Robotica: page 1 of 21. © Cambridge University Press 2016
doi:10.1017/S0263574716000229

A distributed control algorithm for area search by a multi-robot team

Ahmad Baranzadeh* and Andrey V. Savkin

School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, NSW 2052, Australia.
E-mail: a.savkin@unsw.edu.au

(Accepted March 21, 2016)

SUMMARY
In this paper, we present a novel algorithm for exploring an unknown environment using a team of mobile robots. The suggested algorithm is a grid-based search method that utilizes a triangular pattern which covers an area so that exploring the whole area is guaranteed. The proposed algorithm consists of two stages. In the first stage, all the members of the team make a common triangular grid of which they are located on the vertices. In the second stage, they start exploring the area by moving between vertices of the grid. Furthermore, it is assumed that the communication range of the robots is limited, and the algorithm is based on the information of the nearest neighbours of the robots. Moreover, we apply a new mapping method employed by robots during the search operation. A mathematically rigorous proof of convergence with probability 1 of the algorithm is given. Moreover, our algorithm is implemented and simulated using a simulator of the real robots and environment and also tested via experiments with Adept Pioneer 3DX wheeled mobile robots.

KEYWORDS: Mobile robots; Multi-agent networks; Multi-robot search; Distributed control; Search and rescue.

1. Introduction

Using mobile robots has recently become very popular in various tasks. They are employed to achieve from very simple home tasks like vacuum cleaning[1,2] to serious and complicated jobs like finding a dangerous odour source in a warehouse.[3] Search and rescue (SAR),[4,5] patrolling,[6] home and office assistance[7] and fire detection[8] are some important applications of mobile robots. Using only a single robot for such applications has already been studied by many researchers and there is significant progress in this field.

On the other side, using a team consisting of a few robots instead of using only one robot to accomplish these tasks, is a new approach that has attracted many researchers in recent years.[9–13] Compared to single-robot systems, multi-robot systems are more reliable, robust, scalable and flexible. They are more reliable because any failure in one agent would not disable the whole system. Robustness means that using multi-robot systems can reduce errors in odometry, communication, sensing and so on. Scalability represents that with growing amount of work the algorithm applied on multi-robot systems still works, and multi-robot systems are more flexible as they can adopt themselves with any changes in the environment.

Centralized approach and decentralized approach[14,15] are two major approaches to control multi-robot systems. While in the centralized control, the robots are controlled by a central unit, which is easier to apply, decentralized control has important advantages so that it has recently brought researchers to pay more attention to it. In decentralized control, there is no single control centre thus the algorithm is such that the control procedure is distributed among the agents. Thus, one of the main issues in the decentralized control is how the robots share their information among themselves. When robots collaborate to achieve a goal based on an algorithm, they need to share their information.

* Corresponding author. E-mail: a.baranzadeh@unsw.edu.au

The information may be their motion's data like position and velocity, the maps they make during the search procedure or information about the targets they detect. Moreover, in multi-robot systems, a control algorithm must consider some practical limitations such as memory usage,[16] processing speed[17] and communication bandwidth.[18]

Furthermore, mapping and map merging are two other more challenging problems in multi-robot systems. In single-robot systems, a robot makes its map and employs it only by itself. On the other hand in multi-robot systems, each robot makes its map which should be used by the other robots to have a better performance in the algorithm. Therefore, map merging is essential for multi-robot systems; however, it is a challenging issue due to the different coordinates systems of the robots. Researchers proposed several methods for mapping and map merging in multi-robot systems. One solution is based on a global or local positioning system[19] which is not available in many conditions. Another approach assumes that all robots know their positions in a shared map.[8] There are also methods that merge maps of different robots to build a common shared map of the environment.[20,21]

Geometric grid based[22] and topological maps[23] are two main methods for map making of mobile robots. Although the first method is more accurate for the case where the grid size is small enough, it needs considerable memory and takes a remarkable time to process, especially for map merging in multi-robot systems. Thus, in multi-robot systems, topological maps are more efficient and feasible.

Searching an area to find some targets, exploring a whole region and patrolling in a building to detect potential intruders are various related applications of multi-robot systems.[24–26] There are many researches conducted in multi-robot systems with various assumptions about the search area.[27–30] However, few of them propose a comprehensive solution for the shape of the environment and obstacles. For instance, in ref. [27] a structured environment has been assumed as the searching area whereas in ref. [28] an unstructured environment has been considered. In ref. [29], the searching area has been assumed a region without any obstacles while in ref. [30] the operation has been done in an environment with obstacles.

In this paper, in order to address aforementioned issues in multi-robot systems, we propose a new method for distributed control of a multi-robot team whose duty is to search all or a part of an unknown region. The aim of the search can be either finding some targets in the region, patrolling the region frequently or putting some signs on specific points in the region. The suggested algorithm uses a triangular grid pattern, i.e., robots certainly go through the vertices of a triangular grid during the search procedure. To ensure that all the vertices of the region's covering grid are visited by robots, they must have a common triangular grid. In order to achieve that, we use a two-stage algorithm. In the first stage, robots apply an algorithm according to consensus variables to deploy themselves on the vertices of a common triangular grid which covers the region. In the second stage, they begin searching the area by moving between the vertices of the common triangular grid. There are various scenarios can be used in the second stage. In ref. [31], robots use a random-based algorithm to deploy themselves to the one of their six neighbouring vertices. In this paper, we introduce a novel method in that robots move to the nearest vertex which has not already been visited by the other robots.

The proposed algorithm is based in part on ideas from ref. [32] where a distributed random algorithm for self-deployment of a network of mobile sensors has been presented. In ref. [32], authors have proposed a method to deploy a team of mobile sensors on the vertices of a triangular grid that covers entire an area. Indeed, in the proposed search algorithm, we get the benefit of the triangular grid coverage to cover entire an area. It is clear that any triangular grid coverage of a region is a complete blanket coverage of that region. The main advantage of deploying robots in a triangular grid pattern is that it is asymptotically optimal in terms of minimum number of robots required for the complete coverage of an arbitrary bounded area.[33] Therefore, using the vertices of this triangular grid coverage, what is applied in this paper, guarantees complete search of all the region.

One of the advantages of the proposed algorithm is the method by which the robots make and share the maps and use them for exploration. We use a kind of topological map, described in detail in the next sections. Furthermore, we consider a region with an arbitrary shape that contains some obstacles, also we assume the area is unknown to the robots *a priori*. In addition, a mathematically rigorous proof of convergence with probability 1 of the algorithm is given. Moreover, our algorithm is implemented and tested using Mobilesim, a simulator of the real robots and environment. Mobilesim is a powerful simulator which considers robots' real circumstances such as dynamics of motion, encoders, sonar and also borders and obstacles of the environment. We also test our algorithm via

https://doi.org/10.1017/S0263574716000229

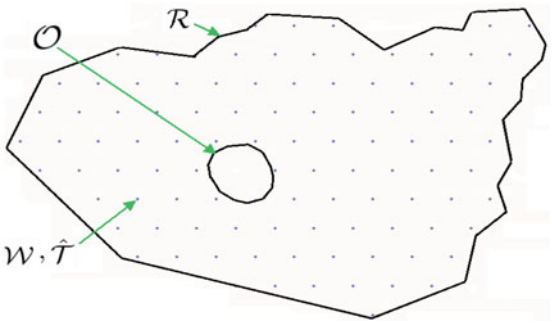


Fig. 1. A triangular grid (dotted area) covers the search area.

experiments by real robots. In fact, we confirm the performance of the proposed algorithm with experiments with Adept Pioneer 3DX wheeled mobile robots in a real world environment.

SAR is another application of the proposed algorithm.[34,35] Clearly, it is significant to save human lives in a disaster such as an earthquake, a plane crash, a missed boat in the ocean and a fire in an industrial warehouse. In many of such cases, SAR operation is usually difficult or even impossible to perform by humans. Therefore, using a team of robots is helpful to perform the operation in order to increase the survival rate and to decrease the risk of the operation. In addition, in a SAR operation, the search area must be completely explored ensuring all possible targets are detected or until all the targets are detected if the number of targets is given. Our suggested algorithm completely covers this goal in the sense that it guarantees exploring all points of the searching area.

The remainder of this paper is organized as follows. The problem statement, some definitions and assumptions are given in Section 2. In Section 3, we describe the first stage of our algorithm which deploys the robots on some vertices of a triangular grid. The second stage of the algorithm is presented in Section 4 that contains the algorithm of searching the region for targets, exploring the whole area or patrolling the area. Simulation results of the proposed algorithm and discussion about them are presented in Section 5. In Section 6, the results of experiments with real robots are given. Finally, a brief conclusion is described in Section 7.

2. Problem Statement

Consider a planar and bounded area $\mathcal{R}$. Also, consider a few number of obstacles $\mathcal{O}_1$,$\mathcal{O}_2$,... $\mathcal{O}_m$ inside the area $\mathcal{R}$ (see Fig. 1). The goal is to search the whole area by a few autonomous mobile robots in order to find some targets. The number of targets may be known or unknown to the robots. We use a distributed algorithm to drive the robots inside the search area as well as avoiding the obstacles and borders. The algorithm is such that the robots follow a pattern to search. The proposed pattern is a triangular grid so that the robots search the area by moving through the vertices of that triangular grid. The grid consists of equilateral triangles with sides r. Figure 2 shows this triangular grid that covers the searching area.

