Deployment of Heterogeneous Robotic Networks in Non-Convex Environments

Luciano C. A. Pimenta, Subhrajit Bhattacharya, and Vijay Kumar

***Abstract*—We address the problem of deploying heterogeneous robotic networks in complex environments. Our work builds on the location optimization framework in [1], [2], with significant extensions. First, we consider robots with different capabilities, allowing, for example, aerial and ground vehicles to collaborate. Second, we extend the previous work allowing for deployment in non-convex environments. Third, we allow for finite size (non-point) robots which enables implementation on real robotic systems. Lastly, we present a novel discrete efficient algorithm which allows for computing the proposed distributed control law in real environments.**

I. Introduction

According to [3], 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. In the present work we present a solution to the problem of optimally placing a robotic network in an environment. In [3] this is refereed as the deployment problem.

We consider that the deployment 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.

We present a general framework which can be used to optimally deploy a heterogeneous robotic network in a environment with complex geometry without collisions. Robots are heterogeneous in the sense that they might have different capabilities.

A possible application of the proposed strategy is efficient sensory coverage of a complex office-like environment with a heterogeneous robotic network equipped with sensors. Robots can have different speeds of traversal and each sensor can have a different field of view. Robots with larger field of view sensors can cover a larger area for the same robot speed, and faster robots can similarly cover larger areas than slower robots with the same sensors in the same time interval. Our algorithm allows the optimal deployment of this heterogeneous network in this complex environment by assigning different weights to different agents and using the geodesic distance instead of Euclidean distance. Robots able to cover larger areas are assigned larger weights.

This work is part of the CNPq/NSF cooperation program, grant no 49.0743/2006-4. It was partially supported by FAPEMIG (Brazil), CNPq (Brazil), CAPES (Brazil), FINEP (Brazil).

L. C. A. Pimenta is with Department of Electronic Engineering, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG 31270-901 Brazil. E-mail: lucpim@cpdee.ufmg.br. S. Bhattacharya and V. Kumar are with the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104 USA. E-mail: {subhrabh, kumar}@seas.upenn.edu.

Related Work

Our approach builds on the work in [2]. 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 [1]. 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 [2] have been proposed in the literature. In [4] the problem of limited-range interaction between agents was addressed. Anisotropic sensor models were incorporated in [5] and [6]. In [7], constraints were added to the minimization problem to deal with agents with heterogeneous resource capabilities and a modified deterministic annealing algorithm was used to overcome local minima issues in the context of UAV mission planning. The problem of learning the distribution density function online while moving toward the optimal locations was addressed in [8]. In [9], the authors considered the deployment of unicycles by incorporating tools from hybrid systems modeling. The problem of considering time-varying distribution density functions was studied in [10] to solve a task of simultaneous coverage and intruders tracking.

The authors of [11] considered the use of generalized Voronoi diagrams such as the power diagram in the environment partitioning to take into account agents with limited

energy to move. Power diagrams were also used in [12] and [13] to enforce some properties in the subregions that are assigned to the agents. In [12] the subregions are constrained to certain values of area and in [13] the subregions are computed to provide an equitable partition of convex environments according to a given measure. Deployment in non-convex environments was considered in [14], [15], and [16].

Some works also considered the discretization of the environment by grid cells to facilitate computation in complex environments. In [17] 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, [18] describe an algorithm to solve the deployment problem in a discrete setup. In [19] 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.

Statement of Contributions

This paper is an extended version of our conference paper [20]. In the conference paper we presented three novel and important extensions to the basic approach in [2] for the first time. In the present paper we present further details on the development of the proposed extensions and we show how the independent solutions can be used together. We also present a novel algorithm to efficiently compute a discrete approximation of the proposed control law in the case of non-convex environments.

In the first extension we address the problem of incorporating heterogeneity in the robot team by allowing the robots to have different capabilities. This is done by using a Power diagram to generate a tessellation of the environment instead of using a traditional Voronoi diagram. We would like to emphasize that this is a minor contribution since the Power diagram has already been used in other works with different motivations as in [11], [12], and [13]. Second, we generalize the basic method to non-convex environments. Other works such as [14], [15], and [16] also considered non-convex domains. However, differently from other works we solve this issue by using the geodesic distance instead of Euclidean distance in the deployment functional. Therefore, our approach is a natural extension of the original approach to more general geometries. In the third extension we overcome the practical limitations of the point robot assumption in the original algorithm by adding constraints to the minimization problem. In this paper, we also propose to use these three extensions together to address the general problem of deployment of heterogeneous, non-point agents in non-convex environments.

Finally, we also overcome an important practical limitation of our approach which is the difficulty in the computation of a geodesic Voronoi diagram in real environments [21]. To address this problem, we present an approximate solution which is based on the discretization of the environment. As previously mentioned, the basic idea of discretizing the environment had already been used in other works such as [17] and [18]. In these two papers the authors consider solutions to the problem formulated in a discrete setup. Differently, in the present paper we are still interested in the continuous solution and the discretization is used to approximate a solution. This was also done in our previous work [19] in which a search based algorithm for finding the geodesic Voronoi diagram was proposed. In this case the environment was uniformly discretized and a graph was created. Each vertex of the graph corresponded to a cell of the discretization with edges to admissible neighbors. The diagram was then generated in two steps: (i) in the first step a Dijkstra's search [22],[23] was run to each robot; (ii) in the second step the robots compared the multiple Dijkstra results to determine their corresponding Voronoi cell. In the present work we improved this initial idea.

Our algorithm also considers the same discretized environment but now it computes directly the proposed control law while the Voronoi diagram is still being computed. This is done by propagating a wavefront from multiple sources and updating the control law during this propagation. A slightly different idea of wavefront propagation to compute the Voronoi tessellations was also used in [24] in the context of deployment on non-planar surfaces embedded in $3D$ space. Differently from our work, in that algorithm there was the necessity of a second step called *back propagation* to reassign vertices to neighbor robots and there was no control law being computed, only the tessellation.

The cost of our approach is the one of a single Dijkstra search $O(V_G \log V_G)$, where V_G is the number of graph vertices. This allows us to say that our algorithm is more efficient than those proposed in the already mentioned papers [17] and [18] which are also based on environment discretization. The complexity of the algorithms proposed in [17] and [18] was $O(V_G^3)$ and $O(nV_G^2)$, where n is the number of robots, respectively.

Paper Structure

In the next section we present the main aspects of the basic method in [2] derived from the *Locational Optimization Framework* [25], using a distance function that is independent of the Euclidean metric. Extensions to consider heterogeneous agents are presented in Section III. Non-convex environments are considered in Section IV. In Section V we add constraints which allows for dealing with finite-size robots. A novel efficient algorithm to compute the proposed control law is discussed in Section VI. Results related to the implementation of this algorithm are shown in Section VII. We provide concluding remarks in Section VIII.

II. Locational Optimization Framework

Let $\Omega \subset \mathbb{R}^N$ be a given representation of the environment, $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the configuration of n robots, where $\mathbf{p}_i \subset \Omega$, and $W = \{W_1, \ldots, W_n\}$ be a tessellation of Ω such that $I(W_i) \cap I(W_j) = \emptyset$, $\forall i \neq j$, where $I(\cdot)$ represents the

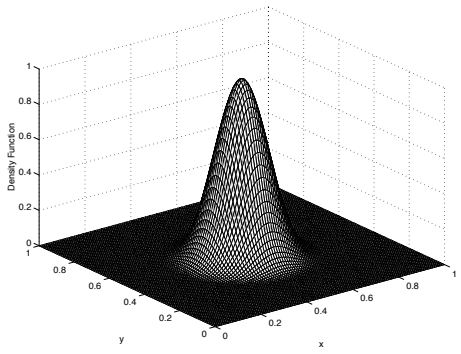


Fig. 1. Example of density function in a 2D environment. Its center is placed at the point $\mathbf{q}_c = [0.5, 0.5]^T$ and its support is equal to $\|\mathbf{q} - \mathbf{q}_c\| < 0.3$.

interior of a given region, and $\cup_{i=1}^{n} W_i = \Omega$. The key idea is that each agent i is responsible for covering a region W_i. By covering region W_i, we mean agent i will be responsible to perform required tasks inside W_i. We are interested in tasks in which the network performance is strongly related to the spatial placement of agents. Examples of such tasks are sensory coverage, intruders tracking, victims detection, etc. As a measure of the system performance we define the *deployment functional*:

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

where d corresponds to a function that measures distances between locations in Ω and agents. Note that we do not require that this function defines a metric in Ω. The function $\phi : \Omega \to \mathbb{R}_+$ is a distribution density function which defines a weight for each point in Ω. The density function may reflect a knowledge of the probability of occurrence of events in different regions, or simply a measure of relative importance of different regions in Ω. Therefore, points with greater weight values should be better covered by the networked robots than points with smaller values. Figure 1 shows an example of a density function in a $2-D$ environment for an application in which high coverage is required in the central area while no coverage is required at the boundary (distances greater than 0.3 from the center). The function $f : \mathbb{R} \to \mathbb{R}$ is a smooth strictly increasing function over the range of d.

The problem of deployment is then translated to the problem of minimizing the functional in (1). In fact, this functional appears in a wide class of applications such as data compression, placement of resources, and quadrature rules, as presented in [26]. In the following subsection the necessary conditions to minimize the functional in Equation (1), which will form the basis for our control law, are established.

