (i, j) link in the MST that must be kept for agent i can introduce a restriction on its next position $p_i(k+1)$. Agent i adjust its control input gain K in Equation (11) so that $p_i(k+1)$ equals the closest point from $p_i(k)$ to $C_{W_i}(k)$ that satisfies all the restrictions.

Note that agents may temporarily miss MST links as they navigate from $p_i(k)$ to $p_i(k+1)$. However, all these MST links are re-established as all robots get to their goal positions $p_i(k+1)$, which is when they really need the network to be connected since they perform operations that require the exchange of data.

Algorithm 2 Lloyd_MSTConnCtrl.

1: Detect (Neighbors) ▷ [37]
2: Coverage_regions = Voronoi_regions(Neighbors) ∩ Sensing_regions(Neighbors, s) ∩ Q
3: Compute centroid $C_{W_i} = \left(\int_{W_i} \phi(q)\, dq\right)^{-1} \int_{W_i} q\, \phi(q)\, dq$ ▷ Equation (5)
4: Compute (MST_Graph) ▷ Algorithm 1
5: Goal_Positions = Limit(C_{W_i}, MST_Graph) ▷ Section 4.1
6: Navigate (Goal_Positions)
7: Send (New_Positions)

Note that Algorithm 2 generates goal positions for each agent $i = 1, \ldots, n$, as if agents had integrator dynamics in a discrete-time setup. Our methods can be used by agents with other kinematic models. The positions $p_i(k+1)$ generated by our methods are high-level control laws. Agents are equipped with local motion controllers, specific for their kinematic models, to drive them nearby the high-level goals $p_i(k+1)$. When they reach the goals with enough precision, agents inform each other and synchronize to start a new iteration of our algorithms. A detailed discussion of this control strategy can be found in Section 7, for a team of six robot with differential drive kinematics.

4.2. Varying Importance Functions

We propose two alternatives to deal with tracking tasks. The first alternative solves the tracking issue through the density function $\phi(q)$ introduced in the coverage algorithm. This approach has been previously used in the literature to set up a dynamic coverage from the Voronoi tessellation, as it is immediate from the Lloyd static method. Each target is given an importance function defined as:

$$\phi(q) = e^{-\|q-o\|^2}, \tag{12}$$

where o is the position of the target. These values weight the calculation of the centroids, forcing the agents to reach the center of each function, where the targets are placed. Therefore, all the agents must be informed about the positions of all targets. Varying the position of these functions depending on the targets motion, we can achieve a tracking behavior. The ground agents will deploy, covering the same area as the aerial robots.

Figure 3 shows an example of the behavior of this method under fixed targets, in order to note how the agents deploy. Later, in Section 5, we will present more examples with moving targets. Agents are represented with small red circles, and their communication tree is the set of gray lines that link them. The big blue circles represent the coverage regions of each agent. Finally, the grayscale concentric circles are the importance functions (with darker gray for higher values) where the targets are placed. In the first steps, the agents start moving towards the area of interest. When they reach it, they group around the first targets they obtain. Then, they deploy to cover all of them.

The process executed at each iteration of the simulation is gathered in Algorithm 3.

We have observed that this method provides an accumulation of ground robots around the targets, which is not the behavior desired for coverage tasks. In addition, the local optimal behavior of this

method sometimes implies that some targets may be uncovered. For these reasons, we propose another alternative, explained next.

Algorithm 3 Varying importance functions.

1: Detect (o_i)
2: Propagate (o_i)
3: Update $(\phi(q))$
4: Execute (Lloyd_MSTConnCtrl Iteration) ▷ Algorithm 2

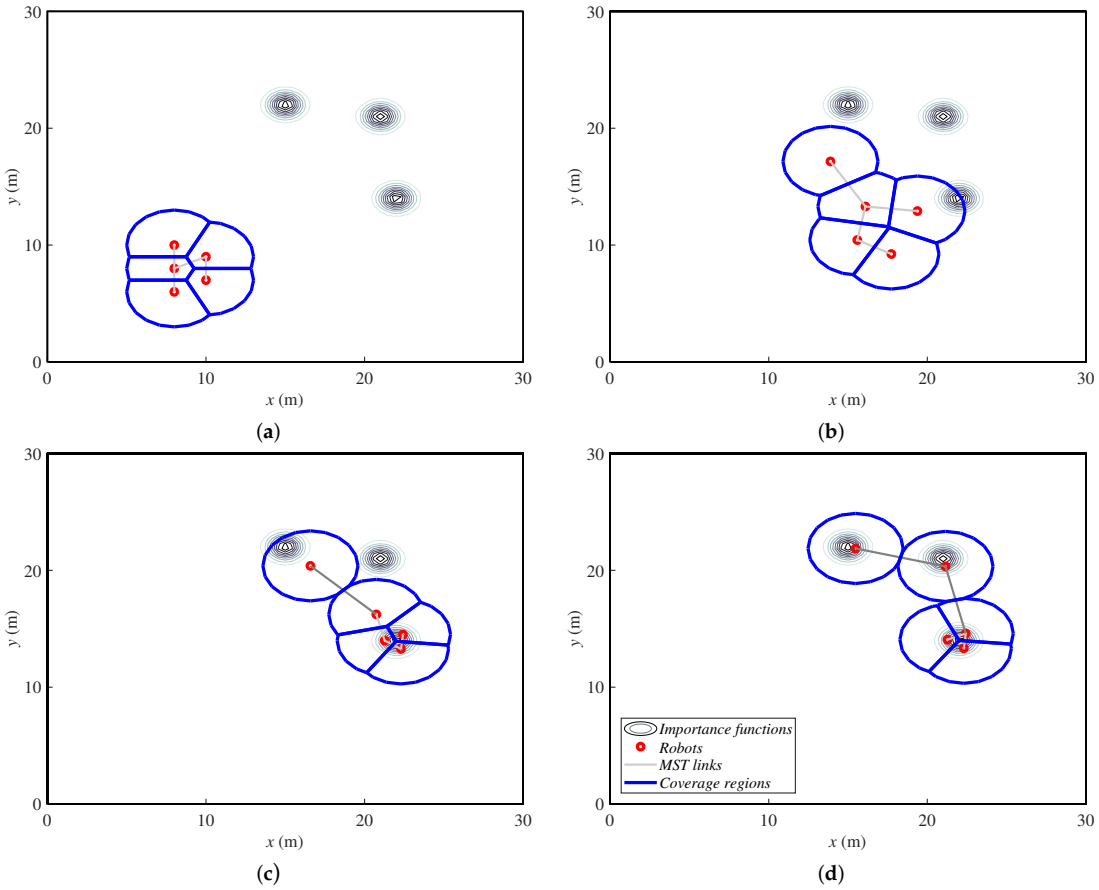

Figure 3. Ground robots' positions in four steps ((**a**) step 1, (**b**) step 6, (**c**) step 15 and (**d**) step 30 of 30) while covering three fixed targets, which are assigned importance functions. MST, Minimum Spanning Tree.

4.3. Redefining Fictitious Boundaries

We propose another alternative based on redefining the boundaries of the working area. Since in coverage problems with constant density functions, agents tend to evenly deploy over the area, here, the idea is to modify in a virtual way the boundaries of this area to make agents adapt accordingly. The interest in this method lies in the even deployment of the agents over the modified working area. In addition, only the agents next to the virtual limits need to be informed about them. As far as we know, this solution has not been used before in the literature.