It is assumed that the robots are equipped with sensors to detect the targets and these sensors have a circular sensing area with radius r_s. As shown in Fig. 2, the small circles of radius r_s with centres located on the vertices of the triangular grid are the sensing ranges of the robots. It means that robot i can gather information of or detect a target if it is located in a disk of center $p_i(k)$ with radius r_s defined by $D_{i,r_s}(k) := \{p \in \mathbb{R}^2 : \|p - p_i(k)\| \leq r_s\}$, where k indicates discrete time instances, $k = 0, 1, 2, \ldots$, $p_i(k) \in \mathbb{R}^2$ denotes the Cartesian coordinates of the robot i at time k and $\|\cdot\|$ denotes the Euclidean norm. As demonstrated in Fig. 2, to have an optimal search operation, the common areas of the robots' sensing regions (small circles) must be minimum. To achieve this goal, we assume $r = \sqrt{3}r_s$. However, this is for an ideal and optimal case but in practice where there is sensor noise and/or position uncertainty, the triangles sides r can be chosen a little smaller than $\sqrt{3}r_s$ that reduces the possibility of unexplored areas.

One of the main benefits of multi-robot systems is that robots share their information in order to improve the strength of the search algorithm in terms of time and cost. The information includes the position of robots, maps, explored areas and detected targets. We assume that the robots share

https://doi.org/10.1017/S0263574716000229

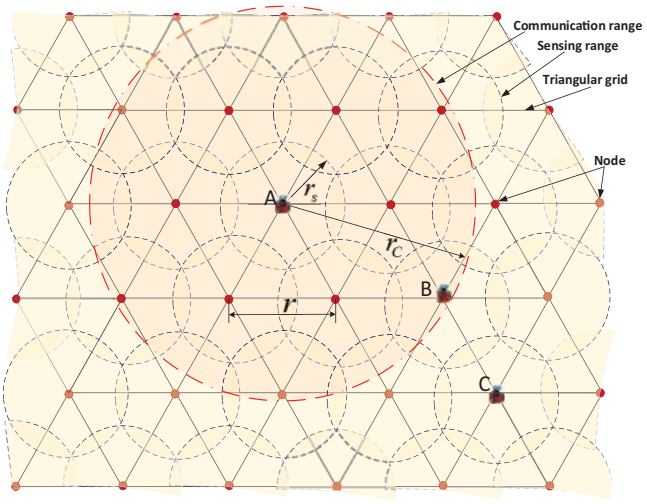


Fig. 2. Robots on vertices of a triangular grid, small circles are robots' sensing ranges and big circle is the communication range of robot A.

their information via a wireless communication so that a robot always sends new information it obtains to the other robots and also listens to receive new information from them. Due to limited communication range of robots, we assume r_c as the communication range that is the same for all the mobile robots. It means that a robot can only receive information from the robots which are located not further than r_c. Therefore, the range of communication for a robot can be defined as the disk of $D_{i,r_c}(k) := \{p \in \mathbb{R}^2 : \|p - p_i(k)\| \leq r_c\}$. In Fig. 2, the big circle of radius r_c is the communication range of robot A, that means robot A can communicate with robot B but not with robot C.

Definition 2.1. Robot j is a neighbour of robot i at time k if it is located on the disk $D_{i,r_c}(k)$. So $\mathcal{N}_i(k) = \{j : p_j \in D_{i,r_c}(k), j \in \{1, 2, \ldots, n\}, j \neq i\}$ is the set of all neighbours of robot i at time k. Also, $|\mathcal{N}_i(k)|$ denotes the number of its neighbours.

It is essential for robots in any search operation to make the map of the searching area when it has not already been available to the robots. That helps the robots to keep the information of the search operation in their memory for future use or to send it to a centre or to the other robots of the search team. The problem of mapping is a challenging problem especially when there is a team of robots instead of only one robot. The main problem is merging the maps which are made by different robots. It is much easier for robots to make maps and merge them in the case that there is a central control station or when robots can use a central positioning system like GPS. On the other hand, without a central positioning system, robots have to make their map themselves in their own coordinate systems. Furthermore, the information of maps needs lots of memory also processing of maps for merging is a time-consuming task. In such cases, i.e., distributed systems, it is useful to employ topological maps instead of grid geometry maps that need a considerable amount of memory.

In this paper, we take advantage of using a kind of topological map that robots make and share among themselves. It is also assumed that the searching area is unknown to all robots already. Therefore, robots have to make the map of the area during the search operation.

To define the problem, we utilize definitions of ref [36]. . . Also, to clarify some terms in the rest of the paper and state our theoretical results, we introduce a number of assumptions and definitions.

Assumption 2.1. The area $\mathcal{R}$ is bounded and connected, also the obstacles $\mathcal{O}_n$ are non-overlapping, closed, bounded, and linearly connected sets for any $n \geq 0$.

Definition 2.2. Let $\mathcal{O} := \cup\mathcal{O}_n$ for all $n \geq 0$, then introduce $\mathcal{W} := \{ p \in \mathcal{R}: p \notin \mathcal{O}\}$.

As mentioned before, the robots can detect borders of the area $\mathcal{R}$ and all the obstacles therein. Accordingly, $\mathcal{W}$ is actually those points in the area $\mathcal{R}$ that robots can go there during the search operation. Also, our search procedure is so that the robots go through vertices of a triangular grid that covers this area and cuts the plane into equilateral triangles. A triangular grid can be determined by

its vertices; therefore, when we say a triangular grid, it means the set of all vertices of it. It is obvious that there exist an infinite number of triangular grids.

Definition 2.3. Consider $\mathcal{T}$, one of the all possible triangle grids that covers the area $\mathcal{R}$. The $\hat{\mathcal{T}} := \mathcal{T} \cap \mathcal{W}$ is called a triangular grid set in $\mathcal{W}$ (see Fig. 1).

Simply, $\hat{\mathcal{T}}$ is the set of all vertices of the covering triangular grid which robots must visit during the search operation. Since we use topological maps, $\hat{\mathcal{T}}$ is actually the map of $\mathcal{W}$.

Definition 2.4. A set consisting of all the vertices of a triangular grid in an area is called a map of that area.

In fact, what the robots actually save in their memory as maps, are the coordinates of the vertices. A robot can also put some tags on these vertices to assign some attributes like visited, unvisited, occupied, etc. The robots share their maps with their neighbours too.

The relationships between robots can be defined by an undirected graph $G(k)$. We assume that any robot of the multi-robot team is a node of the graph $G(k)$ at time k, i.e., i in $V_G = \{1, 2, \ldots, n\}$, the node set of $G(k)$, is related to robot i. In addition, robot i is a neighbour of robot j at time k if and only if there is an edge between the nodes i and j of graph $G(k)$ where $i \neq j$. Therefore, the problem of communication among the team of robots equals the problem of the connectivity of the related graph. It is obvious that it does not need for robot i to be the neighbour of robot j to get the information from it. The information can be transferred through the other robots which connect these robots in the related graph. Figure 2 shows this condition in which robot A cannot directly communicate with robot C but can do it through robot B. To guarantee the connectivity of the graph, we accept the following assumption.[37]

Assumption 2.2. There exists an infinite sequence of contiguous, non-empty, bounded, time-intervals $[k_j, k_{j+1})$, $j = 0, 1, 2, \ldots$, starting at $k_0 = 0$, such that across each $[k_j, k_{j+1})$, the union of the collection $\{G(k) : k \in [k_j, k_{j+1})\}$ is a connected graph.

Since the main goal of the proposed algorithm is to search an area in such a way that robots certainly go through the vertices of a triangular grid, first, robots have to be located on some vertices of that grid. Thus, we divide our algorithm into two stages. In the first stage, robots make a common triangular grid in order to be finally located on some vertices of it. To achieve this goal, we apply consensus variables method which is a known distributed method for multi-robot systems.[38] In the second stage, robots start to search the area based on moving among the vertices of the created grid which is common among all members of the team. The next two sections explain two stages of the suggested algorithm in detail.

3. Distributed Search Algorithm—First Stage

In the first stage of the suggested algorithm, using consensus variables, robots make a triangular grid that is common among all the members of the team. In the beginning, robots are located anywhere in the area $\mathcal{W}$. Each robot assigns its position and heading angle respect to its own coordinate system. The centre of a robot at the starting point is assumed to be the origin of its coordinate system and its heading vector as the x-axis. To define a unique triangular grid with equilateral triangles in a plane, we only need a point and an angle. Thus, the point q which is any vertex of the grid together with the angle θ that is the angle of the grid, uniquely defines the triangular grid $\hat{\mathcal{T}}[q,\ \theta]$ that covers the area $\mathcal{W}$ (see Fig. 1). We consider that at the beginning, the triangular grid a robot makes for itself at each vertex, is based on the position and heading of the robot as q and θ, respectively. Accordingly, each robot has its own grid that is different from the other robots' grids. To combine these different grids to a unique grid which will be common among all robots, we apply the consensus variables approach.

We assume that at any time k the robot i has two consensus variables, $q_i(k)$ and $\theta_i(k)$, on which it builds its triangular grid $\hat{\mathcal{T}}[q_i(k),\ \theta_i(k)]$. At first, these consensus variables are not the same for different robots, so their triangular grids are not the same as well. Using the proposed algorithm will bring the consensus variables $q_i(k)$ and $\theta_i(k)$ from different values of $q_i(0)$ and $\theta_i(0)$ to the same values of q_0 and θ_0 for all the robots. That is, a common triangular grid for all the members of the team is built based on q_0 and θ_0 which are the same for all.

https://doi.org/10.1017/S0263574716000229

Assumption 3.1. The initial values of the consensus variables θ_i satisfy $\theta_i(0) \in [0,\ \pi)$ for all $i = 1, 2, \ldots,\ n$.

Definition 3.1. Consider p be a point on the plane, also q and θ are a vertex and an angle that build the triangular grid $\hat{\mathcal{T}}[q,\ \theta]$. Then, $C[q, \theta](p)$ will be the closest vertex of $\hat{\mathcal{T}}[q,\ \theta]$ to p. If there is more than one vertex, any of them can be chosen.

