Citation: E. Teruel, R. Aragüés, G. López-Nicolás. A distributed robot swarm control for dynamic region coverage. Robotics and Autonomous Systems, vol. 119, pp. 51-63, 2019.

A Distributed Robot Swarm Control for Dynamic Region Coverage

Enrique Teruel, Rosario Aragues, Gonzalo López-Nicolás

Inst. Investigación en Ingeniería, Universidad de Zaragoza, Spain
{eteruel,raragues,gonlopez}@unizar.es

Abstract

We propose a new distributed method for coverage of a moving deformable convex region with a team of robots in a communication network. Robots execute a distributed self-deployment strategy based on Centroidal Voronoi Tesselations (CVT) to cover the region evenly while preventing collisions. The main contribution is the addition of a feedforward action to overcome the well-known slow convergence issue of the basic CVT algorithms. This action is derived by each robot from the information about the region that floods through the network from a few selected leaders. The method allows to quickly adapt to the fastly changing working area in spite of the light communication requirements, and it is well suited for large teams of expendable robots.

Keywords: Autonomous robots, Swarm, Coverage, Tracking

1. INTRODUCTION

We are interested in the following general problem: A dynamic wide area $\Omega(t) \subset \mathbb{R}^2$, possibly complex, with obstacles and narrowings, must be explored or monitored by a large team of robots, which can cover only a very small fraction each, and which are required to stay together in a sort of flexible formation. To do so, a region $Q(t)$, much smaller than $\Omega(t)$, which can be covered appropriately by the available team of robots, is moved and deformed, or manoeuvred, to rake the full area, so that $\bigcup_t \Omega(t) \subset \bigcup_t Q(t)$. It is assumed that Q(t) is appropriately covered when the robots are deployed evenly within the region Q(t), more precisely when they form a Centroidal Voronoi Tesselation (CVT). The mission of the robots is to do so while Q(t)

is being moved and deformed in a way non controlled by them, but decided at a higher level. The current information about the dynamic region is known by, or communicated to, a few robots in the team, acting as leaders, and it is flooded from them to the rest of the team, where it arrives delayed some number of hops, because communication is only possible with whichever robots are near enough. Using only this information, and the position of its neighbors, each robot must move in a fully distributed and autonomous fashion to fulfill the mission. Applications can be found in a variety of scenarios, where the wide area might be a large building, a city, a mountain range, a network of cave galleries, a river, or a sea, and the task of each robot might be measuring magnitudes, the temporal and spatial variations of which are of interest, taking images to be processed in search of lost people, accident remains or hazardous events, or to collaboratively perform actions over the area. For instance, several valleys and canyons must be explored by a team of aerial robots that fly low, adapting to the steep rocks and cliffs, commanded from a rescue helicopter, in order to help find lost people promptly; Or a team of marine robots is devoted to absorb a large moving patch of oil spilled from a crashed trunker, before it reaches the seaside, adapting to the evolving patch, reefs and coastline shapes. In the taxonomy proposed by Robin and Lacroix (2016) we are interested both in mobile search and monitoring problems, with similar applications as those that motivate, e.g. (Cortés et al., 2004; Belta and Kumar, 2004; Svennebring and Koenig, 2004; Wagner et al., 2008; Hou et al., 2009; Yoshida et al., 2014; Kolling et al., 2016), and references therein. The problem can be extended, and approached similarly, to the 3-D case, to explore large volumes, but in this paper the 2-D case is considered alone for simplicity.

We propose the dynamic region $Q(t)$ to be a convex polygon, and to cover it evenly via Centroidal Voronoi Tesselations (CVT). Due to their salient features, the application of CVT to this kind of coverage control is proposed by many authors (Du et al., 1999; Cao and Hadjicostis, 2003; Cortés et al., 2004; Liu et al., 2009; Schwager et al., 2009; Sun et al., 2011; Song et al., 2014; Hateley et al., 2015; Lee et al., 2015; Li and Liu, 2017), but to the best of our knowledge it has just started to be applied to moving regions by Tardos et al. (2018), because it faces the problem of slow convergence of CVT algorithms, making the allowable movement too slow, or risking instability. Despite being so used, Lloyd's method is complicated to analyze, and most of the results on convergence speed are experimental, or refer to the 1-D case (Du et al., 1999). In order to accelerate convergence, variational (Liu et al., 2009;

Song et al., 2014; Hateley et al., 2015) or hierarchical (Wang et al., 2016) approaches have been proposed, but they require more global information, and a greater communication plus computation effort, to make a difference. Instead, we propose using the already available local information about the region's movement to add a feedforward action. Of course, this feedforward action might also be added to any version of the CVT algorithm, but we use the distributed version of Lloyd's method (Cao and Hadjicostis, 2003; Cortés et al., 2004) as a baseline to analize the improvements of our proposal.

For simplicity, we assume that robots have been initially deployed within a static region, which can be achieved by rendezvous techniques to bring robots inside the region, and/or static CVT coverage methods to distribute them. Therefore, when the movement starts, it is assumed that the robots are deployed within the region so that the communication network is connected. The paper by Song et al. (2014) gives nice examples. The successive positions of the corners/vertices of the moving convex polygon are requested by the operator, or by intelligent or teleoperated leaders. There are many options, e.g., leaders might be at the corners, or the region could be defined around a central, overhead or ground leader, or leaders, etc. Anyway, we assume that the current position of each corner/vertex is known by one or more robots, considered "the leaders". In order to compute its next target position, each robot uses information received from a relatively small subset of neighbors. This information consists of: (1) the current position of the neighbors (this might alternatively be measured, e.g., via vision, radar, laser, etc.); and (2) recent positions of the corners/vertices of the region, continuously flooded through the network, e.g., via wireless communications originated at the leaders. For simplicity, information is assumed to be correct at the point of origin, and communications between neighbor robots, within a communication radius r, are assumed to be perfect, i.e., instantaneous and without errors. Therefore information delays are caused by the flooding mechanism alone, due to the number of hops from the leaders. It is also assumed that the robots' movements required to track the dynamic region are attainable, meaning that robots are able to reach their next targets, by means of any suitable lower level local control, during the interval of the (high level) control step.