We decide to define the new area as the minimum rectangle including both targets and agents. As the area reduces, agents deploy with no different importance until the zone is as small as possible. If there are enough agents, they will spread so as to cover all the targets. Therefore, this method

requires the aerial robots, which will act as leaders of the mission, to move in ways that can be covered by the ground robots. This restriction depends on some factors like the number of agents or their sensing radius and will be studied in Section 5.

Figure 4 shows the same example as the previous alternative. Here, the redefined fictitious boundaries are the green lines, and the targets are represented just with small gray circles. In the first steps, the agents move towards the targets while the rectangle reduces. Once the rectangle is as small as the targets allow, the agents spread over the final area. At the end, they cover a greater zone than the previous alternative, deploying uniformly. One can notice that the agents must have a fixed bearing reference in order to identify the orientation of the rectangle that iteratively defines the area to cover. In this paper, we make the assumption that they have this information. In real implementations, a mechanism for achieving this should be included. For instance, agents can be equipped with a compass or run a localization method, as in [1].

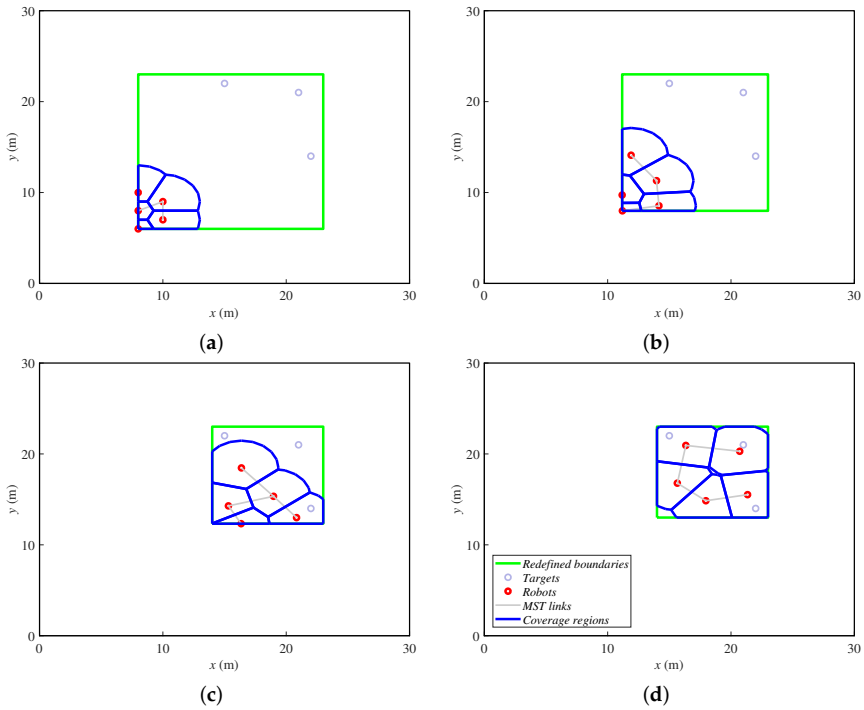

Figure 4. Ground robots' positions in four steps ((**a**) step 1, (**b**) step 6, (**c**) step 15 and (**d**) step 30 of 30) while covering three fixed targets delimited by fictitious boundaries.

The process that this alternative runs in a centralized way at each simulation step to follow the targets is shown in Algorithm 4.

Algorithm 4 Redefining fictitious boundaries.

1: Detect (o_i)
2: $x_{min,Area} = min(p_{i,x}, o_{i,x})$
3: $x_{max,Area} = max(p_{i,x}, o_{i,x})$
4: $y_{min,Area} = min(p_{i,y}, o_{i,y})$
5: $y_{max,Area} = max(p_{i,y}, o_{i,y})$
6: Area $= [x_{min,Area}, x_{max,Area}, y_{min,Area}, y_{max,Area}]$
7: $Q =$ Area
8: Execute (Lloyd_MSTConnCtrl Iteration) ▷ Algorithm 2

4.4. Comparison between Both Alternatives

The example studied allows making a comparison between the proposed alternatives. In addition to the previously-discussed differences between the amount of agents that need to be informed of the positions of the moving aerial robots, in this section, we explain some behavioral differences.

On the one hand, the importance function alternatives tend to group the ground robots around the targets. Depending on the initial relative positions, this could make them leave some others uncovered. On the other hand, the method redefining the boundaries achieves a wider formation, covering a greater zone than the first alternative. Therefore, this method focuses on the area around the targets, while the first method focuses on the targets themselves.

Considering this information, the importance function method is better suited to individual objectives whose environment is not important, while the redefining boundaries alternative is more appropriate for concentrations of targets or wide areas of interest.

5. Simulations and Results

Both alternatives presented so far are built on deployment methods, which were originally designed for static setups. Since our aim is to use these methods to dynamically track variations in the region to be covered, a question that arises is how slow the modifications in the environment must be in order for the agents to efficiently track them. This question is hard to answer from an analytical point of view. Moreover, the speed may depend on several facts (number of agents, area to be covered, sensing radius, communication radius, etc.). In this section, we make a thorough parametric study to identify the factors that influence the allowable speed. Here, the simulations are only executed with MATLAB in order to get repeatable results, abstracting from implementation issues carried by 3D simulation or real experiments. Later, in Sections 6 and 7, we carry out 3D simulations and experiments that involve other specific problems (robot kinematics, collision avoidance, noisy data or restrictions on the motions).

We define now the base experiment for the parametric study. We have a group of six ground robots implementing the redefining boundaries algorithm. We will consider the particular case where the targets fly in a rectangular formation composed of twelve aerial robots and move 0.3 m per simulation step. The sensing radius of the ground robots is $s = 3$ m, and their communication radius is $r = 6$ m.

Starting from this experiment, we are varying four parameters in order to study how they influence the algorithm behavior. These four parameters are the velocity of the formation to track, the sensing radius, the number of agents and the tracking method.

5.1. Formation Velocity

In the first study, we vary the formation velocity in the interval 0.25–0.5 m/step. The formation of targets moves from left to right, and its initial position is described in Figure 5, as well as the initial position of the ground agents.

The velocity is different in each case, and we run 60 steps of the method in all of the setups. Thus, the final position of the formation will be different. In Figure 6, we show the final dispositions of all the robots for 0.25 and 0.5 m/step cases. For lower velocities, the algorithm is able to keep the formation covered as it moves. However, if velocity increases, the agents cannot correctly cover the whole formation. Some of them agglomerate in the rear formation, leaving the front targets uncovered.

Now, in Figure 7a, we represent the cost functions (Equation (1)) for each velocity along the simulation steps. One can realize that the cost function stabilizes at a constant value for each velocity. This behavior means that in all cases, the agents spread to a constant disposition and reach the speed of the formation, advancing with it. Furthermore, the metric minimizes for the lowest speeds, because the agents cover the whole formation of the targets, as we see in Figure 6.

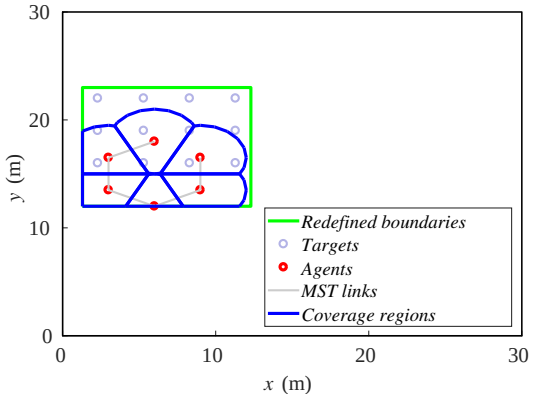

Figure 5. Initial positions of agents and targets.