Now, we propose the following rules for the first stage of our algorithm:

$$\theta_i(k+1) = \frac{\theta_i(k) + \Sigma_{j\in\mathcal{N}_i(k)}\theta_j(k)}{1 + |\mathcal{N}_i(k)|};$$

$$q_i(k+1) = \frac{q_i(k) + \Sigma_{j\in\mathcal{N}_i(k)}q_j(k)}{1 + |\mathcal{N}_i(k)|} \tag{1}$$

$$p_i(k+1) = C[q_i(k),\ \theta_i(k)](p_i(k)). \tag{2}$$

The rule (1) causes that robots achieve consensus on heading using θ_i and achieve consensus on phase shift using q_i. Based on q and θ calculated by each robot using rule (1), a robot makes a triangular grid for itself, which is used in rule (2). Figure 3 illustrates the rule (1). Suppose that there are four robots in the environment. At time k, robots R2 and R3 are located in the communication range of robot R1 but robot R4 is out of the range. Therefore, only robots R2 and R3 are the neighbours of robot R1. As a result, robot R1 updates variable θ_1 using the information from R2 and R3, i.e., $\theta_1(k+1)$ will be the average of θ_1, θ_2 and θ_3 at time k (see Fig. 3(a)). In a similar way, the variable q is updated as shown in Fig. 3(b).

Rule (2) means that whenever a robot makes its triangular grid, it will move to the nearest vertex on it. Figure 4 demonstrates this stage in which robot i located at $p_i(k)$ at time k, makes the triangular grid $\hat{\mathcal{T}}[q,\ \theta]$ using $\theta_i(k)$ and $q_i(k)$. Then, it moves to the nearest vertex of the grid, which will be the position of robot i at time $k+1$, i.e., $p_i(k+1)$. Since rule (1) brings the same q and θ for all the robots, they eventually will build a common triangular grid, and they all will be located on its vertices.

Remark 3.1. The robots initially do not have a common coordinate system otherwise the consensus building problem would be trivial. Therefore, each robot has consensus variables $\theta_i(k)$ and $q_i(k)$ in its own coordinate system. However, each robot knows the bearing and the distance to each of its neighbouring robots. Using this information, at any time instance k, the robot i sends to a neighbouring robot j the consensus variables $\theta_i(k)$ and $q_i(k)$ re-calculated in the coordinate system with the line (p_i, p_j) as the x-axis, p_j as the origin and the angle $\theta_i(k)$ is measured from this axis in counter-clockwise direction. Using this information, each robot can re-calculate the sums (1) at each time step in its own coordinate system.[36]

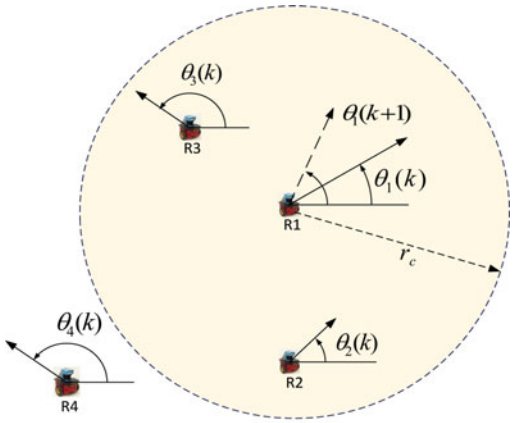
Theorem 3.1. Suppose that Assumptions 2.1, 2.2 and 3.1 hold and the mobile robots move according to the distributed control law (1), (2). Then there exists a triangular grid set $\hat{\mathcal{T}}$ such that for any $i = 1, 2,\ \ldots,\ n$, there exists a $\tau \in \hat{\mathcal{T}}$ such that $\lim_{k\to\infty} p_j(k) = \tau$.

Proof. Assumption 2.1 and the update law (1) guarantee that there exist a θ_0 and q_0 such that

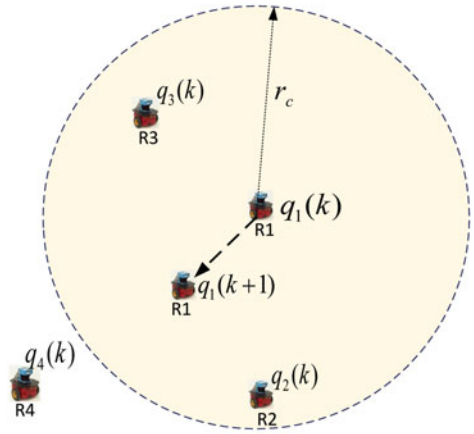
$$\theta_j(k) \to \theta_0,\ q_j(k) \to q_0\ \forall i = 1, 2,\ \ldots,\ n \tag{3}$$

(see[37]). Furthermore, the update law (2) guarantees that $p_j(k+1) \in \hat{\mathcal{T}}[q_j(k),\ \theta_j(k)]$. Therefore, this and (3) guarantee that $\lim_{k\to\infty} p_j(k) = \tau$ where $\tau \in \hat{\mathcal{T}}[q_0,\ \theta_0]$. This completes the proof of Theorem 3.1. □

https://doi.org/10.1017/S0263574716000229



(a)



(b)

Fig. 3. Updating consensus variables θ and q using the information from the neighbours.

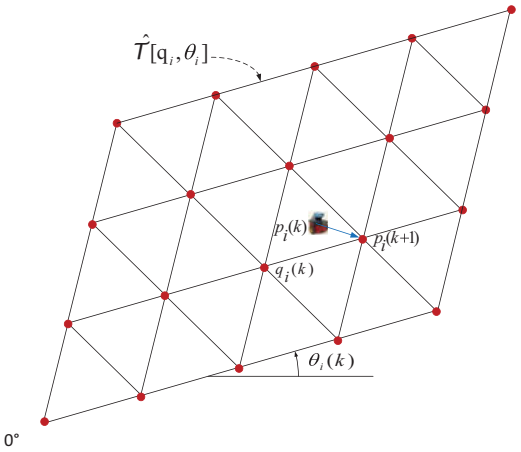


Fig. 4. A robot moves to the nearest vertex of its triangular grid.

4. Distributed Search Algorithm—Second Stage

As described in the previous section, the first stage of the suggested algorithm locates all the robots on the vertices of the common triangular grid $\hat{\mathcal{T}}$. The next step will be the search of the area $\mathcal{W}$ based on moving the robots among the vertices of the map of the area. In this regard, there are a few scenarios that can be considered to search the area. The first scenario is exploring the whole area which can be applicable when the robots are searching for an uncertain number of targets. Therefore, to detect all possible targets, the team of robots have to search the whole area. Patrolling of the area is the other application for this scenario where the robots should move continuously to detect the possible intruders to the area. In the case of given number of targets which is our second scenario, the search operation should be stopped, without searching the whole area, when all the targets are detected.

4.1. Searching the whole area

To make sure that the whole area is explored by the team of robots, each vertex in the triangular covering grid set of the area $\mathcal{W}$ must be visited at least one time by a member of the team. Consider $\hat{\mathcal{T}}$ is a triangular covering grid of $\mathcal{W}$, and also each vertex of $\hat{\mathcal{T}}$ has been visited at least one time by a robot of the team. This guarantees that the area $\mathcal{W}$ has been completely explored by the multi-robot team. Since the robots do not have any map at the beginning, they have to do map making during the search operation so that their maps will gradually be completed.

Definition 4.1. Let $\hat{\mathcal{T}}_i(k)$ be as the set of all vertices of $\hat{\mathcal{T}}$ have been detected by robot i at time k, then $\hat{\mathcal{T}}(k) = \bigcup \hat{\mathcal{T}}_i(k)$ will be the map of the area $\mathcal{W}$ detected by robots until time k.

Note that a detected vertex is different from an explored vertex. These terms are defined in detail in the following definitions.

Definition 4.2. A detected vertex means that vertex is detected by a robot using map making, and it is in the map of that robot though it might be visited or not by the robots.

Definition 4.3. An explored vertex is a vertex that is visited by at least one member of the team.

Definition 4.4. The map of robot i at time k, $\mathcal{M}_i(k)$, is the set of those vertices in $\hat{\mathcal{T}}$ detected by the robot i itself or received from other robots by which they are detected until time k.

Definition 4.5. Suppose a Boolean variable $V_\tau(k)$ which defines the state of vertex $\tau \in \hat{\mathcal{T}}(k)$ at time k. $V_\tau(k) = 1$ if the vertex τ has already been visited by at least one of the robots, otherwise $V_\tau(k) = 0$.

Assumption 4.1. The triangular grid set $\hat{\mathcal{T}}_i(k)$; $k = 0, 1, \ldots$ is connected if $\tau \in \hat{\mathcal{T}}_i(k)$, then, at least, one of the six nearest neighbours of τ also belongs to $\hat{\mathcal{T}}(k)$.

Consider at time k robot i is located at point $p_i(k)$ and it wants to go to the next vertex, the following rule is proposed as the second stage of our algorithm:

$$p_i(k+1) = \begin{cases} \hat{\mathfrak{m}}_i(k) & \text{if } |\hat{\mathcal{M}}_i(k)| \neq 0 \\ p_i(k) & \text{if } |\hat{\mathcal{M}}_i(k)| = 0 \end{cases} \tag{4}$$

where $\hat{\mathcal{M}}_i(k) = \{\mathfrak{m} \in \mathcal{M}_i(\mathfrak{k}); \mathfrak{V}_\mathfrak{m}(\mathfrak{k}) = \mathfrak{o}\}$ is the set of all elements of $\mathcal{M}_i(k)$ have not been visited before, $|\hat{\mathcal{M}}_i(k)|$ denotes the number of elements in $\hat{\mathcal{M}}_i(k)$ and $\hat{\mathfrak{m}}_i(k)$ is the vertex in $\hat{\mathcal{M}}_i(k)$ nearest to robot i at time k.

