Distributed Safe Deployment of Networked Robots

Reza Javanmard Alitappeh and Luciano C.A. Pimenta

Abstract In real applications, it is always important to consider the generation of safe paths for robots during deployment or in future excursions through the environment. In order to include *safety* in the problem of deploying mobile robotic networks, we propose a new strategy based on the locational optimization framework. Our approach models the optimal deployment problem as a constrained optimization problem with inequality and equality constraints. This optimization model is built by incorporating into the locational optimization framework new features such as the classical Generalized Voronoi Diagram (GVD) commonly used as a safe roadmap in the context of path planning and a new metric to compute distance between robots and points in the environment. This new metric induces a new Voronoi partition of the environment. Furthermore, inspired by the classical Dijkstra algorithm, we present a novel efficient distributed algorithm to compute solutions in complicated environments.

Keywords Mobile robotic network · Locational optimization · Deployment problem · Voronoi partitioning

1 Introduction

According to [4], a system composed of a group of robots that sense their own position, exchange messages following a communication topology, process information, and control their motion is called a robotic network. One can find several applications for this type of system such as surveillance, sensing coverage, environment monitoring, search and rescue, etc. An important question to answer when using a robotic network is where each robot should be placed in the environment. In the

R.J. Alitappeh (✉) · L.C.A. Pimenta
Universidade Federal de Minas Gerais, Av. Antonio Carlos 6627, Belo Horizonte, MG 31270-901, Brazil
e-mail: rezajavanmard64@gmail.com

L.C.A. Pimenta
e-mail: lucpim@cpdee.ufmg.br

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_5

present work we show a distributed solution to this question which is refereed as the deployment problem [4]. The solution is distributed in the sense that each agent depends only on information from a small set of other agents called neighbors to compute its actions. Besides, this set of neighbors is dynamic since it might change as the system evolves. As pointed by [7], this allows for scalability and robustness. We are interested in finding optimal deployment configurations. We consider that a configuration is optimal if it is a minimizer of a functional encoding the quality of the deployment. This quality of deployment is related to the time of response of the network after an event that needs servicing happens in the environment. This time is a function of the distance of the agents from the event and the agent capabilities (speed, sensor field of view, etc.). In order to minimize the distance between agents and events, our approach applies the idea of partitioning the environment into subregions which are then assigned to specific agents. Therefore, each agent is responsible for attending the events in its corresponding subregion. Differently from previous works found in the literature we are also concerned with the incorporation of some safety constraints into the deployment. This property guarantees such a safe movement for the robots, with maximum distance from the obstacles, in the environment. The paper is organized as follows: in the next section we present some related work. In Sect. 3 we present some useful tools that will be considered in the rest of the paper. The proposed optimization model and an efficient distributed solution to the problem are explained in Sects. 4 and 5. Implementation results are shown in Sect. 6. Finally, conclusions are presented in Sect. 7.

2 Related Work

Our approach builds on the work in [7]. The authors of this work present a distributed and asynchronous approach for optimally deploying a uniform robotic network in a domain based on a framework for optimized quantization derived in [11]. Each agent (robot) follows a control law, which is a gradient descent algorithm that minimizes the functional encoding the quality of the deployment. Further, this control law depends only on the information of position of the robot and of its immediate neighbors. Neighbors are defined to be those robots that are located in neighboring Voronoi cells. Besides, these control laws are computed without the requirement of global synchronization. The functional also uses a *distribution density function* which weights points or areas in the environment that are more important than others. Thus it is possible to specify areas where a higher density of agents is required. This is important if events happen in the environment with different probabilities in different points. Furthermore, this technique is adaptive due to its ability to address changing environments, tasks, and network topology. Different extensions of the framework devised in [7] have been proposed in the literature. The problem of considering time-varying distribution density functions was studied in [13] to solve a task of simultaneous coverage and intruders tracking. Deployment and exploration in non-convex environments was considered in [3, 5, 10]. In [12], heterogeneous robots in

a non-convex environments were taken into account. Where, instead of point robots, disc shaped robots were considered. Some works also considered the discretization of the environment by grid cells to facilitate computation in complex environments. In [9] the authors consider a discrete partitioning and coverage optimization algorithm for robots with short-range communication. In this case a discrete setup was presented in which a discrete deployment functional is defined. The authors proved that their algorithm converges to a subset of the set of centroidal Voronoi tessellations (CVT) in discrete formulation, named pairwise-optimal partition. Gossip communication was used to allow information exchange among the agents. Similarly, [14] describe an algorithm to solve the deployment problem in a discrete setup. In [2] the environment was also discretized to allow the numerical computation of the environment partition (geodesic Voronoi diagram), but in this case the context was the one of generating an approximation to the continuous setup. In the same spirit of approximating the continuous setup, the authors of [1] discretized the environment and used a graph based approach inspired by Dijkstra algorithm [8] to directly compute the proposed control law in an efficient manner in general Riemannian manifolds with boundaries.