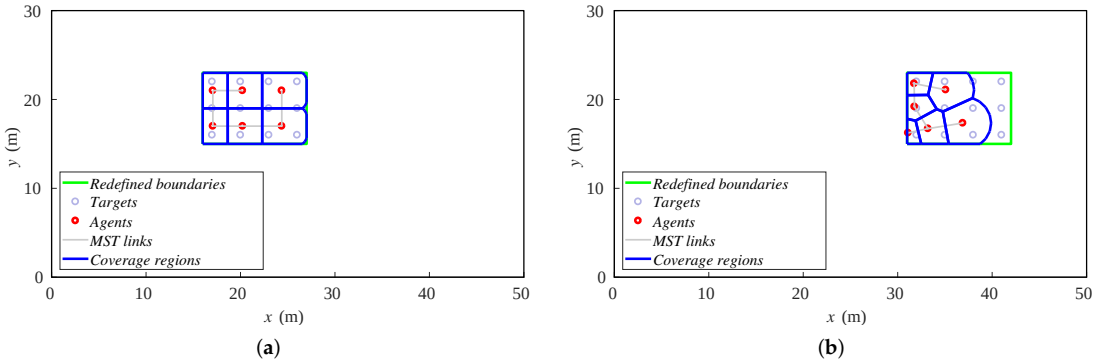

Figure 6. Final positions of agents and targets for (**a**) 0.25 m/step and (**b**) 0.5 m/step.

The behavior of the algorithm can also be monitored with a variable defined as:

$$\sum_{i=1}^{n} \|p_i - O\|, \tag{13}$$

i.e., the sum of the distances of each ground robot to the center of the formation, O. This variable can describe the quality of the coverage performed. Figure 7b gathers this sum of distances along the simulation with the velocities studied. We observe the same behavior of the curves as in the cost function. In the case of higher velocities, the sum of distances increases since the agents cannot cover the whole formation. The metric minimizes also for lower velocities. For this reason, from now on, we will show this performance metric (Equation (13)), since it is easier to understand and its behavior stays similar to the cost function in the experiments remaining.

The results obtained evidence of the existence of a maximum formation speed that the algorithm can correctly track. In [18], the authors prove that the static coverage task for a one-dimensional case, with $\phi(q) = 1$ and a relative precision ε, is achieved in a time that polynomially depends on the number of agents. Our case is dynamic and two-dimensional, so this study would imply a much higher complexity. We obtain the maximum speed experimentally instead. Taking a look at the graphs, we conclude that this maximum velocity is around 0.3 m/step for the described base setup. That is why the base experiment is designed with this formation velocity, in order to highlight the differences in the following studies.

Note that the velocity units are m/step. This will allow a higher maximum speed of the targets for systems that execute a step of the algorithm more quickly. This includes, e.g., the communication between the agents, the computational load or the motion restrictions of the ground robots.

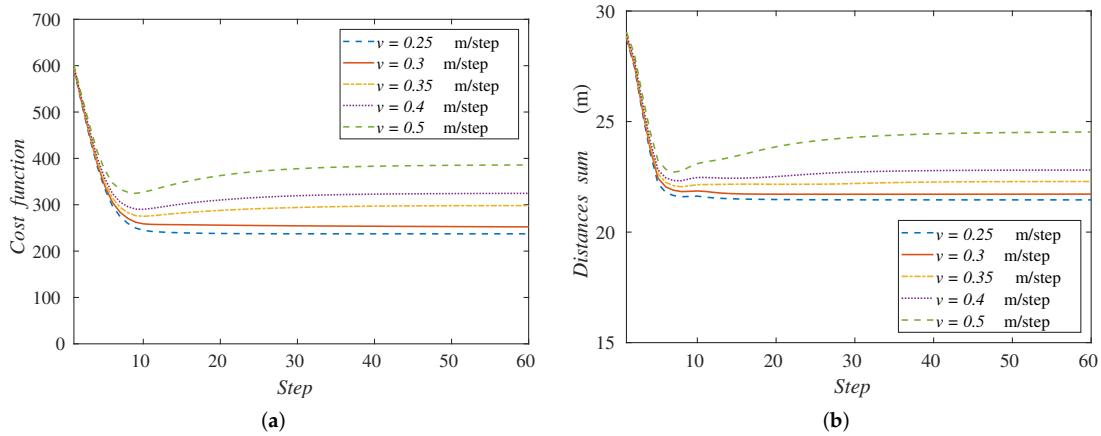

Figure 7. Performance metrics: (**a**) cost function and (**b**) sum of agents' distances to the formation center, along the simulation depending on the formation velocity.

5.2. Sensing Radius

In this case, we test five scenarios with the sensing radius in the interval 2–4 m. In order to maintain the relation $s = r/2$ discussed in Section 3, the communication radius is changed proportionally in the simulations. Initial positions are the same as in the previous experiments, and the simulations are run with 80 steps, so that every experiment reaches the steady state.

The sensing radius is often one of the parameters that affects the allowable speed of the formation to be tracked. Showing the final positions of the robots for 2 and 3.5 m (Figure 8), one can conclude that, with a lower sensing radius, the agents cannot perform a wide deployment to cover the targets, and they accumulate at the rear formation.

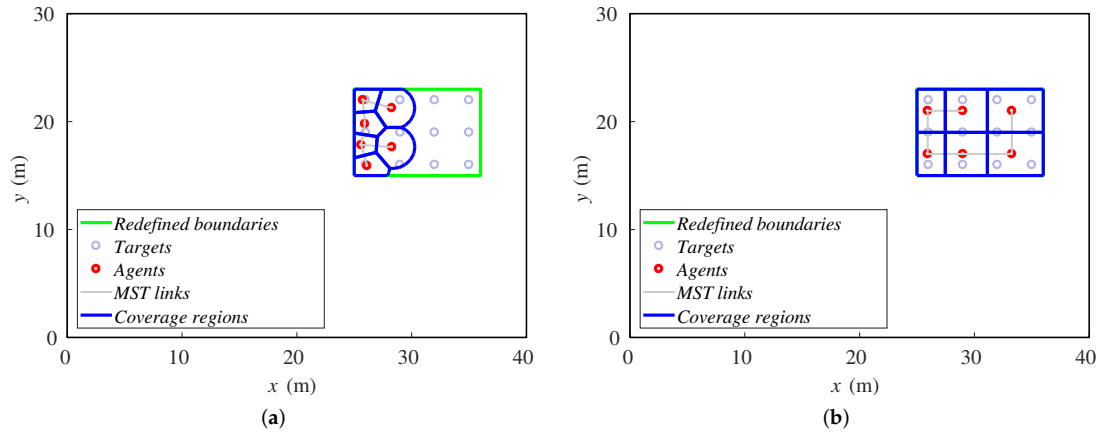

Figure 8. Final positions of agents and targets for (**a**) 2 m and (**b**) 3.5 m.

Figure 9 represents the sum of the distances from the ground agents to the formation center for each sensing radius. As we said before, the smaller radius clearly makes this variable increase because the agents go to the back of the formation. On the other hand, while the radius increases, we can observe that the curves tend to the same value. This behavior is due to the fact that the agents have an equilibrium distribution that balances their advance with the formation speed. Even though the radius is much higher, the balance position would be the same.