A. Centroidal Voronoi Tessellation

An important tool in the Locational Optimization theory is the *Voronoi tessellation*. Given the set of points $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, often called *sites*, distributed over the bounded domain Ω, with boundary $\partial\Omega$, we define the Voronoi region, or Voronoi cell, V_i, associated to the point $\mathbf{p}_i$ according to a given distance function d as:

$$V_i = \{\mathbf{q} \in \Omega | d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \neq i\}. \tag{2}$$

The definition in (2) is in fact a generalization of the ordinary definition of Voronoi regions based on the Euclidean distance [25]. The generalized Voronoi tessellation of the set P, $V(P)$, is the collection of such regions. The Voronoi boundary ∂V_i is defined as:

$$\partial V_i = \cup_{j=1}^{n} l_{ij} \cup \{\partial\Omega \cap V_i\}. \tag{3}$$

where l_{ij} is the bisector:

$$l_{ij} = \{\mathbf{q} \in \Omega | d(\mathbf{q}, \mathbf{p}_i) = d(\mathbf{q}, \mathbf{p}_j), j \neq i\}. \tag{4}$$

Given a robot i we define the neighborhood of i, $\mathcal{N}_i$, as the set of robots that share Voronoi boundaries with V_i:

$$\mathcal{N}_i = \{j \in \mathcal{I} | \partial V_i \cap \partial V_j \neq \emptyset\}\,, \tag{5}$$

where $\mathcal{I} = \{1, \ldots, n\}$ is the robot index set.

When d is the Euclidean distance, the bisectors are hyperplanes and the Voronoi cells are convex, if Ω is convex. In this case, two neighbor agents i and j are associated with cells that share a hyperplane, and this hyperplane intersects the segment $\overline{\mathbf{p}_i\mathbf{p}_j}$ at its midpoint, and perpendicular to the segment. For an extensive treatment of Voronoi tessellations we refer to [25]. The next two propositions relate Voronoi tessellations with the minimization of the objective functional in (1) for a general distance function, d. The proofs follow the same arguments, used by [26] and [27] in the case of the Euclidean distance.

Proposition 1. *A necessary condition for a minimizer of the objective functional in (1) is that the W tessellation corresponds to the Voronoi tessellation, $V(P)$, according to the distance function d.*

Proof: Let $\hat{V}$ be another tessellation than the Voronoi V. For a given point $\mathbf{q} \in V_i$ and $\mathbf{q} \in \hat{V}_j$ we can write:

$$f(d(\mathbf{q}, V_i))\phi(\mathbf{q}) \leq f(d(\mathbf{q}, \hat{V}_j))\phi(\mathbf{q}). \tag{6}$$

Since $\hat{V}$ is not a Voronoi tessellation, the inequality in (6) will hold strictly over some measurable set of Ω. Therefore:

$$\mathcal{H}(P, V) < \mathcal{H}(P, \hat{V}).$$

■

Assuming that W is determined by the Voronoi tessellation of the points in P then $\mathcal{H}(P, W) = \mathcal{H}(P, V(P)) = \mathcal{H}(P)$ and we have the following result

Proposition 2. *A necessary condition for $\mathcal{H}(P)$ to be minimized is:*

$$\frac{\partial \mathcal{H}(P)}{\partial \mathbf{p}_i} = \frac{\partial \mathcal{H}(\mathbf{p}_i, V_i)}{\partial \mathbf{p}_i} = \int_{V_i} \frac{\partial}{\partial \mathbf{p}_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q} = 0. \tag{7}$$

Proof: By applying the differentiation under the integral sign (see [28]) we can write

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial x_i} =& \frac{\partial}{\partial x_i}\int_{V_i} f(d(\mathbf{q},\mathbf{p}_{i_0}))\phi(\mathbf{q})d\mathbf{q}|_{x_i=x_0} \\ &+ \int_{V_{i_0}} \frac{\partial}{\partial x_i} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q})d\mathbf{q} \\ &+ \sum_{j\in\mathcal{N}_i} \frac{\partial}{\partial x_i}\int_{V_j} f(d(\mathbf{q},\mathbf{p}_j))\phi(\mathbf{q})d\mathbf{q}|_{x_i=x_0} \\ =& \int_{\partial V_i} f(d(\mathbf{q},\mathbf{p}_{i_0}))\phi(\mathbf{q})\frac{\partial(\partial V_i)}{\partial x_i}\cdot\mathbf{n}_i ds \\ &+ \int_{V_{i_0}} \frac{\partial}{\partial x_i} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q})d\mathbf{q} \\ &+ \sum_{j\in\mathcal{N}_i}\int_{\partial V_j} f(d(\mathbf{q},\mathbf{p}_j))\phi(\mathbf{q})\frac{\partial(\partial V_j)}{\partial x_i}\cdot\mathbf{n}_j ds\,,\end{aligned}$$

where $\mathbf{p}_i = [x_i, y_i]^T$ in two dimensions, $\mathbf{p}_{i_0}$ is a fixed configuration of agent i, V_{i_0} is the Voronoi region associated to $\mathbf{p}_i = \mathbf{p}_{i_0}$, $\mathbf{n}_i$ and $\mathbf{n}_j$ are the outward facing unit normals of ∂V_i and ∂V_j respectively, and ds is the element of arc length. At the bisector we have $\partial V_i = \partial V_j = l_{ij}$ and $\mathbf{n}_j = -\mathbf{n}_i$. Moreover, since $\partial V_i = \{\cup_{j\in\mathcal{N}_i} l_{ij}\} \cup \{\partial\Omega \cap V_i\}$, and $\frac{\partial(\partial V_j)}{\partial x_i} = 0$ at $\partial\Omega \cap V_i$, we have

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial x_i} =& \sum_{j\in\mathcal{N}_i}\int_{l_{ij}} [\Delta f]\varphi(\mathbf{q})\frac{\partial(\partial V_i)}{\partial x_i}\cdot\mathbf{n}_i ds \\ &+ \int_{V_{i_0}} \frac{\partial}{\partial x_i} f(d(\mathbf{q},\mathbf{p}_i))\varphi(\mathbf{q})d\mathbf{q}\,,\end{aligned}$$

where $\Delta f = f(d(\mathbf{q},\mathbf{p}_{i_0})) - f(d(\mathbf{q},\mathbf{p}_j))$.

Due to the property in (4), $d(\mathbf{q},\mathbf{p}_{i_0}) = d(\mathbf{q},\mathbf{p}_j)$ at l_{ij}. Clearly, $\frac{\partial \mathcal{H}}{\partial y_i}$ can be obtained similarly to $\frac{\partial \mathcal{H}}{\partial x_i}$. Therefore, we conclude that

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \int_{V_i} \frac{\partial}{\partial \mathbf{p}_i} f(d(\mathbf{q},\mathbf{p}_i))\varphi(\mathbf{q})d\mathbf{q}\,, \tag{8}$$

which must be equal to zero at a minimum point. ∎

In [2] the Euclidean distance is used as d, and $f(x) = x^2$. Moreover, since it is assumed a convex environment it is easy to prove that all Voronoi cells are convex polytopes. In this case the necessary configuration to be at a minimum is obtained when each agent is located exactly at the *centroid* of its own Voronoi cell. The centroid is given by:

$$\mathbf{p}_i^* = \frac{\int_{V_i}\mathbf{q}\phi(\mathbf{q})d\mathbf{q}}{\int_{V_i}\phi(\mathbf{q})d\mathbf{q}}. \tag{9}$$

Similarly, we can define a *generalized centroid* for general f and d functions, as follows:

$$\mathbf{p}_i^* = \min_{\mathbf{p}_i\in(V_i\cup\partial V_i)} \int_{V_i} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q})d\mathbf{q}. \tag{10}$$

According to Propositions 1 and 2, every robot must be driven to the generalized centroid of its Voronoi region to minimize the functional (1). The resulting partition of the environment is commonly called *Centroidal Voronoi Tessellation* (CVT). In the next subsection we present a distributed control law that is used in [2] to converge to such configuration.

B. Continuous-Time Lloyd Algorithm

A classic discrete-time method to compute CVT's is the Lloyd's algorithm [1]. In each iteration this method executes three steps: (i) compute the Voronoi regions; (ii) compute the centroids; (iii) move each point site to the corresponding centroid.

In [2] a continuous-time version of this approach is proposed for kinematic models:

$$\dot{\mathbf{p}}_i = \mathbf{u}_i. \tag{11}$$

The following control law guarantees that the system converges to a CVT:

$$\mathbf{u}_i = -k(\mathbf{p}_i - \mathbf{p}_i^*)\,, \tag{12}$$

where k is a positive gain. The control law is a gradient-descent approach, since if d is the Euclidean distance and $f(x) = x^2$,

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = 2\left(\int_{V_i}\phi(\mathbf{q})d\mathbf{q}\right)(\mathbf{p}_i - \mathbf{p}_i^*). \tag{13}$$

It is important to mention that $\mathcal{H}$ is non-convex which implies that the system will in general converge to a CVT that corresponds to a local minimum. In the rest of the paper we present further developments of the method proposed in [2].

III. Heterogeneous Robots in Convex Environments

In this section we consider the problem of deploying a team of heterogeneous agents in an environment represented by a convex polytope. We capture this heterogeneity by associating a weight to each robot. The larger this weight, the better the robot according to a desired criterion. In a coverage task, for example, this weight may represent the fraction of area covered by the footprints of the robot sensors. In this case, larger weights will be assigned to robots equipped with sensors with larger footprints.