Our work is an example of swarm control through leaders, in the terminology of Kolling et al. (2016), which opens up the forms of control available for single and multi-robot systems, and teleoperation. By swarm we understand a large group of expendable autonomous robots that use only information

from their neighbors. Part of this information refers to the working area, because a determined behavior of the team respective to this area is required: it must be completely and evenly covered. This cannot be achieved by swarm control approaches without any reference to global information, such as those by Pendleton and Goodrich (2013); Walker et al. (2013), or when a different kind of global information is used to describe the region, such as summary statistics (Freeman et al., 2006) or abstractions (Belta and Kumar, 2004), or when attention is not paid to the features of the distribution within the working area (Cheah et al., 2008; Hou et al., 2009). In Svennebring and Koenig (2004) and Wagner et al. (2008) many (ant) robots, without leaders, cover a wide area without paying attention to their distribution: they operate almost independently, using trails as a means of communication of the mission (status). They need not being directed by any leader, or stay together in any kind of formation, which is required in our case. On the other hand, compared to methods where the geometry of the formation is determined in order to cover a given region, even when some flexibility is allowed to cover a dynamic region as in Yoshida et al. (2014), in our proposal each robot decides its movement autonomously, after the information received from its current neighbors, irrespective of their identity.

The main contribution of the paper is the introduction of a feedforward action to make CVT techniques applicable to dynamic regions, without requiring additional information. To the best of our knowledge, there is no other method where a swarm covers a dynamic (convex) region evenly and accurately, in an autonomous and fully distributed fashion. Section 2 recalls the basic CVT concepts and algorithms. The feedforward action is introduced in Sec. 3, where its benefits are proven. Performance of the method and scalation are evaluated in Sec. 4 through a simulation example.

2. PROBLEM STATEMENT AND CVT COMPUTATION

Let us begin with a brief description of the coverage problem, and the basic solution strategy based on CVT computation by the Lloyd's method, which, at each step, computes Voronoi cells for current robot positions, and robots are moved towards the centroid of their respective cells. Consider n robots operating in a planar convex region $Q \subset \mathbb{R}^2$, with positions $p_i \in Q$, for $i = 1, \ldots, n$. Each robot can obtain information about the positions of its current neighbors, within its communication radius r, and viceversa, i.e., the communication graph is undirected and time-varying. Robots sense

with better quality the area nearby, so they should be evenly deployed over the area for a good coverage. More precisely, the goal is to partition Q into n disjoint regions $W = (W_1, \ldots, W_n)$, and to place the robots in positions $P = (p_1, \ldots, p_n)$ at the centroids of these regions, such that they form a CVT.

Definition 2.1. *Let $|\cdot|$ denote the Euclidean norm in $\mathbb{R}^2$. Given a set of n robots within a convex region $Q \subset \mathbb{R}^2$:*

1. *A* configuration *for Q, denoted by (W, P), is any partition of Q into n disjoint regions $W = (W_1, \ldots, W_n)$ with n robots in positions $P = (p_1, \ldots, p_n)$ within those regions.*
2. *A* Voronoi Tesselation *is a configuration such that for each i:*

$$W_i = \{q \in Q \mid |q - p_i| < |q - p_j|, \forall j \neq i\}.$$

3. *A* Centroidal Voronoi Tesselation (CVT) *is a configuration such that it is a Voronoi Tesselation with each p_i at the centroid of W_i, see e.g. Du et al. (1999).*
4. *The distance between two configurations, (W, P) and (W', P'), will be measured by the maximum distance between the closest robots, and (abusing notation) it is denoted by*

$$|P - P'| = \max\{|p_i - p'_i| \; \forall i, \text{ assuming } |p_i - p'_i| < |p_i - p'_j| \; \forall j \neq i\}.$$

Although neither communication nor control actions over robots need to be synchronized, consider for simplicity that they take place at discrete steps of duration T, hence, for convenience, time can be measured in steps.

Assumption 2.2. *Assume that the dimensions of the problem, Q and n, are such that r is large enough to reach the Voronoi neighbors, hence keeping the network connected, at least when the configuration is near a CVT, which is assumed to be the mission for the swarm. In such case, the number of robots* required *to cover Q is in the order of A/r^2, with A the area, or $(L/r)^2$, with L a characteristic length, when it resembles a regular polygon or circle. For the flooding mechanism, to communicate the region position from the leaders to all robots, it is assumed that at most h hops are required. Naturally $h < n$, and in fact it is in the order of L/r, i.e., in the order of $\sqrt{n}$.*

If transient disconnections, or communication failures, were admitted, then a larger h might be required, to accommodate the maximum number of delays/steps in the reception of the position of the vertices, instead of the maximum number of hops.

The following algorithm, a distributed version of the Lloyd's descent algorithm (Cao and Hadjicostis, 2003), reaches a CVT starting from a given initial configuration, even when the communication radius r is transiently not sufficient to reach the Voronoi neighbors (Song et al., 2014). At each step k, the Voronoi Tesselation corresponding to the current positions of the robots, $P(k)$, is computed, and their next positions, $P(k+1)$, are the centroids of the Voronoi regions:

Algorithm 2.3. *Starting at a configuration $(W, P) = (W(0), P(0))$, at each iteration k, each robot i:*

1. *Receives the position $p_j(k)$ of its neighbor robots, up to distance r, and updates the position of the vertices of the dynamic region with whichever more recent information that its neighbors have. When complete information about Q is available, robot i can start moving.*
2. *Computes its region $W_i(k)$ as the intersection between: (a) the boundaries of Q; (b) a circle, or approximate polygon, with center $p_i(k)$ and radius $r/2$; and (c) its Voronoi region V_i based on the (known) positions of the robots up to distance r,*

   $$V_i = \{q \in Q \mid |q - p_i(k)| \leq |q - p_j(k)|,\ \forall j \text{ such that } |p_i(k) - p_j(k)| \leq r\},$$

   which is a relatively straightforward geometric problem, see e.g. (Cao and Hadjicostis, 2003). The Voronoi region can be shrunk to prevent collisions between large robots moving within neighbor cells, see e.g. (Kantaros and Zavlanos, 2016).
3. *Computes the mass (area) and centroid of $W_i(k)$:*

   $$M_{W_i}(k) = \int_{W_i(k)} dq, \qquad C_{W_i}(k) = M_{W_i}(k)^{-1} \int_{W_i(k)} q dq,$$

   with closed expressions after the position of vertices, see e.g. (Cortés et al., 2004), and
4. *Moves to this centroid: $p_i(k+1) = C_{W_i}(k)$. (For instance, with integrator dynamics, $p_i(k+1) = p_i(k) + u_i(k)$, the control law might be proportional, $u_i(k) = K(C_{W_i}(k) - p_i(k))$, with gain $K = 1$.)*

This would be repeated until the distance to a CVT is as small as desired, but, since the CVT is unknown, it is repeated until a stop condition, usually in terms of the number of iterations, or in terms of the distance between consecutive configurations, $|P(k+1) - P(k)|$.