From the results, we can observe that as the sensing radius of agents decreases, they have more difficulties following the moving formation. One of the possible reasons for this behavior is that, at every step, agents perform motions that keep them inside the intersection between their own Voronoi

and sensing region. When this region is smaller, agents perform motions that may be too short and may prevent them from tracking the aerial formation.

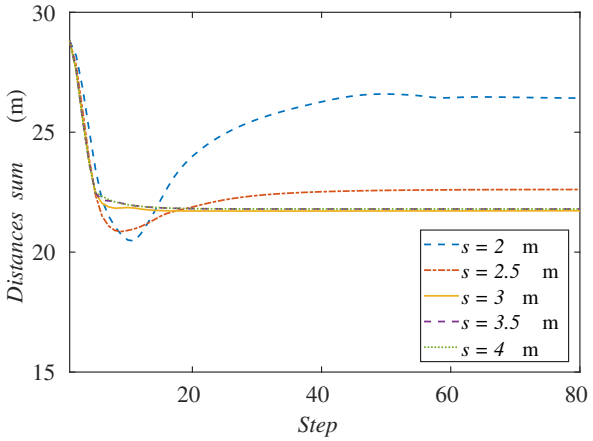

Figure 9. Sum of agents' distances to the formation center along the simulation depending on the sensing radius.

Another possible explanation would be that the communication links were too restrictive for the motion of the agents. Recall that with each variation of the sensing radius, we have changed the communication radius as well in order to keep the relation $r = 2$ s. If the communication radius were higher, one could think that the agents may expand over the region and cover all the targets. For this reason, we have run another set of simulations with a larger communication radius, keeping a sensing radius of 2.5 m, which in the previous experiment could not cover all the formation. In Figure 10, we can see the results of the sum of distances to the center of the formation varying the communication radius from 5–10 m. We can observe that all the graphs for each communication radius concur in the same curve. Therefore, we can conclude that this is not the reason why the agents cannot follow the formation with a lower sensing radius.

These results also allows us to think that the connectivity maintenance method is not interfering with the coverage and tracking goal, as far as the region can be indeed covered by the agents.

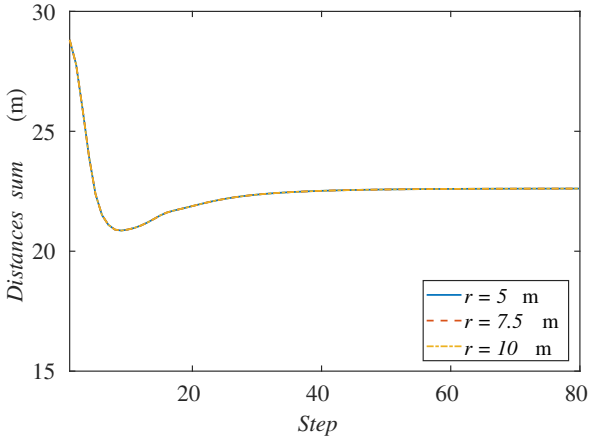

Figure 10. Sum of agents' distances to the formation center along the simulation depending on the communication radius.

5.3. Number of Agents

This time, we vary the number of ground robots in the system, duplicating them iteratively in order to highlight the differences in the experiments. In this case, we test from 4–16 agents. The initial

positions cannot be the same because the number of ground robots is variable. The agents start from positions that are close to the previous experiments and run a 300-step simulation in order to be able to observe the behavior of the curve with the highest number of agents.

Due to the fact that the number of agents is variable, now the sum of the agents' distances to the formation center is not an appropriate variable to represent. We graph the mean distance instead in Figure 11. The curves do not start from the same point as in the previous experiments because of this same reason.

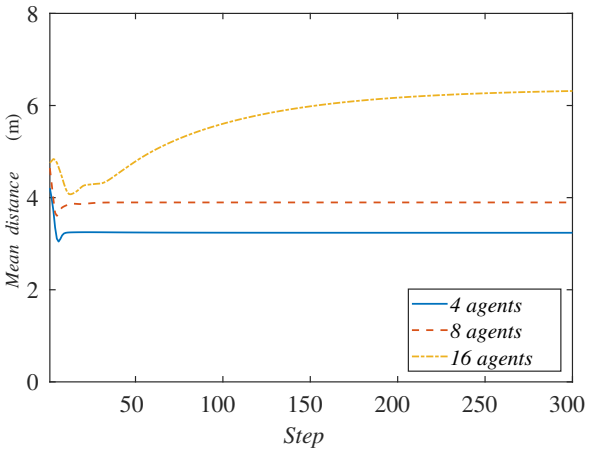

Figure 11. Mean agent distance to the formation center along the simulation depending on the number of agents.

Contrary to what one could have expected, we observe that, with a higher amount of agents, the method does not perform better, even though they have greater capabilities to cover an area. The mean distance to the center of the formation increases as the number of agents is higher. Observing Figure 12, one can see the reason for this behavior. When there are more ground robots than needed, the extra agents tend to accumulate at the rear formation. In case this excess is extreme, the agents are very close and interfere with each other, making their available space to move too short. This prevents them from advance with the formation of targets, and their mean distance to the center of the formation increases.

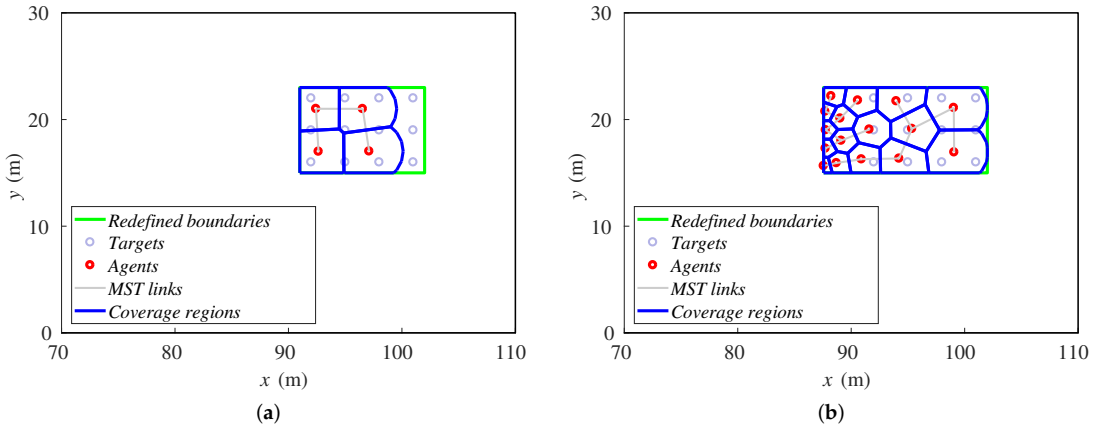

Figure 12. Final positions of agents and targets for (**a**) four robots and (**b**) 16 robots.

The agents that lead the motion of the team are those that have direct contact with a boundary perpendicular to the direction of the target formation, i.e., the ones that are at the front and rear. They have a direct modification of their Voronoi and sensing region, implying a quicker reaction than the agents that are in the middle part of the formation. Therefore, the accumulation of unnecessary

ground robots increases the number of agents in the center of the formation, obstructing the motion of the whole team. In the case that many ground robots were needed, the exploration strategy of the aerial team should consider a limit speed appropriate for the features of the system.

5.4. Tracking Method

Finally, we compare the behavior of the algorithm on the base experiment varying the particular method for the tracking task between that proposed in this paper. The initial positions of the agents are again the positions shown in Figure 5 in both cases.