We will consider that the agent weight, R_i, will determine the radius of a circle (in $\mathbb{R}^2$) $B_i(\mathbf{p}_i, R_i)$, where $\mathbf{p}_i$ is the center position. Mathematically, we describe the team task as the minimization of the functional:

$$\mathcal{H}(P, PV) = \sum_{i=1}^{n}\int_{PV_i} [\|\mathbf{q}-\mathbf{p}_i\|^2 - R_i^2]\phi(\mathbf{q})d\mathbf{q}\,, \tag{14}$$

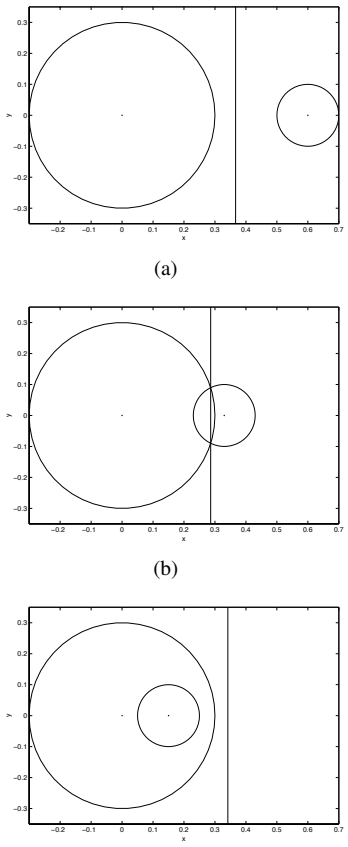
where $f(x) = x$ and $d(\mathbf{q},\mathbf{p}_i) = \|\mathbf{q}-\mathbf{p}_i\|^2 - R_i^2$ is the so-called *power distance* [29]. According to Proposition 1 the required tessellation must then be a Voronoi partition according to the power distance. The resulting tessellation is well-known in the literature and it is often called the *Voronoi Diagram in the Laguerre geometry* [30] or the *power diagram* [29].

The power diagram, PV, associates a power region, PV_i, with each circle $(\mathbf{p}_i, R_i)$ in $\mathbb{R}^2$ defined by:

$$PV_i = \{\mathbf{q}\in\mathbb{R}^2 | d(\mathbf{q},\mathbf{p}_i) \le d(\mathbf{q},\mathbf{p}_j), \forall j\neq i\}\,, \tag{15}$$

where the *power distance* $d(\mathbf{q},\mathbf{p}_i) = \|\mathbf{q}-\mathbf{p}_i\|^2 - R_i^2$.

The power distance is different from the Euclidean distance between $\mathbf{q}$ and the circle $B_i(\mathbf{p}_i, R_i)$, which is given by $d_{AW}(\mathbf{q},\mathbf{p}_i) = \|\mathbf{q}-\mathbf{p}_i\| - R_i$. The power distance, $d(\mathbf{q},\mathbf{p}_i)$, is



(a)

(b)

(c)

Fig. 2. Power cells. Fig. 2(a) Two disjoint circles. Fig. 2(b) Intersecting circles with both centers inside their corresponding power cells. Fig. 2(c) Intersecting circles with one center outside its corresponding power cell.

a monotonically-increasing function of the Euclidean distance, $d_{AW}(\mathbf{q}, \mathbf{p}_i)$. However, the power distance can be negative. When $\mathbf{q}$ lies inside B_i, the power distance is negative but it is still a monotonic function of the Euclidean distance between $\mathbf{q}$ and ∂B_i. When the power distance is positive, it is also the square of the length of the bi-tangent formed by $\mathbf{q}$ with B_i. We will use this power distance to compute the control laws for heterogeneous robots.

Alternatively one can use the distance $d_{AW}(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{q} - \mathbf{p}_i\| - R_i$ and compute distributed control laws based on the induced Voronoi diagram. The Voronoi diagram derived from the distance d_{AW} is called the *additively weighted Voronoi diagram* [25]. It has Voronoi regions whose boundaries are the union of straight line segments and hyperbolic arcs and it is harder to compute.

The power diagram can be viewed as a generalized Voronoi diagram which is closely related to the original Voronoi diagram. Some of its properties are:

Property 1. *The bisector between neighbor power cells PV_i and PV_j is a hyperplane perpendicular to the segment that connects the centers of the circles, B_i and B_j. If the two circles intersect, the bisector passes through the points of intersection. The bisector is defined by the equation:*

$$(\mathbf{p}_i - \mathbf{p}_j)^T \mathbf{q} = \frac{1}{2}(\|\mathbf{p}_i\|^2 - \|\mathbf{p}_j\|^2 - R_i^2 + R_j^2). \tag{16}$$

Property 2. *Each power cell PV_i is convex or empty. A necessary condition for PV_i to be empty is that the center $\mathbf{p}_i$ is contained in the union of the other circles.*

Property 3. *If a circle, B_i, is not intersected by any other circle, then B_i is entirely contained in PV_i.*

Property 4. *If all circles B_i are identical, $PV_i = V_i$.*

Figure 2 presents three different cases for the power regions for two circles. Figure 2(c) presents the situation where an agent is not located inside its own power region. Thus it is also possible that a robot could have an empty power region. In Figure 2(b), if we have a third robot the same size of the smallest robot inside the power region of the large robot, this third robot contained in the power region of the largest robot would have an empty power region.

The next proposition presents the gradient that will be used by the distributed control laws.

Proposition 3. *The gradient of $\mathcal{H}(P, PV)$ in (14) is given by:*

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = 2(\mathbf{p}_i - \mathbf{p}_i^*) \int_{PV_i} \phi(\mathbf{q}) d\mathbf{q}\,, \tag{17}$$

where $\mathbf{p}_i^*$ is the centroid of PV_i:

$$\mathbf{p}_i^* = \frac{\int_{PV_i} \mathbf{q}\phi(\mathbf{q}) d\mathbf{q}}{\int_{PV_i} \phi(\mathbf{q}) d\mathbf{q}}. \tag{18}$$

Proof: According to Proposition 2 we have

$$\begin{aligned}\frac{\partial \mathcal{H}(P, PV_i)}{\partial \mathbf{p}_i} &= \int_{PV_i} \frac{\partial [\|\mathbf{q} - \mathbf{p}_i\|^2 - R_i^2]}{\partial \mathbf{p}_i} \phi(\mathbf{q}) d\mathbf{q} \\ &= \int_{PV_i} -2(\mathbf{q} - \mathbf{p}_i)\phi(\mathbf{q}) d\mathbf{q} \\ &= 2(\mathbf{p}_i - \mathbf{p}_i^*) \int_{PV_i} \phi(\mathbf{q}) d\mathbf{q}.\end{aligned}$$

■

By observing the last proposition we conclude that we can use the same control law in (12). However, one must be careful of the special properties of power diagrams.

As stated in Property 2, a robot that is located in the union of other robots circles may have an empty power region. This is not surprising. Since we are designing strategies for heterogeneous teams of robots that act based only on local information, the best robots have priority during the deployment. Thus, the worst robots may get trapped in configurations in which they do not contribute to the overall mission. Of course, it is possible to let the "trapped" robot perturb the system by executing a deterministic controller to a region outside the circle in which it is trapped.

Now, we present a simulation to verify the proposed approach. We show a team of nine agents with three possible associated circles. Two agents have a large circle of radius 0.08 units, one agent has a medium-size circle of radius 0.05 units, and the remaining robots have the smallest circle of radius 0.02 units. In the left figures we present configurations of the group during the evolution of the simulation and in the right we present trajectories that show how the robots evolved from the configuration shown in the panel immediately