Definition 2.4. *Given a configuration* (W, P)*, the number of iterations of Alg. 2.3 to reach any CVT,* $(\bar{W}, \bar{P})$*, within an arbitrary distance* ε*, is denoted by* $S(P, \varepsilon)$*. When* P *are almost at the centroids of* W*, no iterations are required:* $S(P, \varepsilon) = 0$ *iff* $|P - \bar{P}| \leq \varepsilon$*, with* $(\bar{W}, \bar{P})$ *a CVT. When irrelevant,* ε *is omitted.*

Lloyd's iterations drive the robots closer to the CVT, step by step:

Lemma 2.5. *Consider two configurations,* (W, P) *and* (W', P')*, near a CVT,* $(\bar{W}, \bar{P})$*.* $S(P) \leq S(P')$ *iff* $|p_i - \bar{p}_i| \leq |p'_i - \bar{p}_i|$ *for every* i*, and, when for some* i $|p_i - \bar{p}_i| < |p'_i - \bar{p}_i|$*, then there is a small enough* ε *such that* $S(P, \varepsilon) < S(P', \varepsilon)$.

Proof. It follows from the fact that $(\bar{W}, \bar{P})$ is a fixed point in Alg. 2.3, see Du et al. (1999, Sec. 5.3), or Cortés et al. (2004, App. I). This algorithm can be interpreted as the minimization of a descent function $\int_Q \min\{|q - p_i| \; \forall i\} dq$, which can be interpreted as an energy function: It is minimized when the p_i are at the centroids of their Voronoi cells. □

In addition to the even spatial distribution of autonomous robots, several features of this coverage method are particularly interesting in our context:

- It is fully distributed. All computations are concurrently performed by the robots, and they are quite efficient. The computational cost per robot and iteration is linear with the number of robots placed within a distance r. Each robot requires very little information, merely the position of the nearby robots, which in the long run should include its Voronoi neighbors, and the position of the boundaries of Q.

- It inherently prevents collision problems, particulary when a safety radius around robots is used to shrink the regions, as in (Kantaros and Zavlanos, 2016), and robot trajectories are within their respective regions.

- Robot identity is unimportant, making the method resilient against the withdrawal of failed robots, i.e., a few robots are expendable.

- A density function $\rho(q)$ might be defined to weight the relative importance within Q, see (Du et al., 1999; Cortés et al., 2004). We omit this feature because it prevents the explicit computation of mass and centroid, requiring instead quadrature computations. Moreover, there are alternative methods which are better suited to approach the problem of adaptation to spatially distributed and dynamic, even unknown beforehand, relevances (Schwager et al., 2009; Sun et al., 2011).

Figure 1 illustrates Voronoi tessellations, and the convergence behaviour of Alg. 2.3 to reach a CVT. The region is a 2×1 rectangle defined by the 4 vertices, depicted in red, within which robots (black), and their Voronoi cells (cyan, with a safety area) are shown in the scenes. Dashed lines between robots indicate communication links. The communication radius is $r = 0.5$ in this example, which is little more than enough to guarantee communication to Voronoi neighbors when the configuration is near a CVT, see the bottom-right frame. In this example, the position of the vertices is only known by the robots within r, indicated by dashed red lines, and it is flooded through the communication network. Starting from all the $n = 24$ robots by the boundaries of the region, which is a worst case, after 5 steps (top-left frame) there is only one robot which has complete information about the region, so it is ready to move to the centroid of its Voronoi cell. After 10 steps (top-right) every robot is moving, although some Voronoi neighbors are not yet communicated. After 20 steps (bottom-left) a configuration that is nearly a CVT has been reached, but the final CVT is still being approached, which is revealed by the small but positive maximum distance traveled by any robot in the last step, plotted (scaled by r) in blue below the scenes. After 80 steps (bottom-right) this distance is considered to be (almost) null, so the configuration of the robots is considered to be (very near to) a CVT.

The application of CVT is very extensive (Du et al., 1999), although it faces convergence issues that are significant in large problems, in terms of n, and in dynamic applications. These problems can be circumvented, particularly in non-distributed applications, with alternative variation-based algorithms (Liu et al., 2009). They minimize a nonlinear energy function on all the robots' positions, taking advantage of knowledge about the Hessian. While they can be far more efficient than the Lloyd's method, they require a greater computational effort, and the usage of information from other robots besides the Voronoi neighbors. When the Hessian is approximated using less global information, as in Song et al. (2014); Hateley et al. (2015), the im-