After a 30-step simulation, the final positions of the agents are shown in Figure 13. The distributions of the agents over the formation are very similar. Both alternatives cover the twelve targets when the steady state is reached.

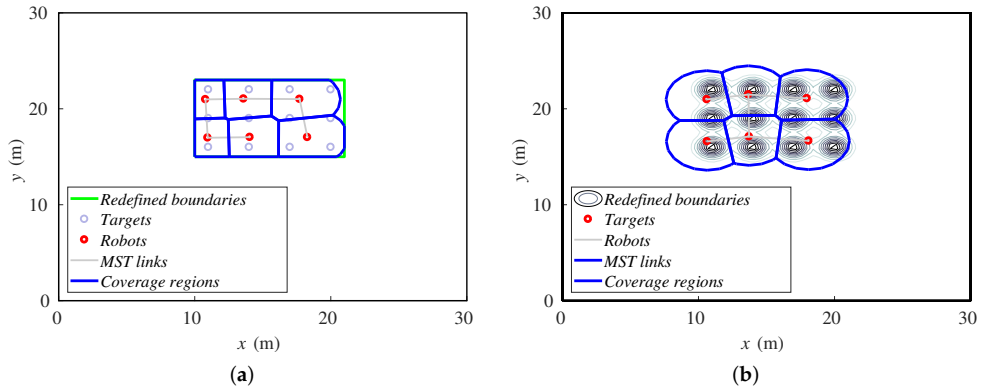

Figure 13. Final positions of agents and targets for (**a**) redefining boundaries and (**b**) importance functions methods.

Representing the sum of the agents' distances to the center of the formation (Figure 14), we can see that the value of the importance functions method quickly decreases, so its deployment is faster and its response time lower. On the other hand, the redefining boundaries method achieves a lower value, so the agents in this case are a bit better deployed.

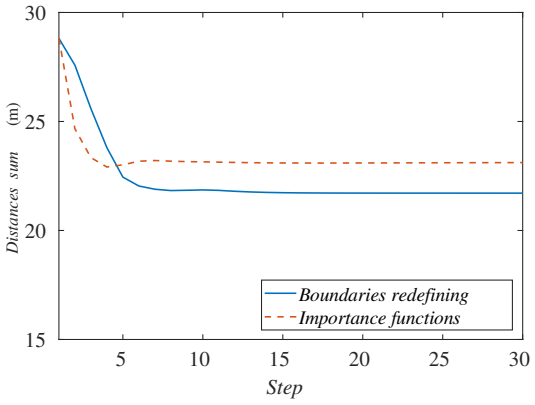

Figure 14. Sum of agents' distances to the formation center along the simulation depending on the tracking method.

We can conclude then that both methods are suitable for this experiment, but the one redefining the boundaries leads to a better deployment. In addition, as we saw in Section 4, this alternative is more appropriate for target agglomerations like the ones considered in the scenario discussed in this section.

6. Realistic Simulations

With the aim of getting one step closer to a real implementation of the methods proposed in Section 4, we have implemented and tested our methods using a realistic 3D physics simulator specialized in robotics: Gazebo. We have used models associated with real robots (Turtlebots) which have a differential drive behavior. They are controlled with ROS through specific packages that include the robot description, the robot controller or the navigation stack. Configuring and connecting the set of packages, we are able to send goal positions to the ground agents once we compute the coverage algorithm. Iteratively, Turtlebots execute Algorithms 3 or 4 described in Section 4. Between iterations, they navigate to their goal positions using local lower level controllers, taking into account their own motion kinematics restrictions. Since in the higher level methods described in Sections 3 and 4, we have not included any collision-avoidance method, we incorporate a reactive local method in the Gazebo implementation so that agents do not collide as they navigate.

Lloyd methods, as other distributed strategies, make some assumptions that make the problem more tractable. In particular, in our simulations, we have made the following assumptions:

- Scenarios are obstacle-free. Robots only need to avoid collisions with other robots, but not with other objects in the environment.
- Ground robots can exchange data in one hop with robots within their communication radius.
- Ground robots share a common orientation, as required by Algorithm 2.
- Targets receive data from the ground robots to know that they have reached their goal destinations. They wait for this confirmation before performing their next motion.
- Ground robots can also localize themselves, localize their neighbors and get information of the positions of the targets (here, aerial robots), by direct measurement or by multi-hop messages. We use the noise-free ground-truth data given by Gazebo.

We have designed four obstacle-free scenarios in order to examine the behavior of the algorithm in some common situations and some extreme cases where it might fail. The reader can find in [39] four videos with the names "*case_i*", where *i* is a number from 1–4. The videos represent the simulations carried out in this section, in order of appearance. They contain the simulation in Gazebo and the graphs obtained from MATLAB. The first one is a simple expansion without targets or zones more important than others. This is just a coverage task, and we do not need to implement any of the tracking alternatives proposed. Figure 15 shows the final layout of the ground agents. We have developed a visualization system that helps to understand the movement of each agent in the Gazebo simulation. The black ground robots are the Turtlebots, which are the ones actually running the algorithms. We add some elements like the targets, implemented as aerial robots that are moved discretely. The colored spheres represent the positions of the ground robots and their goal positions in the current simulation step. Additionally, the gray cylinders represent the communication tree reached at the present step.

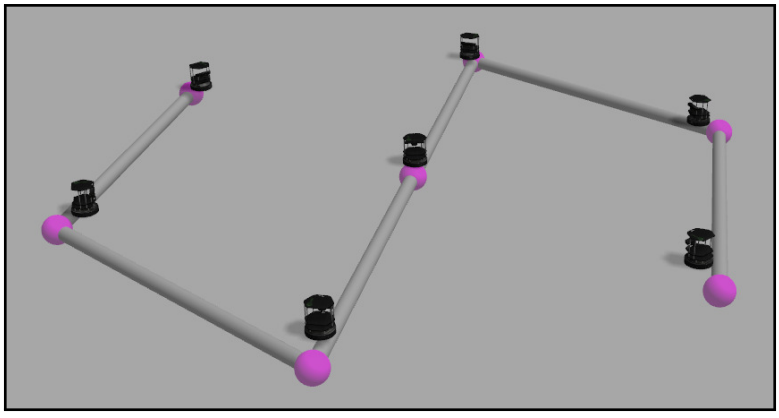

Figure 15. Final layout in an expansion without targets.

The second experiment consists of five agents implementing the redefining boundaries method covering and tracking a formation of ten quadrotors that fly in a triangular formation. The final disposition of both ground and aerial robots is shown in Figure 16.

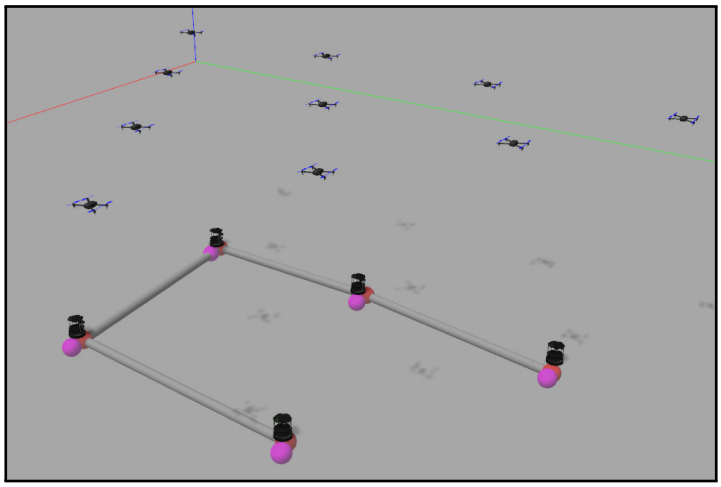

Figure 16. Final layout in a formation-tracking experiment.