Applying the rule (4) ensures that the area $\hat{\mathcal{T}}$ has been completely explored and every vertex of it is visited at least one time by a robot of the team.

Theorem 4.1. Suppose that all assumptions hold and the mobile robots move according to the distributed control law (4). Then for any number of robots, with probability 1 there exists a time $k_0 \geq 0$ such that $V_\tau(k_0) = 1; \quad \forall \tau \in \hat{\mathcal{T}}$.

Proof. The algorithm 4 defines an absorbing Markov chain which contains many transient states and some absorbing states that are impossible to leave. Transient states are all the vertices of the triangular grid $\hat{\mathcal{T}}$ which are occupied by the robots during the search procedure. On the other hand, absorbing states are the vertices where the robots stop. Using the algorithm, a robot goes to the

https://doi.org/10.1017/S0263574716000229

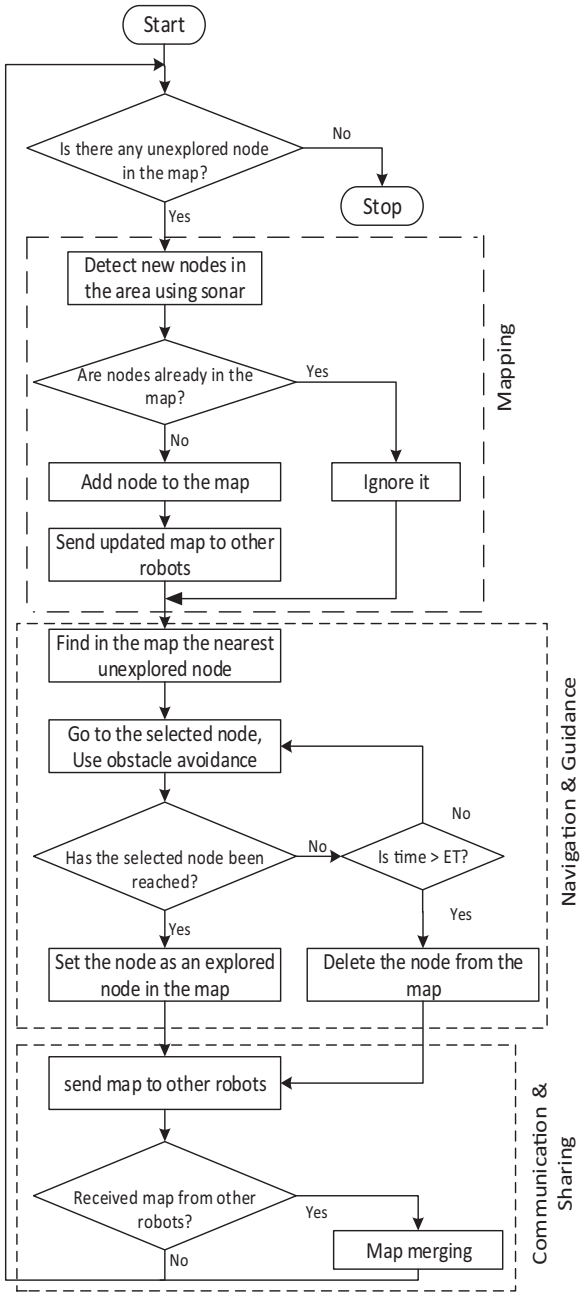


Fig. 5. The flowchart of the suggested algorithm, searching the whole area.

vertices where may have not been visited previously. Therefore, the number of transient states will eventually decrease. This continues until the number of robots is equal to the number of unvisited vertices which will be absorbing states. It is also evident that these absorbing states can be reached from any initial states, with a non-zero probability. This implies that with probability 1, one of the absorbing states will be reached. This completes the proof of Theorem 4.1. □

In Fig. 5, a flowchart of the second stage of the algorithm is presented that shows how our decision-making approach is implemented. At the first step, robots start making their maps using their sonar. Each robot, based on the vertex on which it is located, assumes some probable neighbouring vertices on the common triangular grid. The number of these probable neighbouring vertices and their distance to the robot depend on the robot's sonar range. Then the robot uses the sonar to detect its surrounding environment including borders and obstacles. If any of those probable neighbouring vertices is

located outside the borders or blocked by an obstacle, it will be ignored. The rest of those probable neighbouring vertices will be added to the map of the robot. This step is repeated every time that the robot occupies a vertex. In order to avoid sticking in borders or obstacles, we consider a margin near the borders and obstacles that depends on the size of the robot. If a vertex on the map is closer to the borders or obstacles less than the margin, it will be eliminated from the map.

Whenever a robot makes any changes to its map, it sends the new map to the other robots using transmitting packets. On the other side, whenever a robot receives a packet of data, it extracts the new vertices from the received map and adds them to its map. Since the communication range of the robots is limited, if two robots are far from each other, they cannot directly communicate but can do it via other robots. In other words, since there is a connected network of robots, each robot has the role of a hub in the network in order to share the maps among the robots. Therefore, all the robots that are connected and make a network have a common map. In the second phase of the algorithm when the robots have a common triangular grid map, a robot can go far from the other robots and be disconnected from the team for a while. In this case, sharing maps between the disconnected robot and the others is paused until the robot returns back to the communication range of the team again.

In the next step, each robot chooses the nearest unexplored vertex in the map and goes there. Whenever a robot visits a vertex, it marks that as an explored vertex in its map. Therefore, all the robots know which vertices have not been explored yet. To find the nearest vertex that has not been explored yet, each robot searches the map stored in its memory. This vertex has probably been detected by the robot itself or added to robot's map via sharing map among robots. After finding the nearest unexplored vertex, the robot moves to reach there. If the robot reaches the target vertex, it marks that vertex as an explored vertex in its map and sends it to the other neighbouring robots as well. However, because of some practical issues, maybe it is impossible to reach the target vertex at a limited time or even maybe the target vertex is fake that wrongly created during mapping. To avoid such problems, we include a factor of time. Since the robot knows its location and the location of the target, it can estimate the time needed to achieve the goal based on the distance to the target and velocity of the robot. Here, we consider the parameter ET as the expected time to reach the target, that is a factor of the estimated time. This coefficient, that has a value greater than one, actually reflects the effects of a non-straight route because of the shape of the searching area and also existing obstacles or other robots on the robot's path. If the travelling time was more than ET, the robot would ignore that target vertex, delete it from its map and send the updated map to the other robots.

Since the area has borders and obstacles, the robots should avoid them while they are searching for the targets. That is why we have to use an obstacle avoidance in our algorithm. Moreover, because of practical problems like sonar and encoders accuracy and slipping of the robots, there might be a difference between the actual position of a robot and the coordinates of the vertices stored in its map. Therefore, if a robot is closer to a target vertex than a specified distance, we consider the goal has been achieved.

4.2. Searching for targets

When the robots are looking for the targets in the area, they should continue the searching operation till all the targets are detected. If the number of the targets has not been specified before, they have to search the entire area like what described in Section A. When robots know the number of the targets, they do not need to explore the entire area but until all the targets are detected. Suppose $\mathfrak{T} = \{\mathfrak{T}_1, \mathfrak{T}_2, \ldots, \mathfrak{T}_{n_t}\}$ be the set of n_t static targets should be detected by the robots. As it was stated before, we assume the robots equipped with sensors by which the targets can be detected whenever they are close enough to the robots. The distance by which the robots should be close to the targets to be able to detect them is the sensing range of the robots (r_s).

Definition 4.6. Suppose a Boolean variable $V_{\mathfrak{T}_j}(k)$ which defines the state of target $\mathfrak{T}_j$ at time k. $V_{\mathfrak{T}_j}(k) = 1$ if the target $\mathfrak{T}_j$ has already been detected by at least one of the robots, otherwise $\mathfrak{T}_j(k) = 0$.

Then we modify the rule (4) as the following rule to ensure that the search operation stops after finding all the targets.

$$p_i(k+1) = \begin{cases} \hat{\mathfrak{m}}_i(k) & \text{if } \exists \mathfrak{T}_j \in \mathfrak{T}; \mathfrak{V}_{\mathfrak{T}_j}(\mathfrak{k}) = \mathfrak{o} \\ p_i(k) & \text{if } \forall \mathfrak{T}_j \in \mathfrak{T}; \mathfrak{V}_{\mathfrak{T}_j}(\mathfrak{k}) = \mathfrak{1}. \end{cases} \tag{5}$$

https://doi.org/10.1017/S0263574716000229

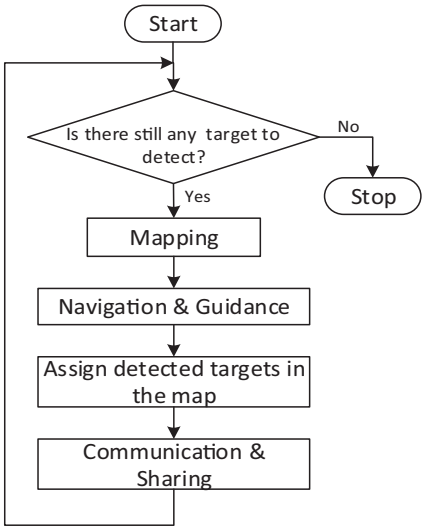


Fig. 6. The flowchart of the suggested algorithm, searching for targets.

Theorem 4.2. Suppose that all assumptions hold, and the mobile robots move according to distributed control law (5). Then for any number of robots and any number of targets, with probability 1 there exists a time $k_0 \geq 0$ such that $\forall j$; $V_{\mathfrak{T}_j}(k_0) = 1$.