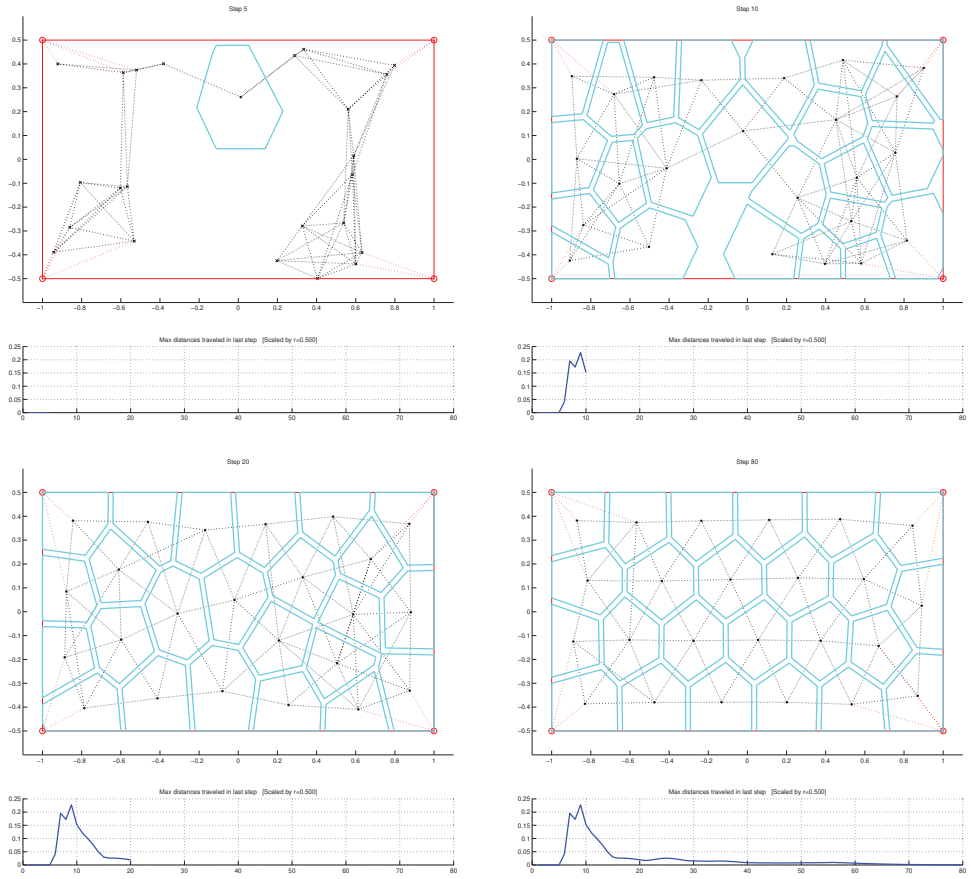

Figure 1: Configuration of 24 robots within a 2×1 rectangular region, after 5, 10, 20, and 80 iterations of Alg. 2.3, starting from all robots by the boundaries. Communications are indicated by dashed lines between neighbor robots, within the communication radius $r = 0.5$. Cyan lines show the Voronoi cells. Maximum distance traveled by any robot in the last step (scaled by r) is plotted in blue, below the scene.

provement is less significant, definitely not enough to allow relatively fast movements of Q. Another approach to speed-up the algorithms is to use hierarchical methods, starting from a coarse tessellation, and iteratively including more generators, as in Wang et al. (2016), but they also require using global information, beyond the neighborhood. Instead, we propose using the local information about Q, which must be known by the robots anyway, to compute simple but effective feedforward actions.

3. COVERAGE OF A DYNAMIC REGION

Assume now that the region where the robots are required to operate changes its position, size, and shape. The motivation to such behavior might be to rake a wide target area, or to track a target that moves and changes its size and shape. In any case, the movement is imposed to the team, it is decided at a higher level, and the mission of the robots is to deploy evenly over the given region. This region is denoted by $Q(t)$, or Q_k in discrete-time, where $t = kT$. Although the movement can be quite general actually, it is required that the successive Q_k are related by affine transformations (Byer et al., 2010, Ch. 12) in order to prove some theoretical results.

Definition 3.1. *Consider $\mathcal{A} : \mathbb{R}^2 \to \mathbb{R}^2$ an affine transformation. (Examples of affine transformations include translation, scaling, homothety, rotation, shear mapping, and compositions of them, in any combination and sequence.)*

1. *The resulting region from applying $\mathcal{A}$ to every point in Q, that is, the region after movement $\mathcal{A}$, is denoted by $Q \xrightarrow{\mathcal{A}} \mathcal{A}(Q)$. Alternatively, $\mathcal{A}$ might be regarded also as a transformation from a coordinate system into a new one, such that the coordinates of point $\mathcal{A}(p)$ in the latter system are the same of those of p in the former, see Fig. 2.*
2. *For successive movements/transformations, from Q_0, $Q_k = \mathcal{A}_k(Q_{k-1}) = \mathcal{A}_k(\mathcal{A}_{k-1}(Q_{k-2})) = \mathcal{A}_k \circ \mathcal{A}_{k-1}(Q_{k-2}) = \cdots = \mathcal{A}_k \circ \mathcal{A}_{k-1} \circ \cdots \circ \mathcal{A}_1(Q_0)$, the composition $\mathcal{A}_k \circ \ldots \circ \mathcal{A}_1$ is denoted by $Q_0 \xrightarrow{\mathcal{A}_{1,\cdots k}} Q_k = \mathcal{A}_{1,\cdots k}(Q_0)$.*
3. *A movement is δ-smooth when its changes are small, more precisely when $|(q_{k+1} - q_k) - (q_k - q_{k-1})| < \delta$, for every $q_k \in Q_k$ and every k. Notice that, for a given δ, and any given movement $Q_0 \xrightarrow{\mathcal{A}_{1,\cdots k}} Q_k$, there exists a time-scaled (slower) movement, where each step is divided into s smaller steps, $Q_0 \xrightarrow{\mathcal{A}_{1_1,\cdots 1_s,\cdots k_1,\cdots k_s}} Q_k$ such that it is δ-smooth.*

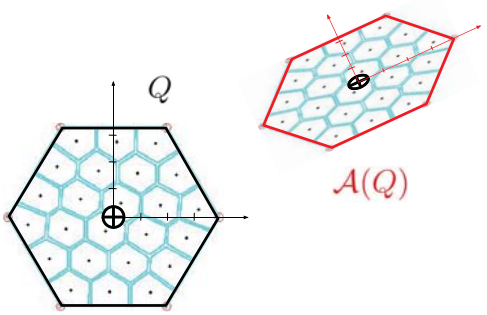

Figure 2: Illustration of the application of an affine transformation $\mathcal{A}$. Notice that the centroid of Q, which is in $(0,0)$, is transformed to the point $(0,0)$ in the transformed coordinates. It is also illustrated that a CVT for Q (cyan) is transformed to a CVT for $\mathcal{A}(Q)$.

4. *Given a configuration (W,P), the configuration reached after movement $\mathcal{A}$ is denoted by $\mathcal{A}(W,P)$.*

Lemma 3.2. *Consider Q a convex region, and $\mathcal{A}:\mathbb{R}^2\to\mathbb{R}^2$ an affine transformation, or movement:*

1. *The resulting region, $\mathcal{A}(Q)$, is also convex. Let Q_C be the centroid of Q. Then, the centroid of $\mathcal{A}(Q)$ is $\mathcal{A}(Q_C)$.*
2. *If (W,P) is a CVT for Q, then $\mathcal{A}(W,P)$ is a CVT for $\mathcal{A}(Q)$, hence $S(\mathcal{A}(P))=0$.*

Proof. Convexity and centroid conservation are properties of affine transformations (Byer et al., 2010, Ch. 12). They imply that Voronoi cells, neighbors, and centroids are preserved. It follows trivially that $S(\mathcal{A}(P))=0$. □

Figure 2 illustrates an affine movement $\mathcal{A}$ (consisting of translation, rotation, and scalation), and it shows the properties of centroid and CVT conservation.