The third and fourth experiments implement the method with the importance functions to track the targets. In the third case, the targets start together and separate into opposite directions. The ground team spreads in a straight line until the point that the communication tree allows them. Figure 17 shows this final straight formation of the agents.

Finally, in the fourth case, two targets describe curve trajectories that cross in the center of the working area. In the first part of the experiment, the ground team covers the aerial targets and tracks them. During the crossing, the ground agents get too close. Their collision avoidance method works well and prevents them from crashing. However, their redeployment can not be correctly performed because of the agglomeration produced, finally leaving one of the targets uncovered. Figure 18 shows this fact at the end of the simulation.

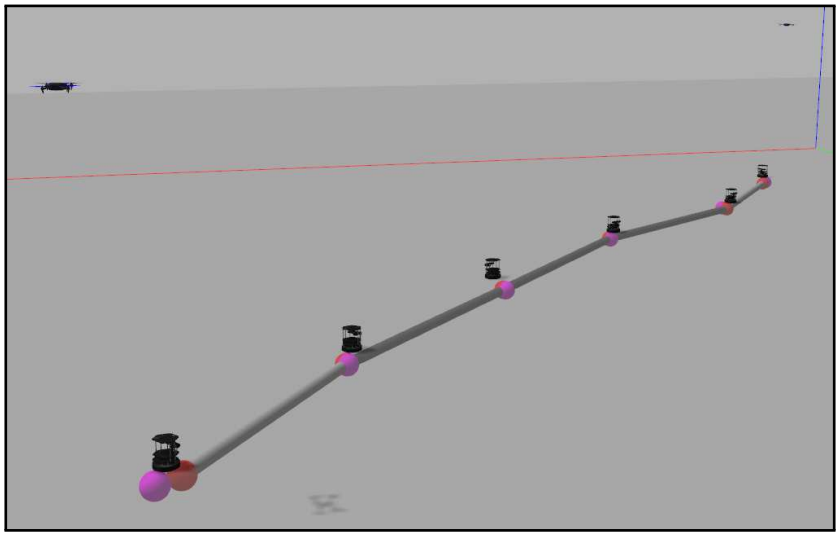

Figure 17. Final layout in an experiment with two targets separating into opposite directions.

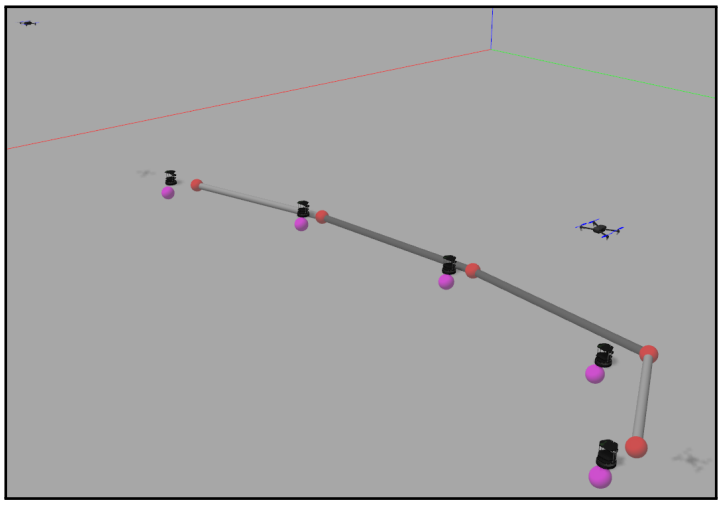

Figure 18. Final layout in an experiment with two targets following crossed trajectories.

7. Experimental Results

In addition to the simulations executed in Sections 5 and 6, we present a real robot experiment in order to analyze the algorithm performance in a more realistic environment. We test the redefining boundaries method as it is the major contribution of this article.

The experimental tests were carried out with the setup shown in Figure 19. Six differential-drive robots act as agents, and each of them includes a Raspberry Pi 2 Model B and a driver controller. The localization of each robot is achieved using aRuCo [40] markers together with a webcam, Logitech C310. Therefore the source of the localization is noisy instead of the ideally precise ground-truth data used in the Gazebo simulation in Section 6. Moreover, motions of the robot wheels are affected by real actuation restrictions and by occasional wheel slips. The robots are controlled by a goal-to-goal controller programmed in MATLAB and a wheel velocity controller running inside each Raspberry Pi with a Python code reading the encoders placed on each wheel and controlling the speed. Commands are sent to the robots by a TCP/IP server opened from MATLAB to each Raspberry Pi, and the Python code sends the voltage messages to the driver controller, which powers the motors.

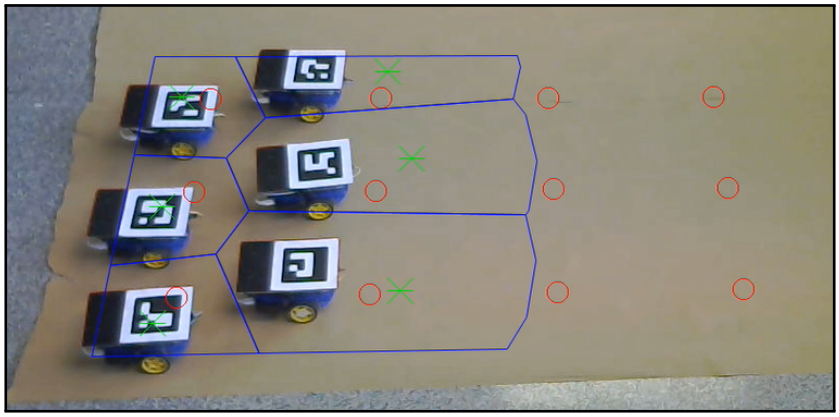

Figure 19. Initial configuration of the experiment setup.

In Figure 20, a frame of the experiment conducted can be observed. The video recording this experiment can be found at [39] with the name "*experiment*". The robots perform the role of agents, and the moving targets that should be covered are plotted with red circles. The movement of the robots reaches the green asterisks representing the centroids of each coverage region, which are drawn in

blue lines. The sensing radius of the agents is set to $s = 500$ mm. All of these elements are plotted from the algorithm by homography in each frame of the video.

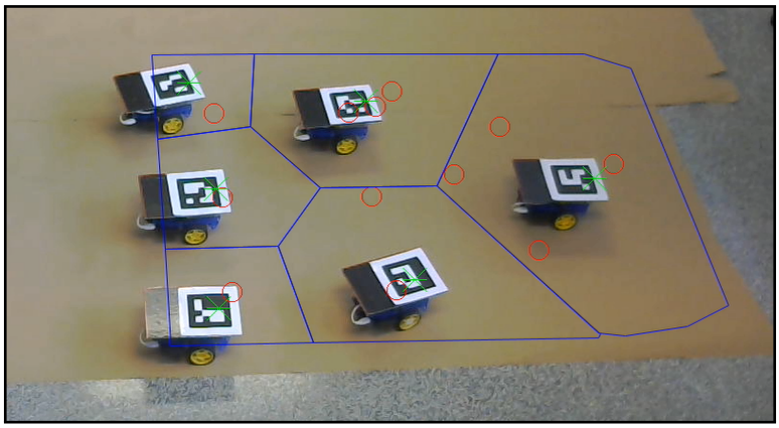

Figure 20. Setup configuration in a certain step of the experiment.