Proof. Proof is similar to the proof of Theorem 4.1. □

The flowchart in Fig. 5 can also be used to describe this operation. Figure 6 depicts the procedure we use to implement this algorithm. Most procedures are the same as the previous one; therefore, we ignore their description. We only need to change the condition that stops the operation in the algorithm. It means that the operation should be stopped when all the targets have been detected. Also, the robots send the information about the detected targets to the other members of the team.

4.3. Patrolling

The above algorithms are appropriate for the cases where the robots should break the search operation, for instance, when the aim is finding some pre-determined objects or labelling some vertices of a grid. In cases where a permanent search is needed, for example in applications such as continuously patrolling or surveying a region, the algorithm should be modified such that the search procedure maintains. This can be done by periodically changing the states of the vertices in the maps of the robots; hence, the robots consider those vertices as the targets again. To do that, the state of the previously visited vertices should be changed to unvisited in the map of the robots. Based on some conditions such as the size of the area and the number of the robots, the period of changing the state of vertices can be chosen.

4.4. Robots' motion

To have a more real estimation of time that the robots spend to search an area to detect the targets, we apply motion dynamics in our simulations. Also, since the environment is unknown to the team, it is essential to use an obstacle avoidance in the low-level control of the robots. As a result, we use a method of reactive potential field control in order to avoid obstacles.[39]

5. Simulation Results

To verify the suggested algorithm, computer simulations are employed. The region $\mathcal{W}$ is considered to be searched by a few robots (see Fig. 1). We suppose a multi-robot team with three robots which are randomly located in the region $\mathcal{W}$ with random initial values of angles. The goal is to search the whole area by the robots using proposed grid-based algorithm.

To simulate the algorithm, MobileSim, a simulator of mobile robots developed by the Adept MobileRobots, is used. We also use Visual C++ for programming and ARIA, a C++ library that

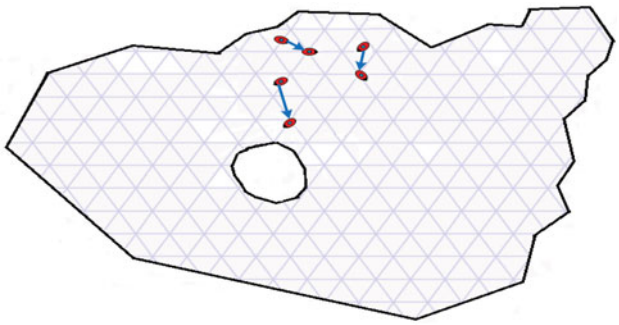


Fig. 7. Applying the first stage of the algorithm; arrows show the initial and final position of each robot.

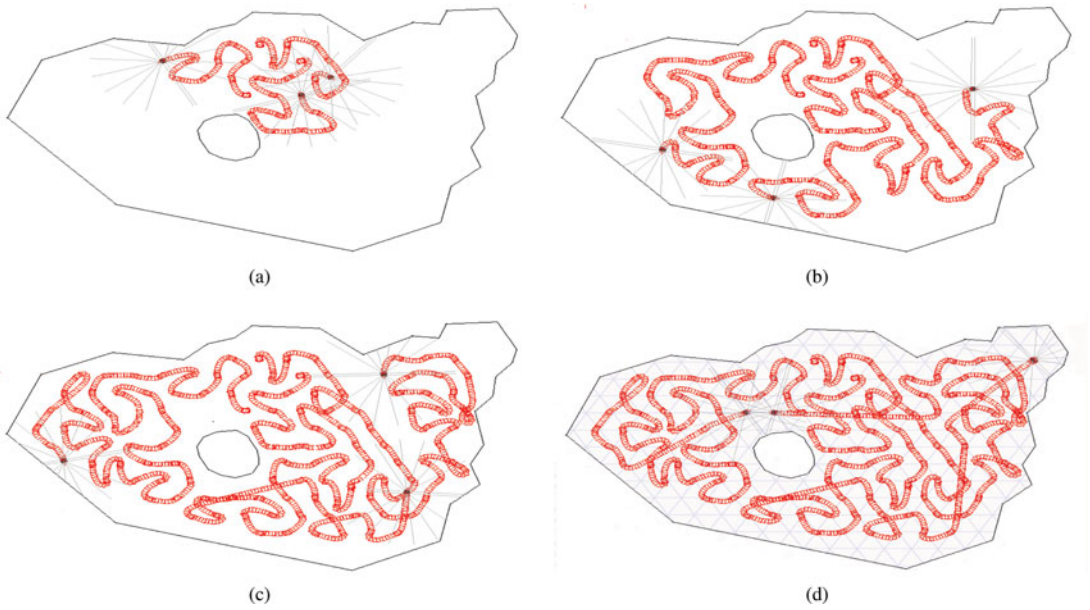


Fig. 8. Robots' trajectories after applying the second stage of the algorithm; (a) after 3 min 21 s, (b) after 5 min 45 s, (c) after 7 min 26 s, (d) after 8 min 23 s.

provides an interface and framework for controlling the robots. In addition, Pioneer 3DX is selected as the type of the robots. Since this simulator simulates the real robots along with all conditions of a real world, the results of the simulations would be obtained in the real world experiments with the real robots indeed. Furthermore, to prevent collisions between robots and to avoid the obstacles and borders, an obstacle avoidance algorithm is applied using functions provided in ARIA library. Moreover, to avoid hitting and sticking to the borders, we assume a margin near the borders such that the robots do not pass it.

5.1. Searching the whole area

As mentioned before, a two-stage algorithm is used to achieve the goal. First, the algorithm (1), (2) is applied which uses consensus variables in order to drive the robots to the vertices of a common triangular grid. Figure 7 shows the initial positions of the robots at $k = 0$ and also their positions after applying this stage of the algorithm. As depicted in Fig. 7, in this case, the robots are located at desired place on the vertices of a common triangular grid at time 1 min 47 s.

The second stage of the algorithm, i.e., the algorithm (4), is applied whenever the first stage is completed. Figure 8 demonstrates the result of applying algorithm (4) on the robots. As seen in Fig. 8, the robots go through the vertices of the common triangular grid based on the proposed algorithm until the whole area is explored by the robots. Figures 8(a), (b)and (c) display the trajectories of the robots at times 3 min 21 s, 5 min 45 s and 7 min 26 s after applying algorithm (4), respectively. Figure 8(d) shows trajectories of the robots at time 8 min 23 s when the search has been completed.

Table I. Time of search duration.

No of robots	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Min	16.98	9.55	7.03	5.68	5.09	4.23	3.72	3.66	3.44	3.38	3.18	3.12	2.98	3.01	2.88
Max	23.02	13.61	9.98	7.90	6.98	5.82	5.05	4.96	4.77	4.65	4.38	4.26	4.12	4.17	3.76
Average	20.51	11.62	8.56	6.73	5.97	5.00	4.42	4.37	4.09	4.00	3.83	3.71	3.56	3.57	3.31
STD	1.77	1.20	0.98	0.54	0.62	0.52	0.37	0.40	0.42	0.36	0.35	0.33	0.34	0.41	0.25

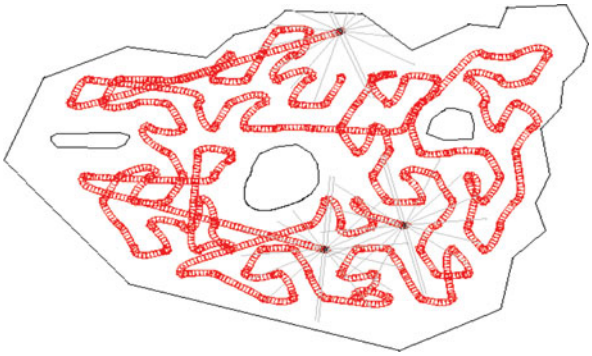


Fig. 9. Searching an area with three obstacles.

It is obvious that the area $\mathcal{W}$ has been completely searched by the robots such that each vertex of the covering triangular grid has been occupied at least one time by the robots. In this case study, we assume that the sides of the equilateral triangles are 2 meter (sensing range of the robots is $\frac{2}{\sqrt{3}}$ m) and the communication range between robots is 10 meter. The area of the region $\mathcal{W}$ is about 528 m^2.

Figure 9 shows another simulation result but for an area with three obstacles with different shapes. It shows that the algorithm indeed works with any number of obstacles.

Although any number of robots can be used for the search operation, it is obvious that more robots complete the operation in a shorter time. However, more number of robots certainly increases the cost of the operation. The question is, how many robots should be used in order to optimize both the time and cost. It seems that may be somehow dependent on the shape of the region and the obstacles and also the sides of the triangles. In order to have a better view of a relation between the number of robots and the search duration, 20 simulations with different number of robots have been done. We consider teams consisting of 1 to 15 robots. Then minimum, maximum, average and standard deviation are calculated for the search duration of each team. Table I displays the results of these simulations that are also depicted in Fig. 10.

As depicted in this figure, the search time decreases by increasing the number of robots. It is noticeable that after increasing the number of robots to a specific number, the search duration almost remains constant. Indeed, increasing the number of robots increases the probable collision between robots during the operation. Consequently, robots have to turn each other to avoid the collision, and that is the main reason which increases the search time. Therefore, increasing the number of robots more than a specific value (six robots in this case) will be ineffectual in term of time and should be avoided to save cost. As obvious from Fig. 10, increasing the number of robots more than six, improves the searching time less than one minute.

5.2. Searching for targets

To demonstrate that how the algorithm works in the case of the search for the certain number of targets, we consider an area the same as in section A with three targets therein. The targets, shown by green disks in Fig. 11, are located in different parts of the area. We evaluate the presented algorithm to figure out how they can find the targets using multi-robot teams with different number of robots. As depicted in Fig. 11, the results of simulation by the teams consisting of one to six robots have been presented. A target is assumed detected whenever it lies in the sensing range of a robot of the team. If there is just one robot in the team, all the targets should be detected by that robot; thus, it will take longer time in compare with the cases that there are more robots in the team.