Statement of Contributions: The present paper further extends the works in [1, 12] to include safety. By merging different Voronoi diagrams, including the well known Generalized Voronoi Diagram (GVD) [6] (traditionally used as a roadmap in path planning) and by considering a constrained optimization problem in the context of the Locational Optimization Framework, we can generate safe routes for the robots during deployment and also after deployment when servicing a given point of the environment. We propose a new Voronoi Diagram which is built according to a new metric that takes into account shortest paths that traverse the GVD. Moreover, in order to consider real world environments we devise a new efficient algorithm to compute the next actions of the robots in the same spirit of the one in [1].

3 Background

In this section we explain the basic tools which will allow us to define our deployment problem in terms of a constrained optimization problem. These tools are the GVD and the locational optimization framework.

3.1 Generalized Voronoi Diagram

Let the set of obstacles $\mathbf{QO} = \{\mathcal{QO}_1, \ldots, \mathcal{QO}_n\}$ in a planar configuration space be called sites. This set induces a structure called Generalized Voronoi Diagram (GVD). $\mathcal{Q}$ indicates to configuration space and a set of points in the free configuration space, $\mathcal{Q}_{free}$, is defined as the Voronoi region of the obstacle $\mathcal{QO}_i$, V_i, if these points are

closer to $\mathcal{QO}_i$ than to all the other sites. Given an obstacle $\mathcal{QO}_i$, the generalized Voronoi region, V_i, is the closure of the set of points closest to $\mathcal{QO}_i$ [6].

$$V_i = \{\mathbf{q} \in \mathcal{Q}_{free} |\, d(\mathbf{q}, \mathcal{QO}_i) \leq d(\mathbf{q}, \mathcal{QO}_j), \forall j \neq i\}\,, \tag{1}$$

where $d(\mathbf{q}, \mathcal{QO}_i)$ shows the minimum distance between $\mathcal{QO}_i$ and $\mathbf{q}$. The two-equidistant surjective surface, $\mathcal{L}_{i,j}$ is the set of points equidistant to two obstacles $\mathcal{QO}_i$ and $\mathcal{QO}_j$ with distinct gradient vectors:

$$\mathcal{L}_{i,j} = \{\mathbf{q} \in \mathcal{Q} | d(\mathbf{q}, \mathcal{QO}_i) = d(\mathbf{q}, \mathcal{QO}_j) \text{ and } \nabla d(\mathbf{q}, \mathcal{QO}_i) \neq \nabla d(\mathbf{q}, \mathcal{QO}_j), j \neq i\}. \tag{2}$$

The points in $\mathcal{L}_{i,j}$ that are part of the GVD are those in which $\mathcal{QO}_i$ and $\mathcal{QO}_j$ are the closest obstacles. Therefore we can define the set:

$$V_{i,j} = \{\mathbf{q} \in \mathcal{L}_{i,j} |\, d(\mathbf{q}, \mathcal{QO}_i) \leq d(\mathbf{q}, \mathcal{QO}_h)\}. \tag{3}$$

This last definition allows us to formally define the GVD:

$$GVD = \bigcup_i \bigcup_j V_{i,j}. \tag{4}$$

An interesting feature of the GVD is that it can be used as a roadmap (RM) for path planning (Fig. 1a).