The movement designed for the targets is composed of two straight sections joined by a semicircle, scheduled beforehand in a three by four formation. The formation speed is $v = 35$ mm/step, much lower than in the simulations. The movement set to the external targets during the curve is scheduled in order to maintain their linear velocity during the curve trajectory. These external agents perform a 750-mm radius curve, while the radius of the internal is 150 mm. The collision between the agents is avoided by narrowing the Voronoi regions by a safety radius of $r_{saf} = 75$ mm.

The control law scheme used in the experiment is the one described in Section 4.1: agents receive goal positions, generated according to Equation (11) (green asterisks). Note the goal positions at each iteration are the result of computing the centroids of the Voronoi-regions and adjusting them so as to keep the communication links, as described in Sections 3 and 4. Note also that the low-cost robots in the experiment have differential-drive kinematics. Thus, agents run their low level controllers to get close to the high-level goal positions. When the agents are near enough to their goal (we use a distance tolerance $d = 65$ mm in our experiments), they stop moving and wait for the remaining agents to finish navigating to their goals. After that, all the agents start a new iteration of Algorithm 4, which includes the re-computation of Voronoi regions.

Figure 21 represents the real trajectories followed by the different agents (continuous line) against the set of consecutive goal positions (Equation (11)) derived from the algorithm (dotted line).

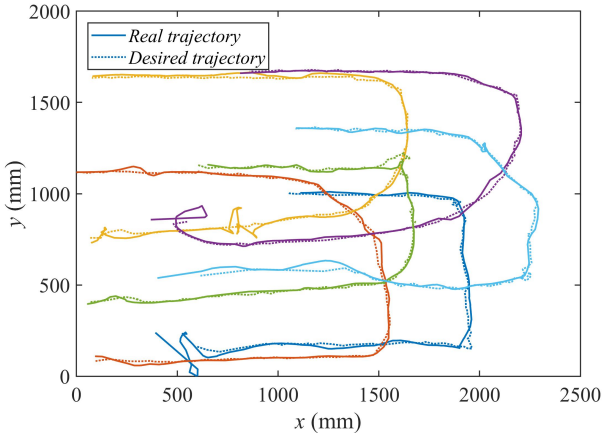

Figure 21. X-Y representation of the consecutive goal positions and the real trajectories of the agents.

In Figure 22, we represent the evolution of the X coordinate of the trajectories and the goal positions along the time of the experiment.

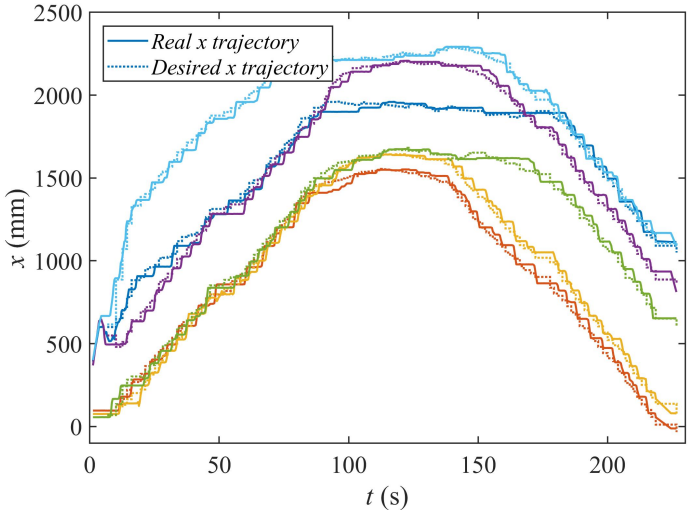

Figure 22. T-X representation of the consecutive goal positions and the real trajectories of the agents.

Observe how the desired high-level goal positions (Equation (11)) remain unchanged for some time. There, the low-level controllers of the agents drive them from their current positions to these new goals. Thus, as can be seen, Algorithms 3 and 4 proposed in this paper can be run by agents with different kinematic models, as long as they are equipped with local controllers that drive them to the high level goal positions generated by Equation (11).

8. Conclusions

We propose and test two strategies to solve the problem of simultaneous coverage and tracking, taking into account sensing and communication limitations, while ensuring that the network remains connected. Both strategies are based on Lloyd methods for static deployment. The first consists of modifying the positions of density functions according to the movement of the targets. This solution has been commonly used in the literature for adapting the static algorithm to tracking tasks. The second strategy iteratively modifies the boundaries of the working area in order to follow the targets while covering their surroundings. As far as we know, this is a novel solution for the problem addressed in this paper.

The density functions solution results in reacting faster to the targets' motion. However, depending on the initial disposition of the agents, some targets may end up uncovered, while the agents concentrate around other targets. In addition, all of the agents need to know the current positions of the targets in order to compute the algorithm.

On the other hand, the redefining boundaries solution has a slower reaction to the targets' motion, but achieves an even deployment over the area of interest. Furthermore, the information about the position of the targets is transformed into the position of the boundaries, and this information is only needed by the agents that have contact with them. Therefore, the communication load is lighter when implementing many agents. A parametric study of the second method shows the existence of a maximum velocity of the targets that it can follow. We have shown that this limit speed depends on the number of agents and their sensing radius and does not depend on the communication radius. We have implemented and validated the system on a realistic simulator. We have carried out real experiments with a team of six low-cost robots. We have shown that the methods can cope with measurement noises and can be combined with several kinematic robot models.

Acknowledgments: Supported by the Spanish Ministerio de Economía y Competitividad DPI2015-69376-R and Vicerrectorado de Política Científica, Universidad de Zaragoza JIUZ-2017-TEC-01, projects.