Table II. Time of detecting targets.

No of robots	1	2	3	4	5	6	7	8	9	10
Min	7.82	3.95	2.35	2.05	1.73	1.54	1.45	1.38	1.32	1.27
Max	15.49	8.02	5.05	4.21	3.63	3.42	2.88	2.56	2.19	2.01
Average	11.95	5.99	3.61	3.02	2.58	2.34	2.07	1.80	1.70	1.60
STD	2.03	1.37	0.79	0.71	0.69	0.61	0.47	0.36	0.30	0.24

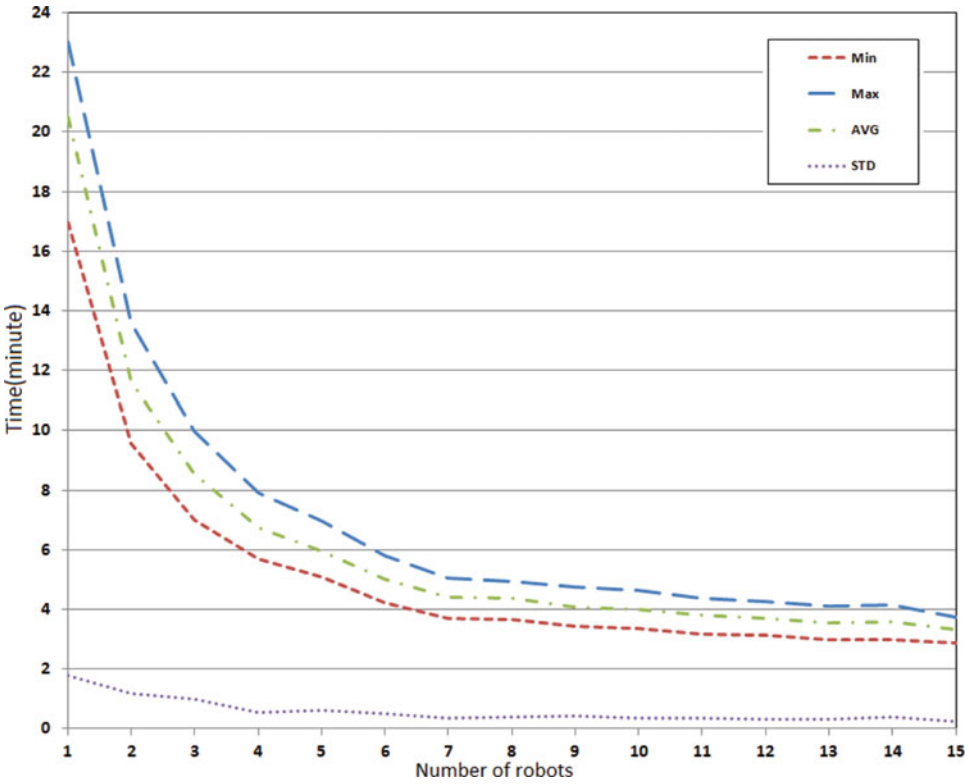


Fig. 10. Duration of search vs. number of robots.

As shown in Fig. 11(a), a robot searches for three targets using the presented search algorithm and they all have been detected after 12 min 49 s. Figure 11(b) shows the same case with two robots in the team so that a robot has detected one target and the other one has detected two other targets in 4 min 3 s. It should be mentioned that in Fig. 11 only the path of the robots after making consensus has been displayed. In Fig. 11(c), three robots search for three targets and each robot finds one of them in 3 min 32 s. As shown in the figure, when a robot detects a target, it continues the search operation until all the targets are detected by the team. Figure 11(d)–(f) show the search operation using 4–6 robots, respectively.

What is very noticeable in this case is that it is expected decreasing the time of detecting targets by increasing the number of robots while it has not occurred in some instances. For example, when the number of robots has been increased from three to four, the time of detecting the targets has been increased from 3 min 32 s to 3 min 50 s. To explain why this happens, we should consider the fact that the time needed to detect the targets depends on many parameters not only on the number of robots. The shape of the area and obstacles therein, the initial position of the robots in the area and also the relative distance of the robots and targets are significant parameters that affect the time of the search. The other serious parameter must be considered is the nature of the search algorithm that is semi-random. Indeed, a robot always chooses the nearest unexplored vertex as its destination vertex,

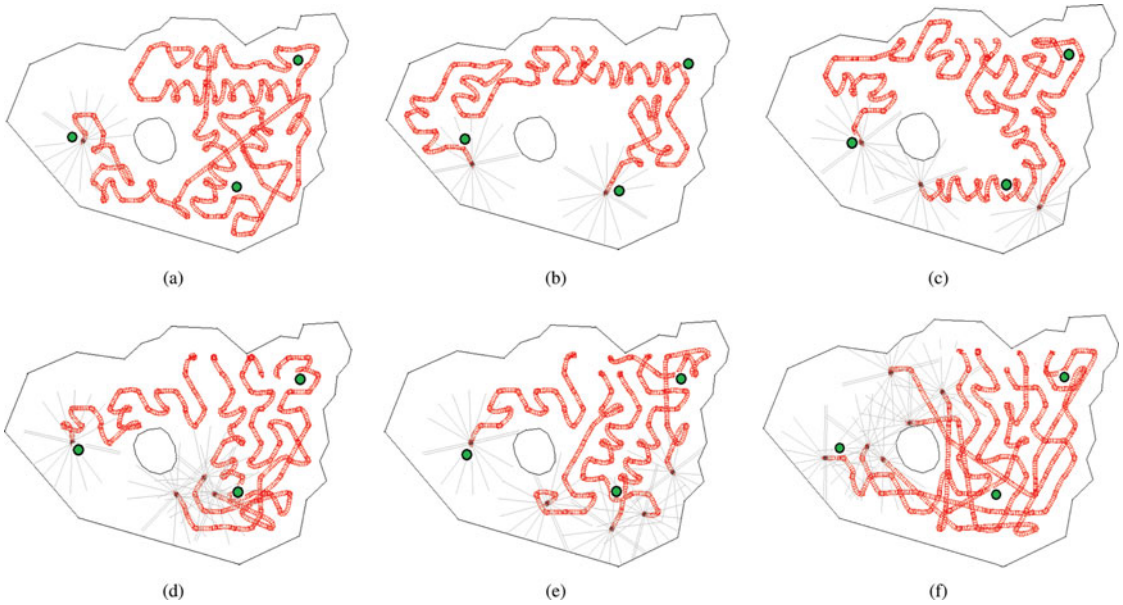


Fig. 11. Robots' trajectories in the case of searching for three targets. (a) one robot detects three targets in 12 min 49 s, (b) two robots detect the targets in 4 min 3 s, (c) three robots detect targets in 3 min 32 s, (d) four robots detect the targets in 3 min 50 s, (e) five robots detect the targets in 1 min 51 s and (f) six robots detect the targets in 2 min 15 s.

but sometimes there are a few vertices which can be chosen as the nearest, and the robot randomly selects one of them. That is we might have different paths for the same cases.

To discover more about how the number of robots effects on the search duration, we should do more simulations for each case. For example, Fig. 13 shows the paths of the robots in the case that two robots are looking for three targets similar to the previous instance. It shows that we have different paths thus different search duration. While in the first simulation the targets are detected in 4 min 3 s, in the last one it is occurred in 7 min 41 s which is much more than the first one. Figure 14 shows the results of three different simulations using five robots. As depicted in that figure, the period of search for these simulations are 1 min 51 s, 1 min 58 s and 2 min 33 s. Comparing to the case with two robots, it is obvious that the differences between periods of search in this case are less than the case of the team with two robots. That is an expected result, because more robots mean more coverage of the searching area hence the chance of detecting targets increases.

In order to have a better view of a relation between the number of robots and the search duration, 20 simulations with a different number of robots have been done. We consider teams consisting of 1 to 10 robots. Then minimum, maximum and the average of search duration for simulations with each team as well as their standard deviations are calculated. Table II displays the results of these simulations that are also depicted in Fig. 12. The results show that although the average of search duration to detect the targets decreases by increasing the number of robots, the standard deviation of the team with less number of robots is significantly high. Simply, the number of robots should be proportional to the searching area to have an adequate chance to detect the targets in an acceptable time.

5.3. Comparison to other methods

To evaluate the proposed triangular grid algorithm against other algorithms, we compare it with two algorithms given in ref. [40], Levy random-walk with potential field algorithm and fixed-length random algorithm. Table III gives the comparison between the proposed algorithm and those in ref. [40] for the searching time. In ref. [40], a 400 m^2 regular square area is considered as the searching area while the area in this paper has an irregular shape with an area of about 528 m^2. Also, the speed of robots in ref. [40] is considered 60 cm/s but we use 40 cm/s as the maximum speed for the robots. As Table III shows, despite the larger area and slower robots, the proposed algorithm is much faster than the algorithms in ref. [40]. For instance, in the case of a team with five robots, the searching time

Table III. Comparison with other methods (Searching time, seconds).