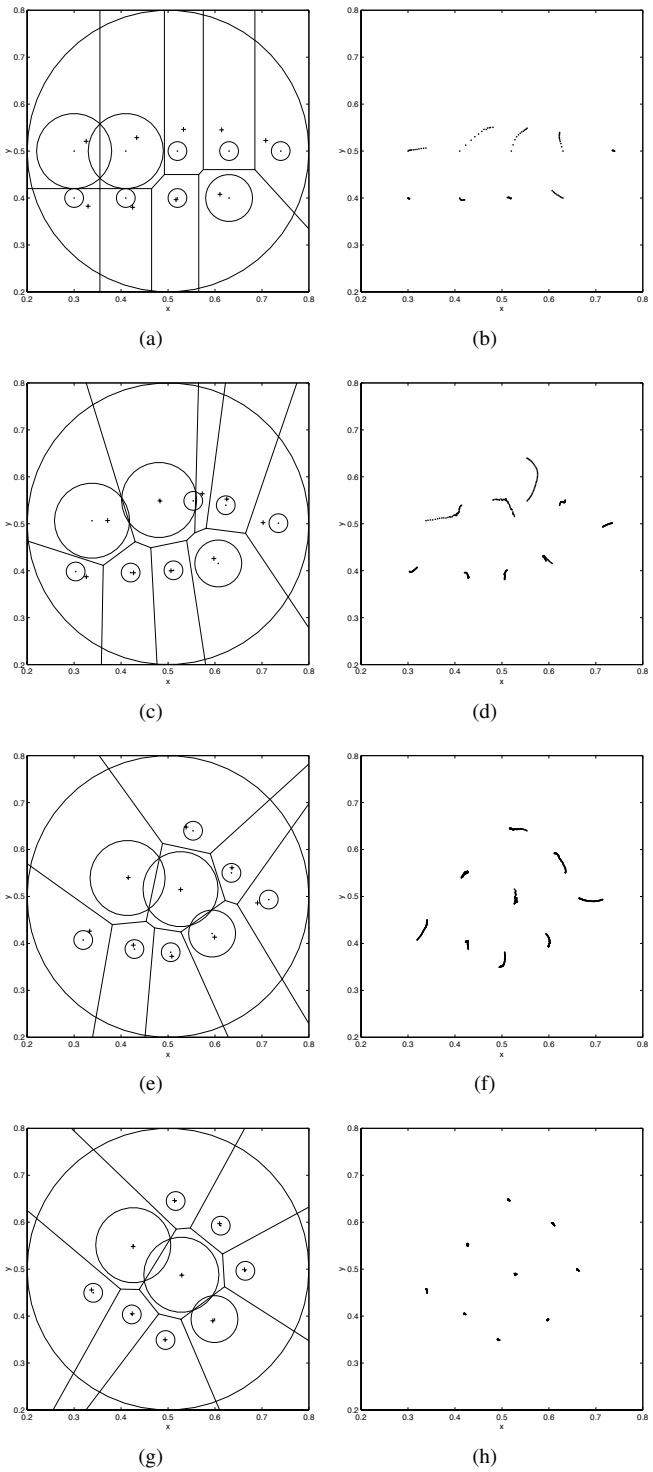


Fig. 3. Simulation results for heterogeneous agents in a environment with density function defined as in Fig. 1. In the panels on the left, the large circle with diameter 0.6 units represents the density function support. A team of nine robots, starting from an initial configuration (Fig. 3(a)), with intermediate configurations (Figs. 3(c) and 3(e)), converge to a centroidal formation (Fig. 3(g)). Figs. 3(b), 3(d), 3(f), and 3(h) correspond to the trajectories followed by the robots starting from the configuration in the figure presented in the left. The crosses represent the centroids of the corresponding power regions. Small circles associated with robots represent their capacity according to a given criterion.

to the left to the next configuration in the panel below. The trajectories in Figure 3(h) end at the optimal configuration. The corresponding power diagram is also presented. The density function is the same as in Figure 1 centered at the point $\mathbf{q}_c = [0.5, 0.5]^T$. This function is a smooth cubic spline determined by the equation:

$$\phi(\mathbf{q}-\mathbf{q}_c, h) = \begin{cases} 1 - \frac{3}{2}\kappa^2 + \frac{3}{4}\kappa^3 & \text{if} \quad 0 \leq \kappa \leq 1\,, \\ \frac{1}{4}(2-\kappa)^3 & \text{if} \quad 1 \leq \kappa \leq 2\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{19}$$

where $\kappa = \|\mathbf{q} - \mathbf{q}_c\|/h$. It can be observed that the function support is determined by $2h$.

We can verify that the final configuration is obtained when a centroidal tessellation is obtained, as expected. It is interesting to note that the best robots converged to the area where we find the higher values of the density function. It is also important to mention that we did not encounter a situation where a robot's Voronoi region was empty. The control law in (12) drives the robots to configurations in which they tend to converge to centroids that are located inside each convex cell.

IV. Non-convex Environments

We start this section by considering a homogeneous group of robots. Thus, we first consider the problem of minimizing the cost functional in a non-convex environment:

$$\mathcal{H}(P, W) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i, W_i) = \sum_{i=1}^{n} \int_{W_i} g(\mathbf{q}, \mathbf{p}_i)^2 \phi(\mathbf{q}) d\mathbf{q}\,, \tag{20}$$

where $g(\mathbf{q}, \mathbf{p}_i)$ is the geodesic distance between $\mathbf{q}$ and $\mathbf{p}_i$.

Let Ω be a compact region in $\mathbb{R}^2$ with boundary, $\partial\Omega$, determined by a simple polygon with m sides and set of vertices $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$. Also, assume that $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \Omega$. By the geodesic distance, $g(\mathbf{o}, \mathbf{w})$, between two points $\mathbf{o}$ and $\mathbf{w}$, we mean the length of the shortest path, $s(\mathbf{o}, \mathbf{w})$, between $\mathbf{o}$ and $\mathbf{w}$, entirely contained in Ω. In fact, it is well known that such a path is formed by the sequence of segments $\{\overline{\mathbf{o}\mathbf{v}_{r1}}, \overline{\mathbf{v}_{r1}\mathbf{v}_{r2}}, \ldots, \overline{\mathbf{v}_{rl-1}\mathbf{v}_{rl}}, \overline{\mathbf{v}_{rl}\mathbf{w}}\}$, where $\mathbf{v}_{ri}$'s are reflex vertices of $\partial\Omega$. By reflex vertices, we mean the vertices with internal angle greater than 180 degrees. According to Proposition 1 we require a Voronoi tessellation computed according to the geodesic metric. Such a geodesic Voronoi tessellation can be computed by means of the algorithms proposed in [21] and [31]. In [21], interesting properties of the geodesic distance in simple polygons are presented. Some of these properties are listed bellow:

Property 5. *As Ω is closed and bounded by a simple polygon, $s(\mathbf{o}, \mathbf{w})$ exists and is unique for every pair of points in Ω. Moreover, $s(\mathbf{o}, \mathbf{w})$ is piecewise linear with "breakpoints" at reflex vertices of $\partial\Omega$.*

Property 6. *The function $g(\mathbf{o}, \mathbf{w})$ is continuous in both $\mathbf{o}$ and also $\mathbf{w}$. Furthermore, this function is continuously differentiable except at a set of measure zero Γ.*

The following proposition allows for devising distributed control laws.

Proposition 4. *If $\mathbf{w} \in \Omega\backslash\Gamma$ is a point where the gradient*

$\nabla_{\mathbf{w}} g(\mathbf{w},\mathbf{q})$ *exists, then this gradient is given by:*

$$\frac{\partial g}{\partial \mathbf{w}} = -\mathbf{z}_{\mathbf{w},\mathbf{q}}\,, \tag{21}$$

where $\mathbf{z}_{\mathbf{w},\mathbf{q}}$ *is a unit vector directed along the first segment of the shortest path* $s(\mathbf{w},\mathbf{q})$.

Proof: If the shortest path, $s(\mathbf{w},\mathbf{q})$, is given by the segment $\overline{\mathbf{w}\mathbf{q}}$ then $g(\mathbf{w},\mathbf{q}) = \|\mathbf{q}-\mathbf{w}\|$. Thus,

$$\frac{\partial g}{\partial \mathbf{w}} = -\frac{(\mathbf{q}-\mathbf{w})}{\|\mathbf{q}-\mathbf{w}\|}. \tag{22}$$

Now, let the shortest path, $s(\mathbf{w},\mathbf{q})$, between the points $\mathbf{w}$ and $\mathbf{q}$, be given by the sequence of segments $\{\overline{\mathbf{w}\mathbf{v}_{r1}}, \overline{\mathbf{v}_{r1}\mathbf{v}_{r2}}, \ldots, \overline{\mathbf{v}_{rl-1}\mathbf{v}_{rl}}, \overline{\mathbf{v}_{rl}\mathbf{q}}\}$. The path length is then determined by:

$$\begin{aligned} g(\mathbf{w},\mathbf{q}) =& \|\mathbf{v}_{r1}-\mathbf{w}\| + \|\mathbf{v}_{r2}-\mathbf{v}_{r1}\| + \ldots + \|\mathbf{v}_{rl}-\mathbf{v}_{rl-1}\| \\ &+ \|\mathbf{q}-\mathbf{v}_{rl}\|\,, \end{aligned}$$

where $\mathbf{v}_{ri}$'s are reflex vertices of the polygon $\partial\Omega$. Therefore,

$$\frac{\partial g}{\partial \mathbf{w}} = -\frac{(\mathbf{v}_{r1}-\mathbf{w})}{\|\mathbf{v}_{r1}-\mathbf{w}\|}. \tag{23}$$

■

From this proposition and from (8), a distributed control law that minimizes $\mathcal{H}$ in (20) can be computed by:

$$\mathbf{u}_i = -k\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = 2k\int_{V_i} g(\mathbf{q},\mathbf{p}_i)\phi(\mathbf{q})\mathbf{z}_{\mathbf{p}_i,\mathbf{q}}d\mathbf{q}. \tag{24}$$

It is important to mention that at the points where we have discontinuities in the gradient, which are also reflex vertices of the polygon $\partial\Omega$, we must use generalized gradients [32].

Definition 1. *(Clarke's Generalized Gradient): For a locally Lipschitz function* $h : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}$ *define the generalized gradient of* h *at* $(\mathbf{x},t)$ *by*

$$\partial h(\mathbf{x},t) = \overline{co}\{\lim \nabla h(\mathbf{x}_i,t_i) | (\mathbf{x}_i,t_i) \to (\mathbf{x},t)\}\,, \tag{25}$$

where $(\mathbf{x}_i,t_i) \notin \Gamma$ *which is the set of measure zero where the gradient is not defined.*

Assume that agent $\mathbf{p}_i$ is located exactly at a reflex vertex $\mathbf{v}_{ri}$. Let $\partial_F \mathcal{H}(\mathbf{p}_i,V_i) \subset \partial\mathcal{H}(\mathbf{p}_i,V_i)$ be the subset of generalized gradients, ξ, at the reflex vertex, $\mathbf{v}_{ri}$, that have feasible directions, *i. e.*, $\exists \delta \in (0,\delta_{max}]$ such that $\mathbf{v}_{ri} - \delta\xi \in \Omega$. We can use the following control law at the reflex vertices:

$$\mathbf{u}_i = -k\xi\,, \tag{26}$$

where k is a positive gain, and ξ could be any vector belonging to $\partial_F\mathcal{H}(\mathbf{p}_i,V_i)$.