Author Contributions: J.T. developed the tracking methods and designed the simulations; R.A. reviewed previous work and stated the static coverage algorithm with connectivity maintenance; C.S. coordinated and revised all the article content; C.R. brought up the real robot experiment setup.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stegagno, P.; Cognetti, M.; Oriolo, G.; Bulthoff, H.H.; Franchi, A. Ground and Aerial Mutual Localization Using Anonymous Relative-Bearing Measurements. *IEEE Trans. Robot.* **2016**, *32*, 1133–1151.
2. Mathew, N.; Smith, S.L.; Waslander, S.L. Multirobot Rendezvous Planning for Recharging in Persistent Tasks. *IEEE Trans. Robot.* **2015**, *31*, 128–142.
3. Roldán, J.J.; Garcia-Aunon, P.; Garzón, M.; de León, J.; del Cerro, J.; Barrientos, A. Heterogeneous Multi-Robot System for Mapping Environmental Variables of Greenhouses. *Sensors* **2016**, *16*, 1018.
4. Deusdado, P.; Guedes, M.; Silva, A.; Marques, F.; Pinto, E.; Rodrigues, P.; Lourenço, A.; Mendonça, R.; Santana, P.; Corisco, J.; et al. Sediment Sampling in Estuarine Mudflats with an Aerial-Ground Robotic Team. *Sensors* **2016**, *16*, 1461.
5. Cao, Y.; Ren, W.; Li, Y. Distributed discrete-time coordinated tracking with a time-varying reference state and limited communication. *Automatica* **2009**, *45*, 1299–1305.
6. Zhao, Y.; Duan, Z.; Wen, G.; Chen, G. Distributed finite-time tracking of multiple non-identical second-order nonlinear systems with settling time estimation. *Automatica* **2016**, *64*, 86–93.
7. Dimarogonas, D.V.; Kyriakopoulos, K.J. Connectedness Preserving Distributed Swarm Aggregation for Multiple Kinematic Robots. *IEEE Trans. Robot.* **2008**, *24*, 1213–1223.
8. Leccese, A.; Gasparri, A.; Priolo, A.; Oriolo, G.; Ulivi, G. A Swarm Aggregation Algorithm based on Local Interaction with Actuator Saturations and Integrated Obstacle Avoidance. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 1865–1870.
9. Cao, Y.; Ren, W.; Egerstedt, M. Distributed containment control with multiple stationary or dynamic leaders in fixed and switching directed networks. *Automatica* **2012**, *48*, 1586–1597.
10. Kan, Z.; Shea, J.M.; Dixon, W.E. Leader–follower containment control over directed random graphs. *Automatica* **2016**, *66*, 56–62.
11. Shi, G.; Hong, Y.; Johansson, K.H. Connectivity and Set Tracking of Multi-Agent Systems Guided by Multiple Moving Leaders. *IEEE Trans. Autom. Control* **2012**, *57*, 663–676.
12. Yoshida, K.; Fukushima, H.; Kon, K.; Matsuno, F. Control of a Group of Mobile Robots Based on Formation Abstraction and Decentralized Locational Optimization. *IEEE Trans. Robot.* **2014**, *30*, 550–565.
13. Sabattini, L.; Secchi, C.; Cocetti, M.; Levratti, A.; Fantuzzi, C. Implementation of Coordinated Complex Dynamic Behaviors in Multirobot Systems. *IEEE Trans. Robot.* **2015**, *31*, 1018–1032.
14. Du, S.; Sun, X.; Cao, M.; Wang, W. Pursuing an evader through cooperative relaying in multi-agent surveillance networks. *Automatica* **2017**, *83*, 155–161.
15. Moon, S.; Frew, E.W. Distributed Cooperative Control for Joint Optimization of Sensor Coverage and Target Tracking. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 759–766.
16. Cortés, J.; Martínez, S.; Karatas, T.; Bullo, F. Coverage Control for Mobile Sensing Networks. *IEEE Trans. Robot. Autom.* **2004**, *20*, 243–255.
17. Martínez, S.; Bullo, F.; Cortés, J.; Frazzoli, E. On synchronous robotic networks—Part I: Models, tasks, and complexity. *IEEE Trans. Autom. Control* **2007**, *52*, 2199–2213.
18. Martínez, S.; Bullo, F.; Cortés, J.; Frazzoli, E. On synchronous robotic networks—Part II: Time Complexity of Rendezvous and Deployment Algorithms. *IEEE Trans. Autom. Control* **2007**, *52*, 2214–2226.
19. Nowzari, C.; Cortés, J. Self-Triggered Coordination of Robotic Networks for Optimal Deployment. *Automatica* **2012**, *48*, 1077–1087.
20. Patel, R.; Frasca, P.; Durham, J.W.; Carli, R.; Bullo, F. Dynamic Partitioning and Coverage Control With Asynchronous One-to-Base-Station Communication. *IEEE Trans. Control Netw. Syst.* **2016**, *3*, 24–33.

21. Cortés, J.; Martínez, S.; Bullo, F. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM Control Optim. Calc. Var.* **2005**, *11*, 691–719.
22. Pierson, A.; Figueiredo, L.C.; Pimenta, L.; Schwager, M. Adapting to sensing and actuation variations in multi-robot coverage. *Int. J. Robot. Res.* **2017**, *36*, 337–354.
23. Kantaros, Y.; Zavlanos, M.M. Distributed communication-aware coverage control by mobile sensor networks. *Automatica* **2016**, *63*, 209–220.
24. Li, W.; Liu, Y. Dynamic Coverage Control for Mobile Robot Network with Limited and Nonidentical Sensory Ranges. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 775–780.
25. Lee, S.G.; Diaz-Mercado, Y.; Egerstedt, M. Multirobot Control Using Time-Varying Density Functions. *IEEE Trans. Robot.* **2015**, *31*, 489–493.
26. Gasparri, A.; Sabattini, L.; Ulivi, G. Bounded Control Law for Global Connectivity Maintenance in Cooperative Multirobot Systems. *IEEE Trans. Robot.* **2017**, *33*, 700–717.
27. Boskos, D.; Dimarogonas, D.V. Robustness and Invariance of Connectivity Maintenance Control for Multiagent Systems. *SIAM J. Control Optim.* **2017**, *55*, 1887–1914.
28. Zavlanos, M.M.; Egerstedt, M.B.; Pappas, G.J. Graph-Theoretic Connectivity Control of Mobile Robot Networks. *Proc. IEEE* **2011**, *99*, 1525–1540.
29. Li, X.; Xi, Y. Distributed Cooperative Coverage and Connectivity Maineinance for Mobile Sensing Devices. In Proceedings of the 2009 Joint 48th IEEE Conference on Decision and Control, Shanghai, China, 15–18 December 2009; pp. 7891–7896.
30. Schuresko, M.; Cortes, J. Distributed Tree Rearrangements for Reachability and Robust Connectivity. *SIAM J. Control Optim.* **2012**, *50*, 2588–2620.
31. Aranda, M.; Aragues, R.; López-Nicolás, G.; Sagüés, C. Connectivity-Preserving Formation Stabilization of Unicycles in Local Coordinates Using Minimum Spanning Tree. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 1968–1974.
32. Aragues, R.; Sagüés, C.; Mezouar, Y. Triggered Minimum Spanning Tree for Distributed Coverage with Connectivity Maintenance. In Proceedings of the 2014 European Control Conference (ECC), Strasbourg, France, 24–27 June 2014; pp. 1881–1887.
33. Soleymani, T.; Garone, E.; Dorigo, M. Distributed Constrained Connectivity Control for Proximity Networks based on a Receding Horizon Scheme. In Proceedings of the 2015 American Control Conference (ACC), Chicago, IL, USA, 1–3 July 2015; pp. 1369–1374.
34. Wagenpfeil, J.; Trachte, A.; Hatanaka, T.; Fujita, M.; Sawodny, O. A distributed minimum restrictive connectivity maintenance algorithm. *IFAC Proc. Vol.* **2009**, *42*, 365–370.
35. Robin, C.; Lacroix, S. Multi-robot target detection and tracking: Taxonomy and survey. *Auton. Robots* **2016**, *40*, 792–760.
36. Gallager, R.G.; Humblet, P.A.; Spira, P.M. A Distributed Algorithm for Minimum-Weight Spanning Trees. *ACM Trans. Progr. Lang. Syst.* **1983**, *5*, 66–77.
37. Cao, M.; Hadjicostis, C. *Distributed Algorithms for Voronoi Diagrams and Application in Ad-hoc Networks*; Tech. Rep. UILU-ENG-03-2222, DC-210; UIUC Coordinated Science Laboratory: Champaign, IL, USA, 2003.
38. Ando, H.; Oasa, Y.; Suzuki, I.; Yamashita, M. Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Trans. Robot. Autom.* **1999**, *15*, 818–828.
39. Google Drive Folder. Available online: https://drive.google.com/open?id=1WkSWwcgT32OVk6Z0jX6m9hqubFKV3kmk (accessed on 20 March 2018)
40. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.; Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.* **2014**, *47*, 2280–2292.