No of robots	Fixed-length random	Levy random-walk	Triangular grid
1	15,200	6,880	1,230
5	5,250	1,170	358
10	2,325	880	240

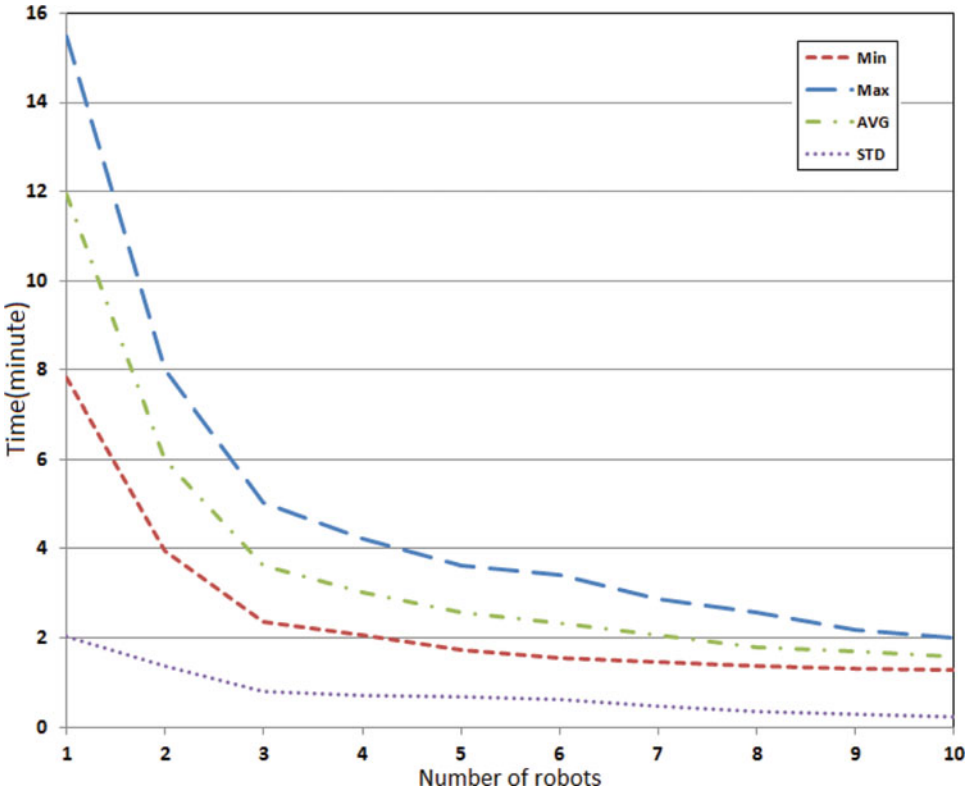


Fig. 12. Time of target detection vs. number of robots.

for the algorithms in ref. [40] are 5250 and 1170 s while for the proposed algorithm, the average is 240 s. Notice that the searching times given in Table III are approximated values extracted from the graph in ref. [40] and also average values from Table I.

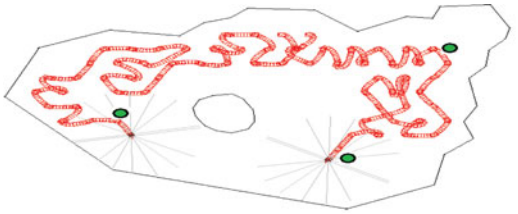
6. Experiments with Real Robots

In this section, the experiment results with the real robots are presented. We use the Pioneer 3DX robots to implement the proposed algorithm. Pioneer 3DX is one of the world's most popular research mobile robots. These robots are equipped with encoders and sonar by which the localization and mapping can be done during the experiments. We do experiment with three robots in a surrounded area. The workspace is an irregularly shaped fenced area of about 16.5 m^2 with a rectangular obstacle of about 0.8 m^2 (see Fig. 16). The goal is searching the whole area by three robots using the proposed algorithm.

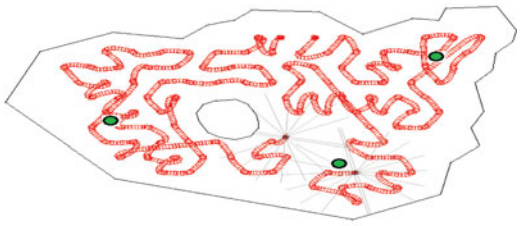
We have used sonar to detect area boundaries, obstacles and other robots. A method of reactive potential field control is applied to avoid collisions with obstacles. In addition, other robots are viewed as obstacles by a robot. We have used ArNetworking, a library provided by Adept MobileRobots for communication between robots. It works with ARIA, the library we have applied to control the robots. The robots exchange the map of the environment that they make during the search operation and the vertices which they visit.

Table IV. Searching time in the experiments (seconds).

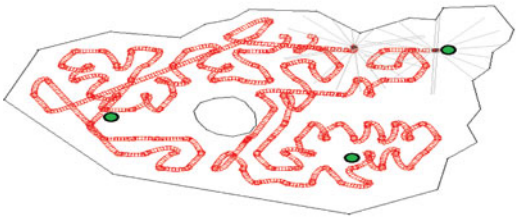
No of robots	**1**	**2**	**3**
Min	57	35	18
Max	66	41	22
Average	61.6	37.8	19.4
STD	4.04	2.39	1.67



(a) Targets are detected in 4m3s



(b) Targets are detected in 6m44s



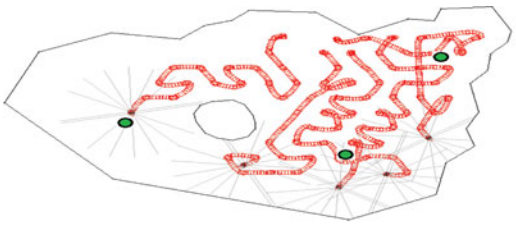
(c) Targets are detected in 7m41s

Fig. 13. Robots' trajectories in the case of searching for three targets by two robots. (a) targets are detected in 4 min 3 s. (b) targets are detected in 6 min 44 s. (c) targets are detected in 7 min 41 s.

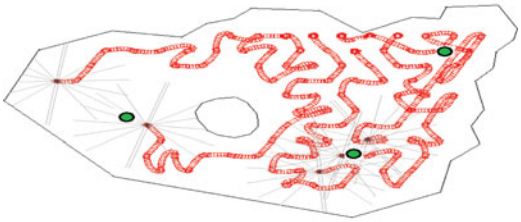
The snapshots of the experiment results are demonstrated in Fig. 15. First, the robots are randomly placed in the field with an obstacle. After applying the algorithm, the robots start to do the first stage of the algorithm, i.e., locating themselves on the vertices of a common triangular grid using consensus variables. Figure 15(a) shows the result of this stage. Then, the robots begin to search the area based on the second stage of the algorithm. Figures 15(b)–(d) are snapshots during this stage. The overall paths taken by the robots in the experiment are depicted in Fig. 16. The results show that the area has been completely explored based on the proposed algorithm. Note that the robots do not necessarily reach the end points at the same time.

Also, to have a better view of a relation between the number of robots and the search duration in the experiments, five runs with one, two and three robots have been done. Table IV displays the calculated values for the minimum, maximum, average and standard deviations of the searching time in the experiments.

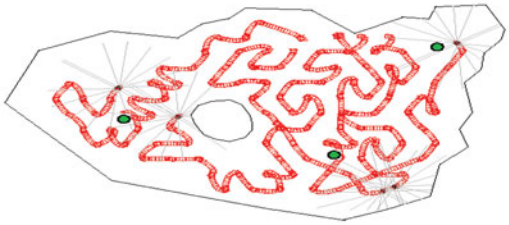
Note that since the area and the triangles sides in the simulations and the experiments are different, and we had only three robots in the experiments, an exact comparison between results is not possible. However, as the Tables I and IV show, the results from simulations and experiments give similar outcomes. For example, in the simulations, three robots explore a 528 m^2 area in about 546 s (with exploring speed of 0.97 m^2/s) and in the experiments three robots explore a 16.5 m^2 area in about 19.4 s (with exploring speed of 0.85 m^2/s). We have used 1 m for the sides of triangles in the



(a) Targets are detected in 1m51s



(b) Targets are detected in 1m58s



(c) Targets are detected in 2m33s

Fig. 14. Robots' trajectories in the case of searching for three targets by five robots. (a) targets are detected in 1 min 51 s. (b) targets are detected in 1 min 58 s. (c) targets are detected in 2 min 33s.

(a) (b)

(c) (d)

Fig. 15. Three robots exploring a whole area.

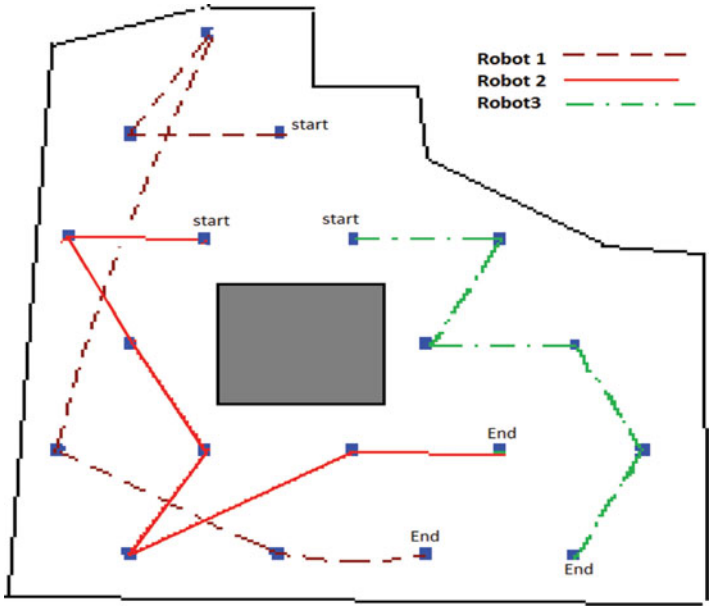


Fig. 16. Robots' trajectories in the experiment.

experiments, while in the simulations, they have been 2 m; therefore, the robots have had more stops in the experiments thus the lower speed of exploring.