On the contrary, in general, although (W,P) is a CVT for Q, it is not a CVT for $\mathcal{A}(Q)$. Actually the distance $|P-\mathcal{A}(P)|$ might be quite large, when the movement/transformation is noticeable. Let W' be the partition of $\mathcal{A}(Q)$ corresponding to P such that (W',P) is a Voronoi Tesselation for $\mathcal{A}(Q)$. In general, it is not a CVT: $S(P,\varepsilon)$ might be strictly positive if the induced displacements are greater than ε. In fact, $S(P)$ might be quite large if the displacements are large: notice, for instance, that with a stationary

movement $|(q_{k+1}-q_k)-(q_k-q_{k-1})|=0$, while $|q_{k+1}-q_k|=|q_k-q_{k-1}|$ might be large. Therefore, due to the slow convergence of Alg. 2.3, it is expected that its application to a region which moves at each step would require a slow movement to converge (see the first dynamic case in the video attachment, where 24 robots are unable to track a slowly moving region).

In *control terms*, $\mathcal{A}$ can be regarded as a *perturbation*, and Alg. 2.3 responds to it *reactively*, which is sufficient only when (the effect of) the perturbation is small, which is not the case for fast movements. To overcome this problem, we propose the introduction of a *feedforward action*, taking advantage of the fact that it is possible, and economic, to *estimate the perturbation* reasonably well: each robot knows about the region, perhaps with some delays, when the information arrives from distant locations through flooding. Of course, feedforward does not need to be perfect (which is in general impossible) in order to make a difference, it suffices that it reduces the error, and therefore it makes the still required job of the *feedfback mechanism* (CVT iterations) easier.

After these ideas, the following algorithm is proposed. For the notation, time (step) is indicated as a subscript, e.g., Q_k is the region at time kT, and $p_{i,k}$ denotes the position of robot i after k steps. Notice the difference wrt. the notation in Alg. 2.3, where, for instance, $(W(k),P(k))$ denoted the configuration after k Lloyd's iterations. At each step k, with robots at P_k, each robot i computes its corresponding Voronoi region, using its estimation of the current position of the region, $\widehat{Q}^i_k$, and computes its next position, $p_{i,k+1}$, as the centroid of this Voronoi region, $C_{W_{i,k}}$, plus its estimation of the next movement of this point with the region, $\widehat{\mathcal{A}}^i_{k+1}(C_{W_{i,k}})$:

Algorithm 3.3. *Starting at a configuration (W_0,P_0), in a region Q_0, which moves according to successive $Q_{k-1}\xrightarrow{\mathcal{A}_k}Q_k=\mathcal{A}_k(Q_{k-1})$, at each iteration k, each robot i:*

1. *Receives the position $p_{j,k}$ of its neighbor robots, up to distance r, and updates the position of the vertices of the dynamic region with whichever more recent information that its neighbors have. When enough information about past positions of the region is available, robot i can start moving.*
2. *Updates its estimations of $Q_{k-h+1},\dots Q_k$, $\mathcal{A}_{k-h+1},\dots\mathcal{A}_k,\mathcal{A}_{k+1}$ denoted by $\widehat{Q}^i_k$, $\widehat{\mathcal{A}}^i_k$, etc, where h is an upper bound for the number of hops (Asn. 2.2). The superscript notation stands for "as estimated by robot*

i": $\widehat{Q}_k^i$ is the estimation, by robot i at step k, of the dynamic region. It is based on the updated positions of the vertices, which correspond to past steps, between $k-h$ and k, depending on the distance from leaders.

3. *If the current position of the robot is outside its estimated region, $\widehat{Q}_k^i$, then find a close position at the boundary of the region, and let it be $p_{i,k}$.*
4. *Computes its region $W_{i,k}$ as the intersection between: (a) the boundaries of $\widehat{Q}_k^i$; (b) a circle, or approximate polygon, with center $p_{i,k}$ and radius $r/2$; and (c) its Voronoi region $V_{i,k}$ based on the (known) positions of the robots up to distance r,*

$$V_{i,k} = \{q \in \widehat{Q}_k^i \mid |q - p_{i,k}| \leq |q - p_{j,k}|,\ \forall j \text{ such that } |p_{i,k} - p_{j,k}| \leq r\},$$

which is a relatively straightforward geometric problem, see e.g. (Cao and Hadjicostis, 2003). The Voronoi region can be shrunk to prevent collisions between large robots moving within neighbor cells, see e.g. (Kantaros and Zavlanos, 2016).
5. *Computes the mass (area) and centroid of $W_{i,k}$:*

$$M_{W_{i,k}} = \int_{W_{i,k}} dq, \qquad C_{W_{i,k}} = M_{W_{i,k}}^{-1} \int_{W_{i,k}} q dq,$$

with closed expressions after the position of vertices, see e.g. (Cortés et al., 2004), and
6. *Moves to this centroid ("feedback action") plus the next estimated displacement ("feedforward action"): $p_{i,k+1} = C_{W_{i,k}} + \widehat{\mathcal{A}}_{k+1}^i(C_{W_{i,k}})$. (For instance, with integrator dynamics, $p_{i,k+1} = p_{i,k} + u_{i,k}$, the control law might be proportional, $u_{i,k} = K(C_{W_{i,k}} + \widehat{\mathcal{A}}_{k+1}^i(C_{W_{i,k}}) - p_{i,k})$, with gain $K = 1$.)*

Remark 3.4. *Some considerations about Alg. 3.3 follow:*

1. *Note that the computational and memory costs associated to the operations performed by each robot at each iteration of this algorithm are very light, they are actually linear with the number of robots placed within a distance r. This is one of the facts that makes distributed strategies such as this one so appealing.*
2. *In practice, steps between successive Q_k, at the higher level of the control hierarchy, involving measurements and communication between*

robots, would be much larger than steps between successive positions of the robots, at the lowest local level of robot motion control. Nevertheless, for simplicity, it is assumed that these steps coincide. It is understood that during such a step robots are able to reach their next targets, P_{k+1}, from their current positions, P_k, by means of a suitable low level control (most probably, with a quite faster cycle).