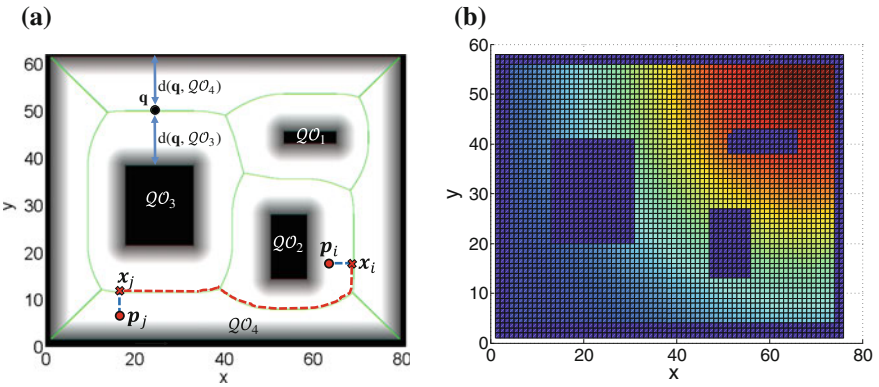

Fig. 1 A map with obstacles, $\mathcal{QO}_i$, GVD and a simple path on the GVD, to show its property to be used as a roadmap and a density function which centered at top-right of the map. **a** *Green line* shows the GVD. **q** is an equidistant point between sites $\mathcal{QO}_3$ and $\mathcal{QO}_4$. *Dash line* illustrates a path between two arbitrary points, (p_i, p_j). **b** An example of Gaussian density function in a 2D map. $A = 7,\ (x_0, y_0) = (67, 54),\ \sigma_x = \sigma_y = \sqrt{30}$

3.2 Locational Optimization Based Deployment

Let $\Omega \subset \mathbb{R}^2$ be the map of the environment. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the configuration of n robots, and $f(d(\mathbf{q}, \mathbf{p_i}))$ indicates the cost of servicing an event at point $\mathbf{q}$ by robot i. This is related to the spatial distance between $\mathbf{q}$ and $\mathbf{p}_i$ as $d(\mathbf{q}, \mathbf{p}_i)$ represents this distance and f is a smooth strictly increasing function. Suppose we have access to a density function $\phi : \Omega \to \mathbb{R}_+$ which gives weights to different points in Ω to reflect the probability of having events at each point (See Fig. 1b). Considering also the tessellation $W = \{W_1, \ldots, W_n\}$ so that $\cup_{i=1}^{n} W_i = \Omega$ and $I(W_i) \cap I(W_j) = \emptyset$, $\forall i \neq j$, where $I(\cdot)$ represents the interior of a given region, it is possible to define the following *deployment functional* that measures the quality of the robotic deployment [7]:

$$\mathcal{H}(P, W) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i, W_i) = \sum_{i=1}^{n} \int_{W_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q}\,, \tag{5}$$

The objective of the Locational Optimization based framework is to drive the robots to a configuration that minimizes (5). In [1], it is considered the case where f is the square function and d is the Euclidean distance. The authors of [1] proposed a distributed control law that guides these robots to the minimum which coincides with the so-called Centroidal Voronoi Tessellation (CVT).

In the present work, we further extend this framework to incorporate safety.

4 Safe Deployment Modeling

In this section we define the safe deployment problem as an optimization problem in the context of the Locational Optimization Framework. Consider the bounded free configuration space $\mathcal{Q}_{free} \subset \mathbb{R}^2$. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the configuration of n robots, where $\mathbf{p}_i \subset \mathcal{Q}_{free}$. The problem to be solved is the one of finding distributed robotic actions, in the sense that only robots in the neighborhood of robot i will be taken into account, that leads the system to a local solution of the constraint minimization problem given below:

$$\min_{\mathbf{p}_i} \mathcal{H}(P, GGVD), \qquad \text{s.t.} \begin{cases} y_{i1}(\mathbf{p}_i) \leq 0\,, \ldots\,, y_{im}(\mathbf{p}_i) \leq 0 \\ h(\mathbf{p}_i) = 0 \end{cases} \tag{6}$$

where $y()$ and $h()$ declare inequality and equality constraints respectively. The next sections will explain the meaning of the terms used in the defined problem and from this explanation it should be clear how safety is then incorporated in the locational optimization framework.

New Metric: The Geodesic distance is a metric which is more realistic than Euclidean distance in non-convex environments. This distance is used in the deployment functional presented in [12] as the general function d, as defined in the last section. In this case, the induced Voronoi Tessellation is the so-called geodesic Voronoi Tessellation. Now, we propose a further extension on this metric which will be called *Geodesic Distance Based* on GVD. This distance function corresponds to the length of the shortest path from two points when using a GVD as a roadmap. A clear example of this path is shown in Fig. 1a, where dash line between a pair (p_i, p_j) defines the whole path: $\{(p_i, x_i), (x_i, x_j), (x_j, p_j)\}$. In general, we can divide this path into three parts: a path from the initial point to GVD ($Path_{\ Init\_To\_GVD}$), a path from a point on GVD to another point on GVD ($Path_{\ GVD\_To\ GVD}$), and a path from GVD to the goal point ($Path_{\ GVD\_To\_Goal}$). The *Geodesic Distance Based on GVD* is then defined as:

$$\begin{aligned} d(\mathbf{p}_i, \mathbf{p}_j) \;=\; & W_1.||\mathbf{p}_i - \Pi_i(GVD)|| + W_2.g(\Pi_i(GVD), \Pi_j(GVD)) \\ & + W_1.||\mathbf{p}_j - \Pi_j(GVD)||\,, \end{aligned} \tag{7}$$

where $g(x_i, x_j)$ gives the shortest distance between two points x_i and x_j on the GVD, if the motion is constrained to remain on the GVD, $\Pi_i(GVD)$ represents the projection of the point $\mathbf{p}_i$ onto the GVD which corresponds to the closest point on the GVD to $\mathbf{p}_i$, and W_1 and W_2 are the weights of each part of the path. For example, by assigning a big value to W_1 the cost of $Path_{\ Init\_To\_GVD}$ or $Path_{\ GVD\_To\_Goal}$ can be increased. These weights help to adjust the cost of two portions of the path so that it is worth first moving to the GVD as soon as possible and perform most of the motion traversing it. As safety regarding the existing obstacles is related to the distance the robot keeps from them and the GVD provides a roadmap which keeps equidistance from the closest obstacles, we can say that this metric can introduce safety in the deployment solution. In the minimization problem defined in (6) the cost function is defined according to the new metric, d:

$$\mathcal{H}(P, GGVD) = \sum_{i=1}^{n} \int_{GGVD_i} d(\mathbf{q}, \mathbf{p}_i)^2 \phi(\mathbf{q}) d\mathbf{q}\,, \tag{8}$$

where *GGVD*, (Geodesic Generalized Voronoi Diagram) will be defined as the Voronoi Tessellation induced by the new metric and $W_1 >> W_2$.

Now, we can also describe the equality constraint $h(\mathbf{p}_i) = 0$. This function is defined as the difference between the distance functions $d(\mathbf{p}_i, \mathcal{QO}_i)$ and $d(\mathbf{p}_i, \mathcal{QO}_j)$ in which $\mathcal{QO}_i$ and $\mathcal{QO}_j$ are the closest obstacles to robot i. Thus, this means the robots must be deployed along the GVD.

Collision avoidance: Since the focus of this work is on safety, besides the static obstacles we should also take into account the possible collisions between robots. A practical problem of the unconstrained minimization executed by the pure gradient-descent law in [7] is that actual robots are not point-robots. Thus, we propose to use here the same strategy presented in [12]. Basically, in this work the basic results for point robots are extended to robots that can be modeled as circular disks, each one

with radius $r_{\mathbf{p}_i} = r_{\mathbf{p}_j}, \forall i, j$. This is done by incorporating the inequality constraints $y_{ik} \leq 0$ in (6), so that the robots remain inside their so-called free Voronoi region. For details refer to [12].

5 Proposed Solution

In order to solve the safe deployment problem in an efficient manner we propose to solve a discrete approximation of the continuous setup shown in the last section. The proposed algorithm builds upon the work in [1] which presents a modified Dijkstra algorithm able to compute simultaneously at each iteration the geodesic Voronoi diagram and the robot next actions in the case of deployment on Riemannian manifolds with boundaries.

5.1 Discrete Approximation

Consider the 2-dimensional configuration space. The graph $G = \{\mathcal{V}(G), \mathcal{E}(G), \mathcal{C}_G\}$ is induced from the uniform square tiling of the configuration space by considering an 8-connectivity neighborhood (See Fig. 2a). The set of vertices (nodes) is given by $\mathcal{V}(G)$, the set of edges by $\mathcal{E}(G) \subseteq \mathcal{V}(G) \times \mathcal{V}(G)$, and a cost function is denoted by $\mathcal{C}_G : \mathcal{E}(G) \rightarrow \mathbb{R}^+$. It is important to mention that a node of the graph is placed in grid cells located inside $\mathcal{Q}_{free}$. Moreover, the cost of each edge is computed based on the defined new metric as will be clarified later. We will also use the notation p_i to denote the node that contains the position of robot i, $\mathbf{p}_i$, and the operator $\mathbf{P}(s)$ to