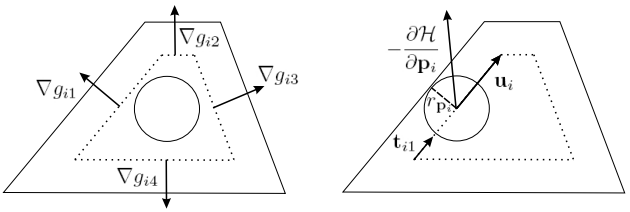
To deploy heterogeneous teams in non-convex environments we define the geodesic power distance:

$$d(\mathbf{a},\mathbf{b}) = g(\mathbf{a},\mathbf{b})^2 - R_i^2. \tag{27}$$

Therefore, we define the new cost function:

$$\mathcal{H}(P,GPV) = \sum_{i=1}^{n}\int_{GPV_i}[g(\mathbf{q},\mathbf{p}_i)^2 - R_i^2]\phi(\mathbf{q})d\mathbf{q}\,, \tag{28}$$

where GPV is the geodesic power tessellation, *i. e.*, the Voronoi tessellation induced by (27). The robots control law is the same as (24), with GPV_i instead of V_i.



(a) (b)

Fig. 4. Linear constraints for a disk-robot with radius $r_{\mathbf{p}_i}$. The dotted lines represent the facets of $\partial\mathcal{F}_{V_i}$, which are given by $g_{i1}=0$, $g_{i2}=0$, $g_{i3}=0$, and $g_{i4}=0$, with gradients as in Fig. 4(a). Given the active set $\mathcal{A}=\{g_{i1}\}$ in Fig. 4(b) we compute the control input $\mathbf{u}_i$ by projecting the negative gradient of $\mathcal{H}$ onto the unit vector $\mathbf{t}_{i1}$ which is tangent to the active facet.

V. Robots with Finite Size

A practical problem of the unconstrained minimization executed by the pure gradient-descent law in (12) is that actual robots are not point-robots. In this section, we extend the basic results to robots that can be modelled as circular disks, each one with radius $r_{\mathbf{p}_i} = r_{\mathbf{p}_j}$, $\forall i,j$. Also, as in [2], we first assume that the robotic network is homogeneous, d is the Euclidean distance, $f(x) = x^2$, and the environment is a convex polytope. Since the Voronoi regions, in the case of the Euclidean distance, are convex we can guarantee that each centroid lies in the interior of the associated region. Therefore, point agents will never invade a neighbor cell and collision avoidance is guaranteed. On the other hand, if the robots have finite size this basic safety property cannot be guaranteed.

Let $\mathcal{F}_{V_i}$ be the *free Voronoi region* defined by the set of points:

$$\mathcal{F}_{V_i} = \{\mathbf{q} \in V_i | \|\mathbf{q}-\mathbf{q}_{\partial V_i}\| \geq r_{\mathbf{p}_i}, \forall \mathbf{q}_{\partial V_i}\}\,, \tag{29}$$

where $\|\cdot\|$ is the Euclidean norm and $\mathbf{q}_{\partial V_i}$ is a point at the boundary of the Voronoi region, ∂V_i. In fact, the boundaries of the free Voronoi regions, $\partial\mathcal{F}_{V_i}$, are hyperplanes parallel to the hyperplanes that define the boundaries of V_i, located at a distance $r_{\mathbf{p}_i}$ from ∂V_i. It is straight forward to check that such free region is convex.

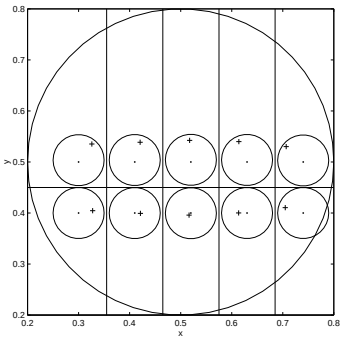
If the robots start from a safe configuration, a sufficient condition to guarantee collision avoidance is that each robot disk lies in the interior of its own Voronoi region. Therefore, we define a *constrained, location optimization* problem as follows. If robot i is constrained to remain inside its own Voronoi region which is bounded by m facets, there are m linear constraints on the position of its center. Accordingly, we find the robot positions:

$$\begin{aligned} &\min_{\mathbf{p}_i} \mathcal{H}(\mathcal{P},V) \\ &\text{s.t.} \\ &g_{i1}(\mathbf{p}_i) \leq 0\,, \ldots, g_{im}(\mathbf{p}_i) \leq 0 \end{aligned} \tag{30}$$

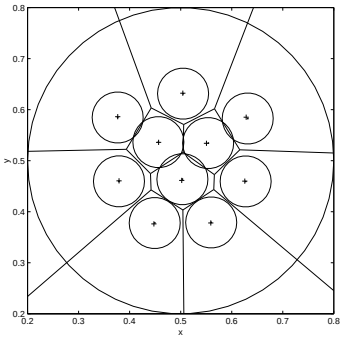
where $g_{il}(\mathbf{p}_i) = 0$ defines the lth facet of $\partial\mathcal{F}_{V_i}$.

Accordingly we choose the control law given by:

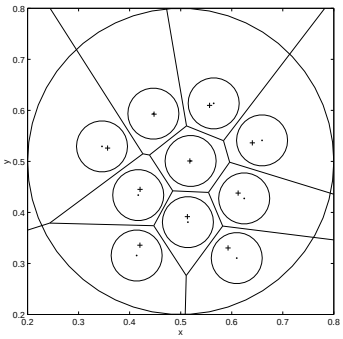
$$\mathbf{u}_i = \beta_i \mathbf{h}_i\,, \tag{31}$$



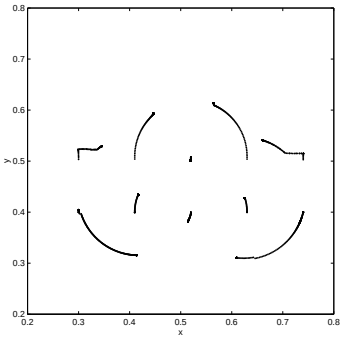
(a)



(b)



(c)



(d)

Fig. 5. Simulation results for finite size agents in an environment with density function defined as in Fig. 1. A team of ten finite size agents start from an initial configuration (Fig. 5(a)) and have final configuration that implies collisions as in Fig. 5(b) when executing the unconstrained minimization as in [2]. By using the proposed technique the agents reach a final configuration (Fig. 5(c)) without colliding with trajectories as in Fig. 5(d). The crosses represent the centroids of the corresponding Voronoi regions. The largest circle represents the density function support while the other circles represent the shape of the robots.

where $\beta_i \propto \left\| \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} \right\|$ and the vector $\mathbf{h}_i$ is determined by:

$$\begin{aligned} &\min_{\mathbf{h}_i} \left(\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} \right)^T \mathbf{h}_i \\ &\text{s.t.} \\ &\|\mathbf{h}_i\| = 1 \, , \nabla g_{ip}^T \mathbf{h}_i \leq 0 \, , \ldots \, , \nabla g_{it}^T \mathbf{h}_i \leq 0 \end{aligned} \tag{32}$$

If $\mathcal{A} = \{g_{ip}, \ldots, g_{it}\}$ is the set of active constraints $g_{ip} = \ldots = g_{it} = 0$, we can compute the controls in (31) as follows:

1) If $\mathcal{A}$ is empty then

$$\mathbf{u}_i = -k \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} \, , \tag{33}$$

2) Otherwise

$$\mathbf{u}_i = k\pi \left(-\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}, \partial \mathcal{F}_{V_i} \right) \, , \tag{34}$$

where $\pi \left(-\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}, \partial \mathcal{F}_{V_i} \right)$ gives the projection of vector $-\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}$ along the vector $\mathbf{t}_{il}$ which is the unit vector tangent to the lth facet (see Fig. 4). This tangent vector provides a feasible direction and is such that $-\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}^T \mathbf{t}_{il} > 0$.

Since the region $\mathcal{F}_{V_i}$ is convex and $\partial \mathcal{F}_{V_i}$ is given by plane faces the projection π is guaranteed to return a feasible vector if one exists. Equilibrium is obtained when the first-order Karush-Kuhn-Tucker (KKT) conditions [33], [34] are satisfied.

Clearly the control law designed to solve the problem in (30) will not guarantee the convergence of robots to the centroids, $\mathbf{p}_i^*$. However, they converge to the *constrained centroids*, $\mathbf{p}_i^c$, which are determined by the projection of $\mathbf{p}_i^*$ on the boundary $\partial \mathcal{F}_{V_i}$. This solution is locally optimal in the sense that this is the best solution which also guarantees that the entire robot disk (and not just the center point) is confined to its own dominance region.

Figure 5 shows the result obtained when solving the constrained minimization for a team of ten disk-shaped robots. The density function is the same as in Figure 1. In this example, collisions are observed when the unconstrained control law (12) is used (see Figure 5(b)). However, the trajectories shown in Figure 5(d) which are obtained by using the control law (31) are free of collisions. Note that in this case not all agents converge to their centroids (see Figure 5(c)). This is the price the robots have to pay to guarantee safety (no collisions).

We can use the strategy presented in this section with heterogeneous robotic networks in non-convex environments by projecting the computed control law onto the facets of the ordinary Voronoi diagram, V, when necessary.