3. *In step 1 of the algorithm, the position of the vertices is flooded through the network of robots, from the leaders. Therefore, in step 2, robots must estimate the current position of the region from a more or less outdated information of the current position of each vertex, depending on the number of hops from the leader where the information about such vertex position originates. These delays can be as small as zero control steps, if the robot is within the communication radius of the leader, or they can be quite large, up to h, in the order of L/r steps, when the leader is most distant. Notice that, from the definition of h in Asn. 2.2, for sure Q_{k-h} is known by every robot, i.e., $\widehat{Q}^i_{k-h} = Q_{k-h}$. With any estimation such that $\widehat{Q}^i_k = \widehat{\mathcal{A}}^i_{k-h+1,\cdots k}(Q_{k-h})$, where $\widehat{\mathcal{A}}^i_{k-h+1,\cdots k}$ is affine, e.g., $(\widehat{\mathcal{A}}^i_{k-h})^h$, with $\widehat{\mathcal{A}}^i_{k-h}$ affine applied h times, it is guaranteed that $\widehat{Q}^i_k$ is convex, because convexity is preserved by affine transformations (Lemma 3.2.1), and the composition of affine transformations is also affine (Def. 3.1).*
4. *There are many options for the estimation of the movement, in step 2. For simplicity, a first order approximation of translation and rotation will be used: translation is computed as the difference between the centroids of the two latest known positions of the region, and rotation as the difference between the angles of their axis of minimum inertia, see Fig. 3. Higher order approximations, and/or including further available information about size or shape, might of course improve the estimation.*
5. *Notice that, given two different robots i and j, the estimations $\widehat{Q}^i_k$ and $\widehat{Q}^j_k$ might well differ from each other, since they might have more or less outdated information about the vertices of the region. The differences would be small when i and j are neighbors, because their latest known positions of the region vertices differ no more than one step/hop, thanks to the flooding mechanism. Naturally, both of them would, in general, differ from the current Q_k. Nevertheless, in the case, indicated in the item 2 above, that the common value h (a bound for the maximum*

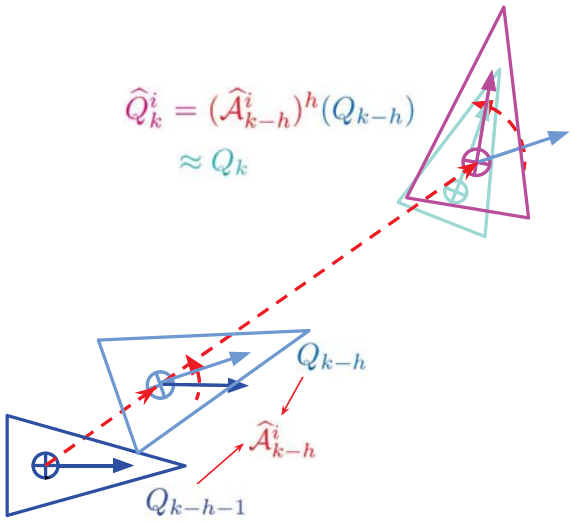

Figure 3: Estimation of $\mathcal{A}_{k-h}$ as a first order approximation of the translation between centroids plus rotation (red, dashed) between Q_{k-h-1} (dark blue) and Q_{k-h} (lighter blue), assumed to be the last two positions of the region known by robot i. With movements changing slowly in h steps, it is expected that $\widehat{Q}_k^i = (\widehat{\mathcal{A}}_{k-h}^i)^h(Q_{k-h})$ (magenta) approximates Q_k (lightest blue). In the figure, movement changes, particularly size changes, have been exaggerated, for better readability: Notice that $\widehat{Q}_k^i$ is the same size as Q_{k-h}, because $\widehat{\mathcal{A}}_{k-h}^i$ consists of translation and rotation alone, but Q_{k-h} happens to be slightly larger than Q_{k-h-1}, while Q_k is actually quite smaller.

number of hops, see Asn. 2.2) was used for all the estimations, $\widehat{Q}_k^i = (\widehat{\mathcal{A}}_{k-h}^i)^h(Q_{k-h})$*, then* $\widehat{Q}_k^i = \widehat{Q}_k^j$*, although the difference with the actual* Q_k *might be larger.*

6. *It is important to understand that applying Alg. 2.3 to the dynamic case, with successive* Q_k *instead of a fixed* Q*, is the particular case of Alg. 3.3 where for every robot* i*:* $\widehat{Q}_k^i = Q_{k-h}$*, and* $\widehat{\mathcal{A}}_k^i = 0$*.*

In the sequel it shall be proven that, thanks to the feedforward action in Alg. 3.3, the movement of the robots stably tracks the movement of the region when it is δ-smooth, i.e., when it changes slowly enough. Moreover, collisions are prevented, and the positions of the robots converge to a CVT when the movement of the region becomes stationary or stops.

In order to grasp the benefit of the feedforward action, consider first the easy case that robots are able to perfectly estimate the movement. That is, for each robot i, $\widehat{Q}_k^i = Q_k$, and $Q_{k+1} = \mathcal{A}_{k+1}(Q_k) = \widehat{\mathcal{A}}_{k+1}^i(\widehat{Q}_k^i)$. In such case, the dynamic algorithm is nothing but a transformed (by $\mathcal{A}_k$) version of the static one.

Proposition 3.5. *Start with a configuration* (W_0, P_0)*, in a region* Q_0*, which moves according to successive* $Q_{k-1} \xrightarrow{\mathcal{A}_k} Q_k = \mathcal{A}_k(Q_{k-1})$*:* $Q_0 \xrightarrow{\mathcal{A}_{1,\cdots k}} Q_k = \mathcal{A}_{1,\cdots k}(Q_0)$*. Assume that, at each iteration* k*, each robot* i *is aware of* Q_k*, i.e.,* $\widehat{Q}_k^i = Q_k$*, and it is able to compute* $\widehat{\mathcal{A}}_{k+1}^i = \mathcal{A}_{k+1}$*. Then Alg. 3.3 reaches the configuration* $(W_k, P_k) = \mathcal{A}_{1,\cdots k}(W(k), P(k))$*, where* $(W(k), P(k))$ *is the configuration reached by Alg. 2.3 when applied to a static region* Q_0 *starting with a configuration* $(W(0), P(0)) = (W_0, P_0)$*.*

Moreover, if for all i *and* k *the trajectory of robot* i *from* $p_{i,k-1}$ *to* $p_{i,k}$ *is contained in the intersection of* $W_{i,k-1}$ *and* $W_{i,k}$*, then no collisions occur.*

Proof. Recall that $\mathcal{A}_k$ can be regarded as a coordinate transformation (Def. 3.1.2), preserving CVT's (Lemma 3.2.2). Therefore, the successive configurations (W_k, P_k), obtained by Alg. 3.3, in the dynamic case, coincide with those obtained by Alg. 2.3, $(W(k), P(k))$ in the static case, after the corresponding coordinate transformation: $(W_k, P_k) = \mathcal{A}_{1,\cdots k}(W(k), P(k))$.