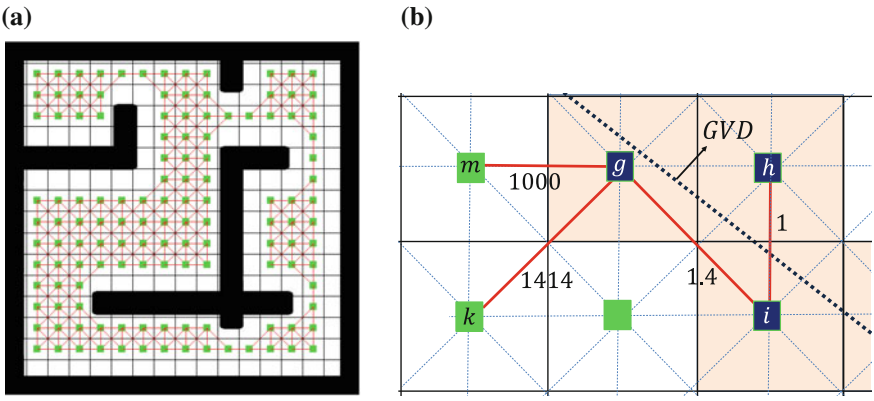

Fig. 2 Representing a discrete grid based map and the technique of modeling GVD based on graph nodes and edges. **a** Discretization and graph representation. **b** The nodes g, h, and i are in the set $\mathcal{V}GVD(G)$

return the position of the center of the grid cells. Therefore, $\mathbf{P}(p_i)$ returns the center of the cell that contains robot i. Furthermore, we will use $\mathcal{N}_G(p_i)$ as the set of graph vertex neighbors: $\mathcal{N}_G(p_i) = \{q \in \mathcal{V}(G) \mid [p_i, q] \in \mathcal{E}(G)\}$.

We compute the GVD before discretizing the environment into cells, allowing the GVD to be independent from the discretization resolution. The GVD is embedded in our graph by labeling the set of grid cells that contain a piece of the GVD as the approximate GVD, $\mathcal{V}GVD(G)$. Now, we can define the edge cost function:

$$\mathcal{C}_G(i,j) = \begin{cases} W_2 \cdot c(i,j), & \text{if } i\,, j\ \in \mathcal{V}GVD(G) \\ W_1 \cdot c(i,j), & \text{otherwise}\,, \end{cases} \tag{9}$$

where $c(i, j)$ is given by the Euclidean distance between the centers of the cells i and j. Since it is our objective to deploy and also move the robots along the GVD we will use $W1 >> W2$. For instance, see an example in Fig. 2b.

The shortest path between two vertices s and q corresponds to the sequence of nodes (consequently edges), $\{s, v_1, v_2, \ldots, v_m, q\}$, connecting this pair such that the sum of the edge costs is minimum. We will define this minimum cost sum as $d^*(\mathbf{P}(s), \mathbf{P}(q))$:

$$d^*(\mathbf{P}(s), \mathbf{P}(q)) = \mathcal{C}_G(s, v_1) + \mathcal{C}_G(v_1, v_2) + \ldots + \mathcal{C}_G(v_m, q). \tag{10}$$

This allows us to define the discrete version of the deployment functional:

$$\mathcal{H}^* = \sum_{i=1}^{n} \sum_{q \in GGVD_i^*} d^*(\mathbf{P}(q), \mathbf{P}(p_i))^2 \phi(\mathbf{P}(q)) \bar{w}\,, \tag{11}$$

where $GGVD_i^*$ corresponds to the set of grid cells so that $d^*(\mathbf{P}(q), \mathbf{P}(p_i))$ is less than $d^*(\mathbf{P}(q), \mathbf{P}(p_j))$, $\forall j \neq i$, and $\bar{w}$ is a constant related to the integral element of area. Assuming that the robots are located at the center of the grid cells, i.e. $\mathbf{P}(p_i) = \mathbf{p}_i$, we can compute the gradient of $\mathcal{H}^*$:

$$\frac{\partial \mathcal{H}^*}{\partial \mathbf{p}_i} = \sum_{i=1}^{n} \sum_{q \in GGVD_i^*} 2\mathbf{z}_{p_i,q} d^*(\mathbf{P}(q), \mathbf{P}(p_i)) \phi(\mathbf{P}(q)) \bar{w}\,, \tag{12}$$

where $\mathbf{z}_{p_i,q}$ is the vector with direction given by the first edge of the shortest path between p_i and q, i.e. the direction of $\mathbf{P}(p_i) - \mathbf{P}(v_1)$, and magnitude given by W_2 if $p_i\,, v_1\ \in \mathcal{V}GVD(G)$ or W_1 otherwise. Based on last equation we propose a gradient descent based approach in the next subsection.