7. Conclusions

In this paper, we have developed a distributed control algorithm to drive a multi-robot team to explore an unknown area. We have used a triangular grid pattern and a two-stage algorithm for the control law so that the robots move through the vertices of the grid during the search procedure. Therefore, a complete search of the whole area has been guaranteed. A mathematically rigorous proof of convergence of the presented algorithm has been demonstrated. Furthermore, the computer simulation results using MobileSim, a powerful simulator of real robots and environment, have been presented to demonstrate that the algorithm is effective and practicable. Also, the experiments with Pioneer 3DX wheeled mobile robots have been done to confirm the performance of our suggested algorithm. The presented results of experiments with real robots show that the algorithm is quite practical. During the experiments, no visible or severe drift on wheel odometry had been observed, but it can be significant if the number of vertices of the grid is large. Addressing this problem is a direction for our future work.

Acknowledgement

This work was supported by the Australian Research Council.

References

1. H. H. Viet, V.-H. Dang, M. N. U. Laskar and T. Chung, "Ba*: An online complete coverage algorithm for cleaning robots," *Appl. Intell.* **39**(2), 217–235 (2013).
2. J. Hess, M. Beinhofer and W. Burgard, "A Probabilistic Approach to High-Confidence Cleaning Guarantees for Low-Cost Cleaning Robots," *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Hong Kong (2014) pp. 5600–5605.
3. A. Marjovi and L. Marques, "Multi-robot olfactory search in structured environments," *Robot. Auton. Syst.* **59**(11), 867–881 (2011).
4. M. Guarnieri, R. Kurazume, H. Masuda, T. Inoh, K. Takita, P. Debenest, R. Hodoshima, E. Fukushima and S. Hirose, "Helios System: A Team of Tracked Robots for Special Urban Search and Rescue Operations," *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, IEEE, St. Louis, MO, USA (2009) pp. 2795–2800.

5. G.-J. M. Kruijff, M. Janíček, S. Keshavdas, B. Larochelle, H. Zender, N. J. Smets, T. Mioch, M. A. Neerincx, J. V. Diggelen, F. Colas *et al.*, "Experience in System Design for Human-Robot Teaming in Urban Search and Rescue," **In**: *Field and Service Robotics* (K. Yoshida and S. Tadokoro, eds.) (Springer, 2014) pp. 111–125.
6. D. Portugal and R. P. Rocha, "Multi-robot patrolling algorithms: Examining performance and scalability," *Adv. Robot.* **27**(5), 325–336 (2013).
7. U. Reiser, T. Jacobs, G. Arbeiter, C. Parlitz and K. Dautenhahn, "Care-o-bot® 3–Vision of a Robot Butler," **In**: *Your Virtual Butler* (R. Trappl, ed.) (Springer, 2013) pp. 97–116, 2013.
8. A. Marjovi, J. G. Nunes, L. Marques and A. de Almeida, "Multi-Robot Fire Searching in Unknown Environment," **In**: *Field and Service Robotics* (A. Howard, K. Iagnemma, and A. Kelly, eds.) (Springer, 2010) pp. 341–351.
9. V. Zadorozhny and M. Lewis, "Information Fusion Based on Collective Intelligence for Multi-Robot search and Rescue Missions," *IEEE 14th International Conference on Mobile Data Management*, IEEE, Los Alamitos, CA, USA (2013) pp. 275–278.
10. T. Pereira, A. P. Moreira and M. Veloso, "Coordination for Multi-Robot Exploration using Topological Maps," **In**: *Proceedings of the 11th Portuguese Conference on Automatic Control, CONTROLO2014* (P. A. Moreira, A. Matos, and G. Veiga, eds.) (Springer, Cham, Switzerland, 2015) pp. 515–524.
11. F. F. Carvalho, R. C. Cavalcante, M. Vieira, L. Chaimowicz and M. F. Campos, "A Multi-Robot Exploration Approach Based on Distributed Graph Coloring," *Latin American Robotics Symposium and Competition (LARS/LARC),* IEEE, Arequipa, Peru (2013) pp. 142–147.
12. S. Sharma, C. Sur, A. Shukla and R. Tiwari, "Multi-Robot Area Exploration using Particle Swarm Optimization with the Help of cbdf-Based Robot Scattering," **In**: *Computational Vision and Robotics* (I. K. Sethi, ed.) (Springer, 2015) pp. 113–123.
13. A. V. Savkin, T. M. Cheng, Z. Li, F. Javed, Z. Xi, A. S. Matveev and H. Nguyen, *Decentralized Coverage Control Problems For Mobile Robotic Sensor and Actuator Networks* (Wiley-IEEE Press, New Jersey, USA, 2015).
14. M. O. F. Sarker, T. S. Dahl, E. Arcaute and K. Christensen, "Local interactions over global broadcasts for improved task allocation in self-organized multi-robot systems," *Robot. Auton. Syst.* **62**(10), 1453–1462 (2014).
15. T. M. Cheng and A. V. Savkin, "Decentralized control for mobile robotic sensor network self-deployment: Barrier and sweep coverage problems," *Robotica* **29**(2), 283–294 (2011).
16. P. Chand and D. A. Carnegie, "A two-tiered global path planning strategy for limited memory mobile robots," *Robot. Auton. Syst.* **60**(2), 309–321 (2012).
17. P. Chand and D. A. Carnegie, "Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots," *Robot. Auton. Syst.* **61**(6), 565–579 (2013).
18. A. V. Savkin and H. Teimoori, "Decentralized navigation of groups of wheeled mobile robots with limited communication," *IEEE Trans Robot.* **26**(6), 1099–1104 (2010).
19. L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Auton. Robots* **12**(3), 231–255 (2002).
20. A. Cunningham, K. M. Wurm, W. Burgard and F. Dellaert, "Fully Distributed Scalable Smoothing and Mapping with Robust Multi-Robot Data Association," **In**: *IEEE International Conference on Robotics and Automation (ICRA),* IEEE, Saint Paul, MN, USA (2012) pp. 1093–1100.
21. G. Erinc and S. Carpin, "Anytime merging of appearance-based maps," *Auton. Robots* **36**(3), 241–256 (2014).
22. T.-D. Vu, J. Burlet and O. Aycard, "Grid-based localization and local mapping with moving object detection and tracking," *Inform. Fusion* **12**(1), 58–69 (2011).
23. D. Marinakis and G. Dudek, "Pure topological mapping in mobile robotics," *IEEE Trans. Robot.* **26**(6), 1051–1064 (2010).
24. N. Agmon, S. Kraus and G. A. Kaminka, "Multi-Robot Perimeter Patrol in Adversarial Settings," *IEEE International Conference on Robotics and Automation, ICRA*, IEEE, Pasadena, CA, USA (2008) pp. 2339–2345.
25. L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M. Hsieh, H. Hsu, J. Keller, V. Kumar, R. Swaminathan and C. Taylor, "Deploying Air-Ground Multi-Robot Teams in Urban Environments," **In**: *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III* (L. E. Parker, F. E. Schneider, and A. C. Schult, eds.) (Springer, 2005) pp. 223–234.
26. K. M. Wurm, C. Stachniss and W. Burgard, "Coordinated Multi-Robot Exploration using a Segmentation of the Environment," *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, IEEE, Nice, France (2008) pp. 1160–1165.
27. J. L. Baxter, E. Burke, J. M. Garibaldi and M. Norman, "Multi-Robot Search and Rescue: A Potential Field Based Approach," **In**: *Autonomous Robots and Agents* (S. C. Mukhopadhyay and G. S. Gupta, eds.) (Springer, 2007) pp. 9–16.
28. R. Zlot, A. Stentz, M. Dias and S. Thayer, "Multi-Robot Exploration Controlled by a Market Economy," *IEEE International Conference on Robotics and Automation, ICRA '02*, IEEE, Washington, DC, USA, vol. 3 (2002) pp. 3016–3023.
29. K. Guruprasad and D. Ghose, "Performance of a class of multi-robot deploy and search strategies based on centroidal voronoi configurations," *Int. J. Syst. Sci.* **44**(4), 680–699 (2013).

30. J. Fink, M. A. Hsieh and V. Kumar, "Multi-Robot Manipulation Via Caging in Environments with Obstacles," *IEEE International Conference on Robotics and Automation, ICRA*, IEEE, Pasadena, CA, USA (2008) pp. 1471–1476.
31. A. Baranzadeh, "A Decentralized Control Algorithm for Target Search by a Multi-Robot Team," *Proceedings of Australasian Conference on Robotics and Automation*, ARAA, Sydney, Australia (2013).
32. A. V. Savkin, F. Javed and A. S. Matveev, "Optimal distributed blanket coverage self-deployment of mobile wireless sensor networks," *IEEE Commun. Lett.* **16**(6), 949–951 (2012).
33. R. Kershner, "The number of circles covering a set," *Am. J. Math.* **61**(3), 665–671 (1939).
34. S. Tadokoro, *Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue* (Springer Science & Business Media, London, UK, 2009).
35. R. R. Murphy, J. Kravitz, S. Stover and R. Shoureshi, "Mobile robots in mine rescue and recovery," *IEEE Robot. Autom. Mag.* **16**(2), 91–103 (2009).
36. A. V. Savkin and F. Javed, "A Method for Decentralized Self-Deployment of a Mobile Sensor Network with Given Regular Geometric Patterns," in *Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, IEEE, Adelaide, Australia (2011) pp. 371–376.
37. A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control* **48**(6), 988–1001 (2003).
38. W. Ren, R. W. Beard and E. M. Atkins, "A Survey of Consensus Problems in Multi-Agent Coordination," *Proceedings of the American Control Conference,* IEEE, Portland, Oregon, USA (2005) pp. 1859–1864.
39. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**(1), 90–98 (1986).
40. D. K. Sutantyo, S. Kernbach, P. Levi and V. Nepomnyashchikh, "Multi-Robot Searching Algorithm using Lévy Flight and Artificial Potential Field," *IEEE International Workshop on Safety Security and Rescue Robotics (SSRR),* IEEE, Bremen, Germany (2010) pp. 1–6.