For the absence of collisions, the result follows, as in the static case, from the fact that the successive $W_i(k)$ are disjoint. □

Notice, at step 5 of Alg. 3.3, that it is the feedforward term that makes the difference, because in the next step, the centroid, $C_{W_{i,k+1}}$ will be at

$\mathcal{A}_{k+1}(C_{W_{i,k}})$, hence it is required to add $\widehat{\mathcal{A}}^i_{k+1}(C_{W_{i,k}})$ to obtain the centroid that corresponds (is expected to correspond) to $W_{i,k+1}$, rather than $W_{i,k}$.

Of course, in general, the assumptions of Prop. 3.5 do not hold, unless, for instance, the movement consists of a stationary translation plus rotation, the static case trivially included. In the general case, (W_k, P_k), as computed by Alg. 3.3, differs from $\mathcal{A}_{1,\cdots k}(W(k), P(k))$. Nevertheless, this difference is not significant when the movement is δ-smooth (see Def. 3.1):

Theorem 3.6. *Start with a configuration (W_0, P_0), in a region Q_0, which moves according to successive $Q_{k-1} \xrightarrow{\mathcal{A}_k} Q_k = \mathcal{A}_k(Q_{k-1})$: $Q_0 \xrightarrow{\mathcal{A}_{1,\cdots k}} Q_k = \mathcal{A}_{1,\cdots k}(Q_0)$. If the movement is δ-smooth, with δ small enough, then the successive configurations (W_k, P_k) reached by Alg. 3.3 are such that $S(P_k) = S(P(k)) = S(\mathcal{A}_{1,\cdots k}(P_0))$, where $(W(k), P(k))$ is the configuration reached by Alg. 2.3 when applied to a static region Q_0 starting with a configuration $(W(0), P(0)) = (W_0, P_0)$.*

Moreover, if for all i and k the trajectory of robot i from $p_{i,k-1}$ to $p_{i,k}$ is contained in the intersection of $W_{i,k-1}$ and $W_{i,k}$, then no collisions occur.

Proof. The result trivially holds for $k = 0$, since $(W_0, P_0) = (W(0), P(0))$. Assume now that it holds up to k, i.e., $S(P_k) = S(P(k)) = S(\mathcal{A}_{1,\cdots k}(P_0))$, and let us prove that it also holds for $k + 1$.

The goal is to prove that the distance between P_{k+1} and $P(k + 1) = \mathcal{A}_{k+1}(P(k))$ is small, in the sense that they are both equally distant to a CVT, in terms of the number of Lloyd's steps required, see Def. 2.4. Actually, it must be proven that this distance is very similar to the distance $|P_k - P(k)|$, which is small by the induction hypothesis. Considering how they are defined, or calculated, the differences can be due to:

- errors in the estimation of the current region position, $\widehat{Q}^i_k \neq Q_k$, which is used to compute the position of the centroids of the current Voronoi cells, for the "feedback action", and/or
- errors in the estimation of the next movement, $\widehat{\mathcal{A}}^i_{k+1} \neq \mathcal{A}_{k+1}$, which is used to anticipate the next displacement, for the "feedforward action".

Therefore, the goal is to prove that these errors are both as small as required, when δ is small enough.

Regarding the estimation of the region position, $\widehat{Q}^i_k$, notice that its difference with Q_k is only relevant in the case of robots (Voronoi cells) adjacent to the boundary of the region, where this estimation is effectively

used in step 4.(a) of Alg. 3.3. In the worst case, if at most $h > 0$ hops are required to propagate any information about the region, one could take $\widehat{Q}_k^i = \widehat{\mathcal{A}}^i_{k-h+1,\cdots k}(Q_{k-h})$, hence the error in the estimation of the region position is due to the error in the estimation of the last h steps of the movement, because $Q_k = \mathcal{A}_{k-h+1,\cdots k}(Q_{k-h})$. For a δ-smooth movement, where changes in successive $\mathcal{A}_k$ are bounded by δ, this estimation error is bounded, because estimations are based on previous values. Naturally, the actual errors depend both on the actual derivatives/changes of the movement, the quality of the estimation, and the maximum number of hops, h, which depends on the size of the region (and the number of robots), and the distance of a robot from the leaders. In any case, there would be a difference between the position of the current centroid of the "true" Voronoi cell of the robot, and its position as computed by the algorithm, but this difference would be small in a δ-smooth movement, as small as required with δ small enough.

Concerning the estimation of $\mathcal{A}_{k+1}$ by $\widehat{\mathcal{A}}^i_{k+1}$, which might cause a deviation in the position of the next centroids, notice that the estimation error is small in a δ-smooth movement, as small as required with δ small enough, because changes in successive $\mathcal{A}_k$ are bounded by δ, and estimations are based on previous values.

In summary, the differences between (W_{k+1}, P_{k+1}) and $(W(k+1), P(k+1))$, which would depend on δ, are small when δ is small. In particular, for a given ε, an appropriate δ exists such that $S(P_{k+1}, \varepsilon) = S(P(k+1), \varepsilon)$.

For the absence of collisions, in the case that all estimations are derived from the same information, $\widehat{Q}_k^i = \widehat{\mathcal{A}}^i_{k-h+1,\cdots k}(Q_{k-h})$, see Rmk. 3.4.4, then the successive $W_{i,k}$ are disjoint, hence so they are the robot trajectories. In the case when the most recent information is used, it would be required to shrink the cell by a sufficiently large safety area, to guarantee that adjacent $W_{i,k}$ are disjoint. Since the difference in the estimations of two adjacent robots is small (their informations differ in one hop, at most, see Rmk. 3.4.4), then this safety area would be in the order of the maximum change in the distance that the region moves in one step, which should be much smaller than a characteristic length of the cells $W_{i,k}$, that is, much smaller than r. □

Notice, again, that it is the feedforward term that makes the difference, because otherwise, i.e., with a $\widehat{\mathcal{A}}^i_k = 0$ (see Rmk. 3.4.5), the "estimation error" would be as large as the displacements induced by $\mathcal{A}_k$, $\max\{|q_{k+1} - q_k|\}$, which can be very large in spite that the movement is δ-smooth, or even stationary. This is even worse, some h times worse, for h steps.