5.2 Distributed Algorithm

The problem defined in (6) has some constraints, which means that our solution should also take these constraints into account to define the next action. The collision avoidance inequalities are implemented by first verifying if any of these constraints are active, i.e., $y_{ik} = 0$ for some k. If this is the case, it means there are at least two robots in the imminence of a collision, thus the involved robots will be allowed to move only if the desired direction of motion is orthogonal or has a negative projection onto the segment joining the two robot centers. The equality constraint which enforces the robots to be deployed along the GVD is imposed in our solution by means of two steps. If a robot is not in a cell that is part of the $\mathcal{V}GVD$ the next action for this robot is to move towards the closest cell in $\mathcal{V}GVD$ which is not occupied by any other robot at that time. This can be considered as the first step of the proposed approach. The second step is activated when the robot enters a cell which is part of the $\mathcal{V}GVD$. Now, the next action of this robot is a motion along a straight line from the current grid cell to a neighbor cell which is also part of the $\mathcal{V}GVD$. This next cell is computed based on the gradient descent direction given by the negative of the expression in (12). As in [1] we present an algorithm that computes the gradient descent direction and the Voronoi tessellation, $GGVD^*$, simultaneously in every time-step by means of a wavefront propagation procedure similarly to the process in Dijkstra's algorithm [8]. The wavefront in a given iteration represents the set of points equidistant to the start node also called source. In our case we consider wavefronts emanating from multiple sources (given by the locations of the robots). As the wavefronts propagate two operations are executed: (i) graph vertices in the wavefronts are associated to robots (sources) at shortest distance (according to the proposed metric) giving rise to the Voronoi regions; and (ii) terms of the summation in (12) associated to vertices in the wavefronts are added to a variable responsible to store the gradient descent direction. The places where the wavefronts collide determine the Voronoi boundaries. The ideas previously discussed are organized in the form of the Algorithms 1 and 2. We consider these are the algorithms running in robot i.

Termination: The commands in while loop in Algorithm 1 are executed until termination criteria are met. An interesting criteria is the observation of the variation of positions of robots in the most recent iterations. If this variation is below a given threshold the algorithm can terminate.

Complexity: It is clear that the bottleneck of our iterative algorithm is the function described in Algorithm 2. Since this function runs exactly in the same format of the Dijkstra algorithm, the graph vertices have a constant degree, and a heap is maintained as a priority queue to store the unvisited nodes, the running time is given by $O(V_G \log(V_G))$ (where V_G is the number of vertices in the graph).

Algorithm 1: Distributed main algorithm running in robot i.

Input: $G, \mathcal{VGVD}, \phi, p_i$
G is the graph, $\mathcal{VGVD}$ is the approximate generalized Voronoi diagram, ϕ is the density function, $p_i \in \mathcal{V}_G$ is robot i initial location (graph node).
Output: p_i, o
p_i is robot i final location and $o : \mathcal{V}_G \rightarrow \{1, 2, \ldots, n\}$ is the discrete tessellation map $GGVD^*$ as computed by robot i.

while *(Termination criteria is not met)* **do**
  Broadcast position $\mathbf{p}_i$ //Robot i sends its position to other robots.
  $\mathcal{N}_i \leftarrow$ *Get_Location_of_Neighbor_Robots*() //Robot i receives location of other robots.
  $\mathcal{N}_i^* \leftarrow \mathcal{N}_i \cap \mathcal{VGVD}$ // Set of neighbors already in the $\mathcal{VGVD}$.
  if $p_i \notin \mathcal{VGVD}$ **then** //check if the robot is not on the $\mathcal{VGVD}$.
    Set the current direction of motion as the one towards the closest cell in $\mathcal{VGVD}$ which is not occupied by another robot
  else
    Call *Modified_Dijkstra*$(G, \mathcal{VGVD}, \phi, p_i, \mathcal{N}_i^*)$ //Compute both the next action (cell) p_i' and the $GGVD^*$ as seen by robot i.
    Set the current direction of motion to reach p_i'
  if *(There is no active inequality constraint) OR (There is an active inequality constraint AND current direction of motion is not obstructed by another robot* **then** //collision avoidance constraint.
    Move according to the current direction of motion
  else
    Stop

6 Results

In this section we illustrate our approach by simulating the deployment of robots in two different environments. Videos are available at:
http://www.cpdee.ufmg.br/~coro/movies/DARS2014/

Simple map: A simple room with some obstacles and size 3.79×2.91 m (or image with 728×582 pixels) is the input map. By using a discretization resolution equal to 10 pixels, we have a grid map with 72×58 cells. Discretization rate can be defined based on the real size of robot and map. Density function is defined by a Gaussian function with parameters: $(x_0, y_0) = (67, 54)$, $\sigma_x = \sigma_y = \sqrt{30}$. In this experiment 5 robots are considered. By observing robots' movement during deployment, it is evident at the beginning, two robots have a large Voronoi region when other robots do not have it (See Fig. 3b, c). After some iterations the decrease/increase of size of large/small Voronoi regions contribute to minimize the cost function as it is shown in Fig. 3a.