VI. Algorithm Design

In this section we present a novel practical algorithm which allows for computing the proposed control law in real scenario. The proposed algorithm is based on the discretization of the environment. In fact we will consider uniform square tiling of the 2-dimensional configuration space. From this tiling we will generate a graph and perform efficient graph algorithms to compute the control law in Section IV for heterogeneous teams in non-convex environments. It is important to mention

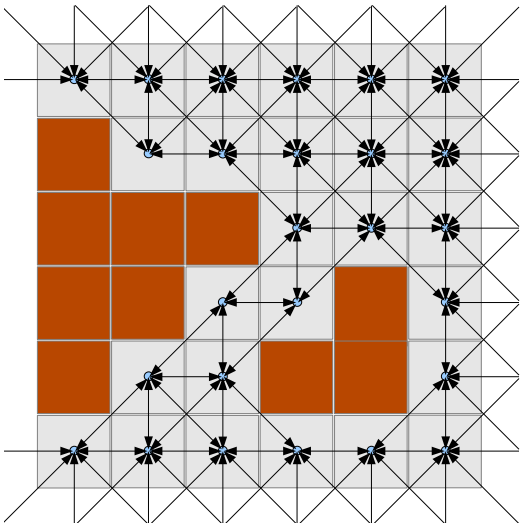


Fig. 6. A graph created by uniform square discretization of an environment. The brown cells represent obstacles.

that our algorithm is not restricted to environments that can be represented as simple polygons. We will be also able to deal with holes.

The graph $G = \{\mathcal{V}(G), \mathcal{E}(G), \mathcal{C}_G\}$ is constructed from the uniform square tiling of the configuration space. A vertex of the graph is placed in every accessible discretized cell, and edges are established along neighboring vertices. The cost of each edge is computed from the metric induced by the configuration space (typically the Euclidean length of the edges). Figure 6 illustrates construction of such a graph by uniform square discretization of a 2-dimensional configuration space.

Thus a graph consists of 3 components: A vertex set $\mathcal{V}(G)$, an edge set $\mathcal{E}(G) \subseteq \mathcal{V}(G) \times \mathcal{V}(G)$, and a cost function $\mathcal{C}_G : \mathcal{E}(G) \to \mathbb{R}^+$. An element in $\mathcal{E}(G)$ is represented by the ordered pair $[a, b] \in \mathcal{E}(G)$, which implies that there exists an edge in G connecting $a \in \mathcal{V}(G)$ to $b \in \mathcal{V}(G)$.

A. Algorithm for Tessellation and Control Computation

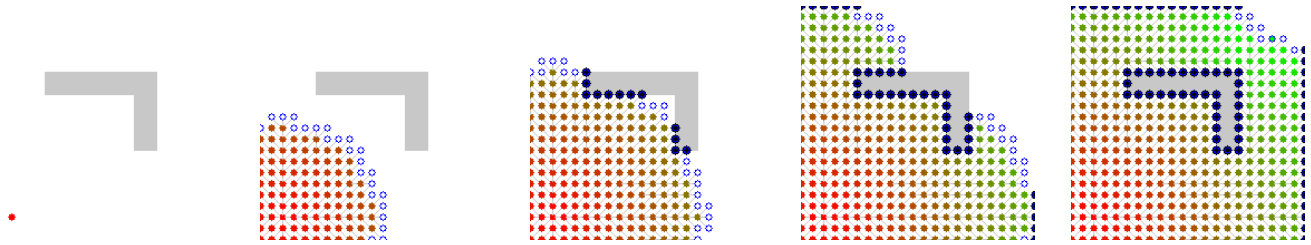
Our proposed algorithm for computing the tessellations and robot control commands in every time-step is, in essence, very similar to the Dijkstra's algorithm [22]. The intuition behind the Dijkstra's Algorithm is that starting from the start vertex p in the graph, a wavefront of *almost* equal geodesic distances is propagated through the graph. Figure 7 illustrates the progress of Dijkstra's algorithm. Notice the wavefront marked by the empty blue circles.

The idea behind the basic modification we make in the Dijkstra's algorithm for creating Voronoi tessellations is to have multiple start vertices from which we start propagation of the wavefront. Thus the wavefronts emanate from multiple sources. The places where the wavefronts collide will hence represent the boundaries of the Voronoi tessellations. In addition, we can conveniently alter the distance function, the level-set of which represents the boundaries of the Voronoi tesellation. This enables us to create geodesic Power Voronoi tessellation (see Section IV).

In order to compute the control command for the robots (*i.e.* the action of the robot in the next time step), we use the formula in Equation (24), with GPV_i instead of V_i. In a general metric setup, the vector $\mathbf{z}_{\mathbf{p}_i,\mathbf{q}}$ is the unit vector along the tangent at $\mathbf{p}_i$ to the geodesic joining $\mathbf{p}_i$ and $\mathbf{q}$. This fact can easily be proven by observing that the constant-value manifolds for the function $d(\mathbf{w}, \mathbf{q})$, as we keep $\mathbf{q}$ fixed and vary $\mathbf{w}$, are orthogonal to the geodesic joining $\mathbf{q}$ and $\mathbf{w}$ at $\mathbf{w}$.

Thus, in a discretized setup, the unit vectors $\mathbf{z}_{\mathbf{p}_i,\mathbf{q}}$ reduce to unit vectors along edges of the form $[p_i, p_i']$ for some $p_i' \in \mathcal{N}_G(p_i)$ (where, $\mathcal{N}_G(u) = \{w' \in \mathcal{V}(G) \mid [u, w'] \in \mathcal{E}(G)\}$ is the set of neighbors of u). Due to the way we expand the vertices in the algorithm, for a given q, we know the index of the robot, $\tau(q)$, whose tessellation it belongs to. Thus we can compute the geodesic joining the vertices $p_{\tau(q)}$ and q. The neighbor of $p_{\tau(q)}$ through which the geodesic passes is the desired $p'_{\tau(q)}$, and it is maintained in the variable $\eta(q)$ in an efficient way as described in the pseudo-code below. We can also compute the integration of (24) on the fly as we compute the tessellations.

The complete pseudo-code for computing tessellations and control commands is as follows,



(a) $iter = 0$. Start vertex in red. (b) $iter = 96$. The wave front (empty blue circles) is visible. (c) $iter = 190$. (d) $iter = 286$. (e) $iter = 376$.

Fig. 7. Illustration of progress of Dijkstra's algorithm.

$\{\tau, \{p_i'\}\}$ = **Tessellation_and_Control_Computation** $(G, \{p_i\}, \{R_i\}, \overline{\phi})$

Inputs: a. Graph G
b. Agent locations $p_i \in \mathcal{V}(G),\ i = 1, 2, \cdots, N$
c. Agent weight $R_i \in \mathbb{R}^+,\ i = 1, 2, \cdots, N$
d. Discretized weight/density function $\overline{\phi} : \mathcal{V}(G) \to \mathbb{R}$

Outputs: a. The tessellation map $\tau : \mathcal{V}(G) \to \{1, 2, \cdots, N\}$
b. The next position of each robot, $p_i' \in \mathcal{N}_G(p_i),\ i = 1, 2, \cdots, N$

```
1  Initiate g: Set g(v) := ∞, for all v ∈ V(G)  // Geodesic distances
2  Initiate ρ: Set ρ(v) := ∞, ∀v ∈ V(G)  // Power dists.
3  Initiate τ: Set τ(v) := −1, ∀v ∈ V(G)  // Tessellation
4  Initiate η: Set η(v) := ∅, ∀v ∈ V(G)
       // Pointer to robot neighbor. η : V(G) → V(G)
5  for each (i ∈ {1, 2, ··· , N})
6      Set g(p_i) = 0
7      Set ρ(p_i) = −R_i^2
8      Set τ(p_i) = i
9      Set I_i := 0   // The control integral. I_i, 0 ∈ C
10     for each (q ∈ N_G(p_i))  // For each neighbor of p_i
11         Set η(q) = q
12 Set Q := V(G)   // Set of un-expanded vertices
13 while (Q ≠ ∅)
14     q := arg min_{q'∈Q} ρ(q')  // Maintained by a heap
                                  data-structure.
15     if (g(q) == ∞)
16         break
17     Set Q = Q − q   // Remove q from Q
18     Set j := τ(q)
19     Set s := η(q)
20     if (s != ∅)   // Equivalently, q ∉ {p_i}
21         Set I_j += φ̄(q) × g(q) × (P(s) − P(p_j))
22     for each (w ∈ N_G(q))   // For each neighbor of q
23         Set g' := g(q) + C_G([q, w])
24         Set ρ' := PowerDist(P(g'), R_j)
25         if (ρ' < ρ(w))
26             Set g(w) = g'
27             Set ρ(w) = ρ'
28             Set τ(w) = j
29             if (s != ∅)   // Equivalently, q ∉ {p_i}
30                 Set η(w) = s
31 for each (i ∈ {1, 2, ··· , N})
32     Set p_i' := arg max_{u∈N_G(p_i)} (P(u) − P(p_i)) · I_i
           // Choose action best aligned along I_i.
33 return {τ, {p_i'}}
```

where, $\mathscr{C}$ is the configuration space (a normed vector space) of each robot (assumed to be the same for all robots). Typically, $\mathscr{C} \subseteq \mathbb{R}^2$. Also, the function $\mathbf{P} : \mathcal{V}(G) \to \mathscr{C}$ is such that $\mathbf{P}(q)$ gives the coordinate of the vertex at q. Thus, $\mathbf{P}(p_i) = \mathbf{p}_i$ and $\mathbf{P}(q) = \mathbf{q}$ in relation to Equation (24). In an uniform discretization setting we take $\overline{\phi}(q) = \kappa\ \phi(\mathbf{P}(q))$ for an arbitrary constant κ. $\mathbf{PowerDist}(x, R) = x^2 - R^2$ is the *power distance*.