Office-like map: In the second experiment, the method was tested on a more complicated map with size 40.0×60.0 m and grid graph size of 80×120. Initially, some of the robots are on the GVD and others are not. We define density Gaussian function as: $(x_0, y_0) = (10, 110)$, $\sigma_x = \sigma_y = \sqrt{50}$. Because of the large input map,

Algorithm 2: *Modified_Dijkstra()function.*

Input: $G, \mathcal{VGVD}, \phi, p_i, \mathcal{N}_i^*$
G is the graph, $\mathcal{VGVD}$ is the approximate generalized Voronoi diagram, ϕ is the density function, $p_i \in \mathcal{V}_G$ is the current robot i location (graph node), and $\mathcal{N}_i^*$ is the set of locations of other robots already in $\mathcal{VGVD}$.
Output: p_i', o
p_i' is the next cell for robot i and $o : \mathcal{V}_G \rightarrow \{1, 2, \ldots, n\}$ is the discrete tessellation map $GGVD^*$ as computed by robot i.

Initiate d^*: $d^*(v) \leftarrow \infty$, for all $v \in \mathcal{V}(G)$ // New metric distance.
Initiate o: $o(v) \leftarrow -1$, $\forall v \in \mathcal{V}(G)$ // Tessellation.
Initiate η: $\eta(v) \leftarrow \emptyset$, $\forall v \in \mathcal{V}(G)$ // robot graph vertex neighbor. $\eta : \mathcal{V}(G) \rightarrow \mathcal{V}(G)$
$\mathbf{I}_i \leftarrow \mathbf{0}$ // The gradient descent of the discrete functional.
foreach $i \in p_i \cup \mathcal{N}_i^*$ **do**
    $d^*(p_i) \leftarrow 0$
    $o(p_i) \leftarrow i$
    foreach $q \in \mathcal{N}_G(p_i)$ **do** // For each graph vertex neighbor of p_i
        $\eta(q) \leftarrow q$
$Q \leftarrow \mathcal{V}(G)$ // Set of unvisited nodes.
while $(Q \neq \emptyset)$ **do**
    $q \leftarrow \arg\min_{q' \in Q} d^*(q')$ // Maintained by a heap data-structure.
    $Q \leftarrow Q - q$ // Remove q from Q
    $k \leftarrow o(q)$
    $s \leftarrow \eta(q)$
    if $(s \,!= \emptyset)$ *AND* $(k == p_i)$ **then** // Equivalently, q is not a vertex occupied by a robot and $q \in GGVD_i^*$.
        $\mathbf{I}_i \leftarrow \mathbf{I}_i + \phi(q) \times d^*(q) \times (\mathbf{P}(s) - \mathbf{P}(p_i))$
    foreach $w \in \mathcal{N}_G(q)$ **do** // For each graph vertex neighbor of q
        $d' \leftarrow d^*(q) + \mathcal{C}_G(q, w)$ //relaxation.
        if $d' < d^*(w)$ **then**
            $d^*(w) \leftarrow d'$
            $o(w) \leftarrow k$
            if $(s \,!= \emptyset)$ **then**
                $\eta(w) = s$
$p_i' \leftarrow \arg\max_{u \in \mathcal{N}_G(p_i) \cap \mathcal{VGVD}} \frac{(\mathbf{P}(u) - \mathbf{P}(p_i))}{\|\mathbf{P}(u) - \mathbf{P}(p_i)\|} \cdot \mathbf{I}_i$ // Choose next action as the cell in $\mathcal{VGVD}$ best aligned with the gradient descent direction.

we consider three groups with two robots in each one. They start their movement from three different parts of the map. Figure 4a shows the final positions. Figure 4b illustrates robot trajectories and the evolution of the deployment functional, which is minimized as desired is depicted in Fig. 4c.

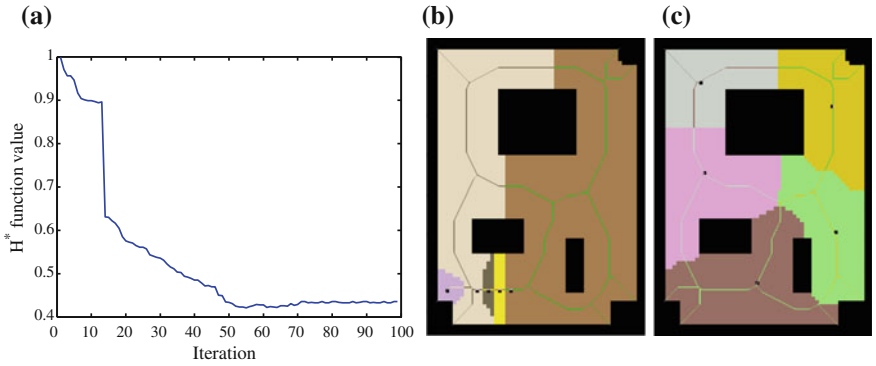

Fig. 3 Snapshots when running the proposed algorithm for 5 robots. **a** $\mathcal{H}^*$ function converged after 65 iterations. **b** *iter*1. **c** *iter*65

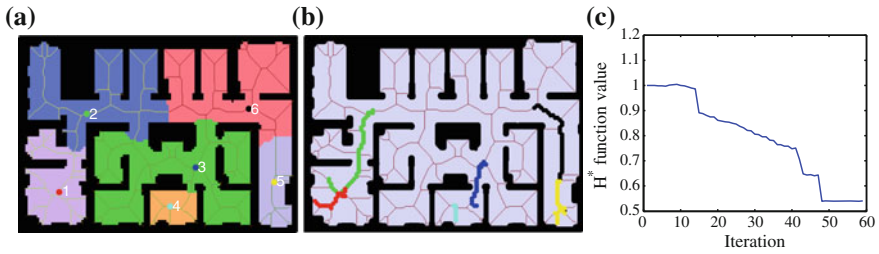

Fig. 4 Result of running the algorithm on office-like map. **a** *iter*50. **b** Robot trajectories in the office-like environment. **c** $\mathcal{H}^*$ function converged after 50 iterations

7 Conclusion

We addressed the problem of deriving optimal distributed control laws to deploy robotic networks in complex environments safely. The deployment problem is translated to a constrained optimization problem so that a deployment functional defined with the use of a new distance function must be minimized while satisfying constraints of two types: (i) inequality constraints for inter-robot collision avoidance, and (ii) an equality constraint to enforce the robots to be deployed at the generalized Voronoi diagram of the environment for maximizing distance from static obstacles. It is also interesting to mention that the proposed framework can also be used with other roadmaps different from the GVD. We presented a distributed algorithm strongly based on the one proposed in [1] which allows for efficient computation of a discrete solution for the discrete approximation of the problem. Simulations were also presented to illustrate the proposed method performance.

Acknowledgments We gratefully acknowledge support from CNPq and FAPEMIG.

References

1. Bhattacharya, S., Ghrist, R., Kumar, V.: Multi-robot coverage and exploration on Riemannian manifolds with boundaries. Int. J. Robot. Res. **33**(1), 113–137 (2013)
2. Bhattacharya, S., Michael, N., Kumar, V.: Distributed coverage and exploration in unknown nonconvex environments. In: Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems, pp. 1–14. Springer (2010)
3. Breitenmoser, A.: Voronoi coverage of non-convex environments with a group of networked robots. In: The IEEE International Conference on Robotics and Automation (ICRA), pp. 4982–4989. IEEE, Anchorage, Alaska (2010)
4. Bullo, F., Cortés, J., Martinez, S.: Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms. Applied Mathematics Series. Princeton University Press, Princeton (2009)
5. Caicedo-Nunez, C.H., Zefran, M.: A coverage algorithm for a class of non-convex regions. In: Proceeding of the IEEE Conference on Decision and Control, pp. 4244–4249 (2008)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, Boston (2005)
7. Cortes, J., Martinez, S., T.K., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
8. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische mathematik **1**(l 959), 269–271 (1959)
9. Durham, J.W., Carli, R.: Discrete partitioning and coverage control for gossiping robots. IEEE Trans. Robot. **28**(2), 364–378 (2012)
10. Haumann, D., Breitenmoser, A., Willert, V., Listmann, K., Siegwart, R.: DisCoverage for non-convex environments with arbitrary obstacles. In: Proceeding of IEEE International Conference Robotics Automation, pp. 4486–4491 (2011)
11. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
12. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In: IEEE Conference on Decision and Control 2, pp. 3947–3952. IEEE, Cancun, Mexico (2008)
13. Pimenta, L.C.A., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R.C., Pereira, G.A.S.: Simultaneous coverage and tracking (SCAT) of moving targets with robot networks. In: Algorithmic Foundation of Robotics VIII, Springer Tracts in Advanced Robotics **57**, 85–99 (2010)
14. Yun, S.k., Rus, D.: Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. Robotica **32**(02), 257–277 (2013)