B. Overall Algorithm: Adapted Lloyd's Algorithm

The overall algorithm consists of iterating over "**Tessellation_and_Control_Computation**" and updating the positions of the robots at each iteration.

$\{\tau^f, \{p_i\}^f\}$ = **Adapted_Lloyds** $(G, \{p_i\}, \{R_i\}, \overline{\phi})$

Inputs: a. Graph G
b. Initial agent locations $p_i \in \mathcal{V}(G),\ _{i=1,2,\cdots,N}$
c. Agent weight $R_i \in \mathbb{R}^+,\ _{i=1,2,\cdots,N}$
d. Discretized density function $\overline{\phi} : \mathcal{V}(G) \to \mathbb{R}$

Outputs: a. Final tessellation map $\tau : \mathcal{V}(G) \to \{_{1,2,\cdots,N}\}$
b. Robot final position, $p_i' \in \mathcal{V}(G),\ _{i=1,2,\cdots,N}$

```
1  Initiate H^0 := {P(p_1), P(p_2), ··· , P(p_N)}
       // Initiate history of robot positions
2  Set t := 0
3  while (t < m  OR  er(H^{(t−m):t}) > ε) // not converged
4      Set {τ, {p_i'}} :=
           Tessellation_and_Control_Computation
                                   (G, {p_i}, {R_i}, φ̄)
5      for each (i ∈ {1, 2, ··· , N})
6          Move i^th robot from P(p_i) to P(p_i')
7          Set p_i = p_i'   // Update robot positions
8      Set t += 1
9      Set H^t := {P(p_1), P(p_2), ··· , P(p_N)}
           // Append current position to history
10 return {τ, {p_i}}  // Latest tessellation & positions
```

where,

$$\mathbf{er}(H^{a:b}) = \frac{1}{1+b-a} \max_{i \in \{1,2,\cdots,N\}} \left(\sum_{h=a}^{b} \left| H_i^h - \tfrac{1}{1+b-a} \sum_{l=a}^{b} H_i^l \right|^2 \right)$$

is a measure of the variation in the position of the sensors over the most recent $1+b-a$ time steps. Convergence is hence detected by checking if the variation in the positions over the most recent m time-steps is less than a desired threshold, ϵ.

C. Distributed Implementation

In a distributed implementation, each robot is assumed to have a copy of the graph G, discretized density $\overline{\phi}$, and have knowledge about the weight and location of

neighbor robots. This information may be available assuming communication among neighbor robots. By neighbors we mean robots that share boundaries of the computed tessellation. Hence each robot is able to run locally the "**Tessellation_and_Control_Computation**" algorithm on its own local processor using only local information. Thus, it might be interesting to analyze the computational complexity of this algorithm.

In the distributed version of the "**Adapted_Lloyds**" algorithm, each robot is interested in computing its own control command and issuing it. The control commands for neighbor robots are not computed on the local processor or are discarded. After every time step the robots broadcast their new positions (from a ground truth or any other localization method) and receive updated positions of neighbors. This communication closes the control loop and accounts for errors in the execution of computed controls.

*D. Complexity of the **Tessellation_and_Control_Computation** Algorithm*

In Dijkstra's algorithm (as well as in our **Tessellation_and_Control_Computation** algorithm), one starts with all the vertices unexpanded (the initial content of the set Q), and in every iteration pops out a vertex (q with the lowest value of ρ). Thus the contents of the main loop (lines $14-30$) runs $O(V_G)$ times, where V_G is the number of vertices in the graph.

In every iteration the neighbors of the vertex that is being expanded are checked and their properties are updated. Since, by the process of construction of our graph, the graph has a constant *degree* (*i.e.* the number of edges emanating from a vertex), the contents of the inner loop (lines $23-30$) runs constant number of times in every iteration. However, for increased efficiency, in solving the 'argmin' problem of line 14 we maintain a heap data structure containing only the vertices of Q with finite value of ρ (call that set $Q_{<\infty}$). This however requires that we re-order the heap every time we update the value of ρ for a vertex. Using a binary heap the update operation is of $O(\log(|Q_{<\infty}|))$ complexity. It is difficult to estimate $|Q_{<\infty}|$ and how it will change with iterations since it largely depends on the environment, placement of obstacles and position of the robots in it. Typically $|Q_{<\infty}|$ will start at 0, go up to a maximum, and again fall back to 0. Considering the wost-case scenario, $|Q_{<\infty}| \cong V_G$. Thus in the worst case, the complexity of re-ordering the heap is $O(\log(V_G))$.

Thus, the complexity of the total algorithm, with the assumption of constant degree for the vertices in the graph, is estimated to be $O(V_G \log(V_G))$.

VII. Results

In this section we instantiate our results in the solution of sensory coverage tasks. We consider a heterogeneous team of robots equipped with sensors. We model coverage using circular footprints. Robot i has a footprint of radius R_i. Robots with larger footprints cover a larger fraction of the area, and they are implicitly assigned larger weights in the algorithm.

We implemented the proposed algorithm in C++ programming language. The 2-dimensional configuration space of the robots was uniformly discretized into square cells and the graph G was created out of it by placing a vertex in each free cell. Nodes were not placed inside obstacles.

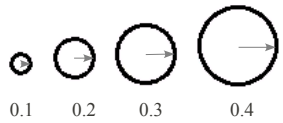


Fig. 10. Agents' weights in simulation of Fig. 9.

A. L-shaped Environment

The first set of results demonstrate the coverage of the L-shaped environment described in Figure **??** by 4 homogeneous robots. The environment is discretized into 240×240 cells. The density function is the same as the one described in (19), with $\mathbf{q}_c = [0.7, 0.7]^T$ and the value of ϕ is normalized between 0.1 and 2.0. Figure 8 shows snapshots of the robot configurations at different iterations. The intensity of green represents the density function. The curves in red are the boundaries of the tessellations, and the white circles represent the weights of the agents. The robots start near the bottom of the environment. Note how in the final configuration the distribution of the robots is biased toward the center of the density function, $\mathbf{q}_c$. The **Adapted_Lloyds** algorithm along with the plotting procedures run at a rate of 17 Hz on a single processor (2.1 GHz, 3Gb RAM) machine.

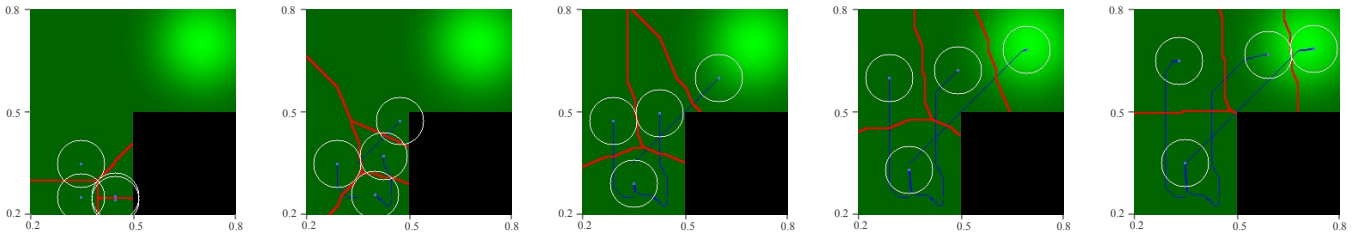
B. In a Complex Indoor Environment

Next we test the algorithm on a more complex office-like indoor environment of dimension 2.84×3.33 units (284×333 discretized) and the origin at the center of the environment. There are 4 heterogeneous robots. The agents' weights are chosen as $0.1, 0.2, 0.3$ and 0.4 units as illustrated in Figure 10. Figure 9 demonstrates the results. The density is chosen to be the Gaussian function $\phi(\mathbf{q}) = 0.135\ e^{2.0|\mathbf{q}-\mathbf{q}_c|^2}$, with $\mathbf{q}_c = [0.5, 1.3]^T$.

Figure 11(a) shows the final converged solution in the same environment, but with an uniform density function. Figure 11(b) shows how the value of $\mathcal{H}(P, PV)$ changes with time of execution (running on a single-processor, 2.1 GHz, 3Gb RAM machine, and including plotting operations). The algorithm reaches convergence in about 285 iterations. Thus the main loop of the "**Adapted_Lloyds**" algorithm runs at a rate of about 10 Hz for this environment.

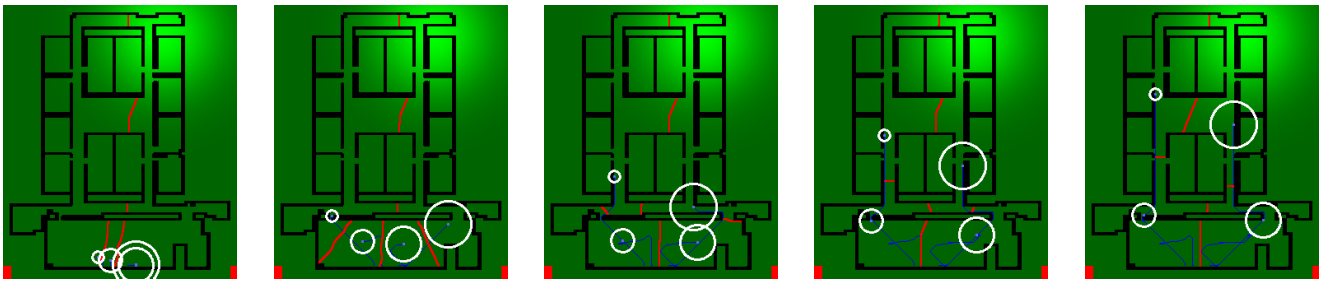
VIII. Conclusions

We addressed the problem of deriving optimal distributed control laws to deploy heterogeneous robotic networks in realistic environments. This paper incorporates three extensions into the previous work [1], [2] to address: (a) robots with different capacity according to a pre-specified criterion, (b) non-convex environments, and (c) disk-shaped robots. These extensions were first presented in our conference paper [20]. In the present work we give further details of the involved techniques and also show how they can be used together.

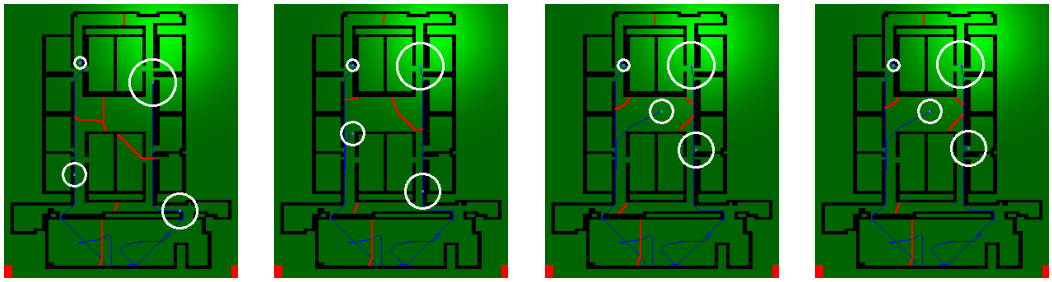


(a) $t = 0$. (b) $t = 50$. (c) $t = 100$. (d) $t = 150$. (e) $t = 200$ (*Converged*).

Fig. 8. Adapted Lloyd's algorithm in a L-shaped environment. The environment is 0.6×0.6 units in size, discretized into 240×240 cells, and have an obstacle occupying the lower right quadrant. Intensity of green represent magnitude of the density function. The red lines are the boundaries of the tessellations. The robots start off from the lower left corner of the environment, follow the **Adapted_Lloyds** algorithm, and finally attain optimal deployment.



(a) $t = 10$. (b) $t = 60$. (c) $t = 110$. (d) $t = 160$. (e) $t = 210$.



(f) $t = 260$. (g) $t = 310$. (h) $t = 360$. (i) $t = 410$ (*Converged*).

Fig. 9. Adapted Lloyd's algorithm in a real indoor environment. The environment is 2.84×3.33 units in size, discretized into 284×333 cells, so that each cell is 0.01×0.01 units in size. The weights of the robots in the task are $0.1, 0.2, 0.3$ and 0.4 units. The robots start off in the big room in the bottom of the environment, and follow the **Adapted_Lloyds** algorithm to attain optimal deployment. Note that the final distribution of the robots is biased towards regions of high density function.

The extensions are based on the use of different distance functions, power distance and geodesic distance, and the incorporation of constraints to allow collision avoidance. We should also mention that other distance functions such as the actual distance between points and circles can also be applied.

We also presented for the first time a distributed algorithm which allows for efficient computation of a discrete approximation of the proposed control law. This algorithm is based on wavefront propagation in the same spirit of the Dijkstra Algorithm, assuming a discretized environment. The running time complexity of the proposed algorithm is given by $O(V_G \log(V_G))$, where V_G is the number of discretization cells. This algorithm is more efficient than those previously proposed in the context of discrete environments.

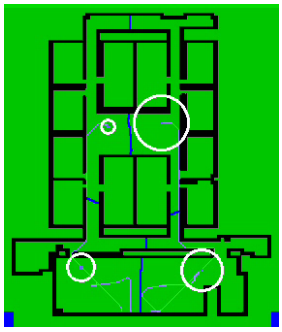
The main limitation of the proposed strategy stems from the fact that neighbor robots are the ones that share Voronoi boundaries in the tessellation, independently of distance. If the necessary information between neighbors is exchanged by direct communication between robots, performance might degrade if such robots are too far one from another. This problem is out of scope of this paper and it is left as future research direction. A possible direction to address this issue is the incorporation of connectivity maintenance between neighbor agents in the control law.

Extensions to spheres (in $\mathbb{R}^3$) are straight forward for the techniques proposed in Sections III and V to deal with heterogeneous robots and finite size, respectively. We can also apply the approach in Section IV to deal with non-convex environments in three dimensions by using the algorithm in Section VI to deal with $3 - D$ geodesic Voronoi diagrams efficiently.

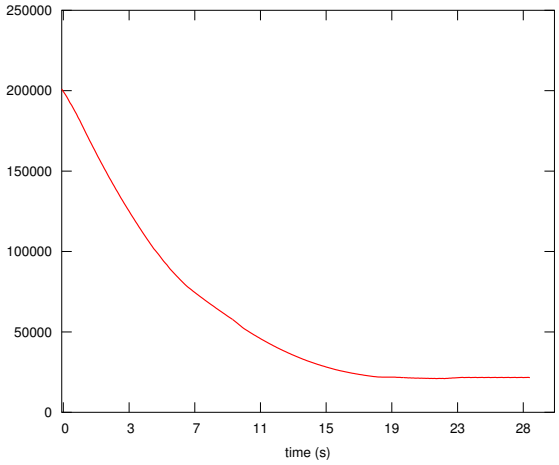
REFERENCES

[1] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[2] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. and Automat.*, vol. 20, no. 2, pp. 243–255, 2004.



(a) Final converged solution.



(b) Plot showing change in $\mathcal{H}(P, PV)$ with execution time in seconds. The converged solution was reached in 285 iterations.

Fig. 11. Coverage in an indoor environment by 4 heterogeneous robots.

[3] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*, ser. Applied Mathematics Series. Princeton University Press, 2009.

[4] J. Cortez, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESIAM: Control, Optimisation and Calculus of Variations*, vol. 11, pp. 691–719, 2005.

[5] A. Gusrialdi, S. Hirche, T. Hatanaka, and M. Fujita, "Voronoi based coverage control with anisotropic sensors," in *Proc. of the American Control Conference*, 2008, pp. 736–741.

[6] K. Laventall and J. Cortés, "Coverage cotrol by robotic networks with limited-range anisotropic sensory," in *Proc. of the American Control Conference*, 2008, pp. 2666–2671.

[7] S. Salapaka, A. Khalak, and M. A. Dahleh, "Constraints on locational optimization problems," in *Proc. of the IEEE Conference on Decision and Control*, Maui, Hawaii, USA, 2003, pp. 1744–1746.

[8] M. Schwager, D. Rus, and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, March 2009.

[9] A. Kwok and S. Martínez, "Unicycle coverage control via hybrid modeling," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 528–532, 2010.

[10] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira, *Algorithmic Foundations of Robotics VIII - Selected Contributions of the Eighth International Workshop on the Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics. Springer-Verlag, 2010, vol. 57, ch. Simultaneous Coverage and Tracking (SCAT) of Moving Target with Robot Networks, pp. 85–99.

[11] A. Kwok and S. Martínez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal of Robust and Nonlinear Control*, vol. 20, pp. 725–842, 2010.

[12] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," *IEEE Trans. on Automatic Control*, vol. 55, no. 3, pp. 749–754, 2010.

[13] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Trans. on Automatic Control*, vol. 56, no. 8, pp. 1834–1848, 2011.

[14] A. Breitenmoser, M. Schwager, J. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 4982–4989.

[15] D. Haumann, A. Breitenmoser, V. Willert, K. Listmann, and R. Siegwart, "Discoverage for non-convex environments with arbitrary obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4486–4491.

[16] C. H. Caicedo-Nunez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *Proc. of the IEEE Conference on Decision and Control*, 2008, pp. 4244–4249.

[17] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," vol. 28, no. 2, pp. 364–378, 2012.

[18] S. kook Yun and D. Rus, "Distributed coverage with mobile robots on a graph: Locational optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 634–641.

[19] S. Bhattacharya, N. Michael, and V. Kumar, "Distributed coverage and exploration in unknown nonconvex environments," in *Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems*. Springer, 2010, pp. 1–14.

[20] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proc. of the IEEE Conference on Decision and Control*, 2008, pp. 3947–3952.

[21] B. Aronov, "On the geodesic Voronoi diagram of point sites in a simple polygon," *Algorithmica*, vol. 4, no. 1, pp. 109–140, 1989.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. MIT Press, 2001.

[24] A. Breitenmoser, J.-C. Metzger, R. Siegwart, and D. Rus, "Distributed coverage control on surfaces in 3d space," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots and Systems*, 2010, pp. 5569–5576.

[25] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed., ser. Wiley Series in Probability and Statistics. New york: Wiley, 2000.

[26] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, December 1999.

[27] M. Schwager, J. McLurkin, and D. Rus, "Distributed coverage control with sensory feedback for networked robots," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, 2006, pp. 1–8.

[28] H. Flanders, "Differentiation under the integral sign," *American Mathematical Monthly*, vol. 80, no. 6, pp. 615–627, 1973.

[29] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

[30] H. Imai, M. Iri, and K. Murota, "Voronoi diagram in the Laguerre geometry and its applications," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 93–105, 1985.

[31] E. Papadopoulou and D. T. Lee, "A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains," *Algorithmica*, vol. 20, no. 4, pp. 319–352, 1998.

[32] F. Clarke, *Optimization and Nonsmooth Analysis*. New York: Wiley, 1983.

[33] W. Karush, "Minima of functions of several variables with inequalities as side constraints," Master's thesis, Department of Mathematics. University of Chicago, 1939.

[34] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 1951, pp. 481–492.