In the particular case that (W_0, P_0) is already a CVT for Q_0, the successive (W_k, P_k) are CVT's for the respective Q_k, when the movement is δ-smooth:

Corollary 3.7. *Start with a CVT configuration (W_0, P_0), in a region Q_0, which moves according to successive $Q_{k-1} \xrightarrow{\mathcal{A}_k} Q_k = \mathcal{A}_k(Q_{k-1})$: $Q_0 \xrightarrow{\mathcal{A}_{1,\cdots k}} Q_k = \mathcal{A}_{1,\cdots k}(Q_0)$. For a given ε, a δ exists such that, if the movement is δ-smooth, then every configuration (W_k, P_k) reached by Alg. 3.3 is a CVT of Q_k.*

In practice, the movement is not required to be as slow as the proof of Th. 3.6 appears to require: Even when there are errors in the estimation of the successive $\mathcal{A}_k$, or they increase transiently, if they do not become too large, then Alg. 3.3 produces successive (W_k, P_k) which do not get too far from the corresponding $(W(k), P(k))$, and when the movement becomes stationary, or ceases, a CVT is reached after a few Lloyd's iterations, and it is kept from then on, as the Fig. 7 later on shows. The proof schema indicates that, in practice, for a given problem size, particularly for a given maximum number of hops, h, a movement that changes relatively fast, with significant changes ocurring in h steps or less, will be difficult to track. All this is better illustrated in the next section, by means of some simulation cases.

4. PERFORMANCE EVALUATION

In this section the benefits, and also the limits, of the method will be illustrated by a simulation example. A selected few of these simulation cases are provided on the video attachment. Essentially, the goal is to demonstrate that feedforward in Alg. 3.3 allows to track a region that moves fast, compared to the inability of the conventional CVT algorithm (Alg. 2.3, or Alg. 3.3 with $\widehat{Q}_k^i = Q_{k-h}$, and $\widehat{\mathcal{A}}_k^i = 0$). Time-scalation, or a speed factor, s, with the interpretation given in Def. 3.1.3, shall be used to make a movement slower (with the same duration of the steps), or smoother (with shorter time steps).

It is required to first understand the case, scalation in time and size, and the performance figures, in order to be able to interpret the plots in Subs. 4.4, where the results are discussed to illustrate the following facts:

1. The introduction of feedforward allows to track a dynamic region that moves much faster than what the basic CVT algorithm is able to track, particularly when the position of the region is immediately known by

all robots, which is a reasonable assumption in many practical cases (this information is broadcasted from a leader).

2. Even when the information about the dynamic region reaches the robots with delays due to flooding from distant leaders, the introduction of feedforward increases significantly the speed of the region that the method is able to track.
3. As it should be expected, performance degrades with the size of the problem, because flooding requires more hops. Nevertheless, feedforward allows to track significantly larger regions, with many more robots.

In summary, feedforward allows to track the same region moving much faster, a much larger region moving at the same speed, or both: a larger region moving faster, but not that much, unless the current position of the region is broadcasted to every robot in the team.

4.1. Case description

In the simulations performed in this section, the initial region is a 2×1 rectangle defined by $m = 4$ vertices, see for instance step 20 in Figure 4, with vertices numbered inside red circles. At $t = 0$, after $s6$ initial steps, while an initial CVT is still being approached starting from all the robots at the boundaries, which is a worst case, the region to be covered starts to change, perturbing the CVT convergence process. First the region just translates, it starts to rotate $s60$ steps later, and $s60$ later it also shrinks and expands. The velocities (of translation, rotation, or resizing) vary with time sinusoidally (see attached video). Notice that, for the same size of the region, the larger the value of s, the slower the movement. Figure 4 shows the region at several steps, for the case with $n = 24$ robots and $s = 4$.

Each robot is aware of the current position of any other robot within its communication radius, r. Figures 5, 6, and 8 show simulations with different policies for propagating the positions of the vertices. In Fig. 5 these positions are broadcasted to all the robots. This prevents the delays due to flooding, hence it makes the tracking task much easier, while it is a realistic assumption for many applications. In Figs. 6 and 8, only the robots within the communication radius of the leaders at the vertices are instantly aware of their position, while the rest must wait to update their positions by a flooding propagation. Since correct information is available at the corners, then the central robots receive it more or less simultaneously from the equally distant

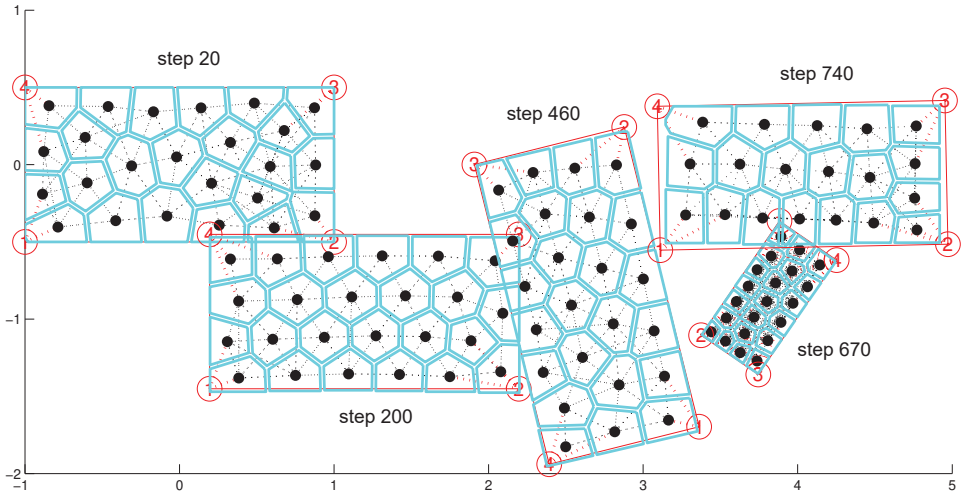

Figure 4: Five frames during the movement, with $n = 24$ robots, and speed factor $s = 4$. Communications are indicated by black dashed lines between robots, and red dashed lines lines with leaders. Cyan lines show the Voronoi cells at each step. Observe that in the first frame (step 20), just before starting the movement, a CVT is not yet reached — it coincides with the second configuration shown in Fig. 1. In step 670 only 21 robots remain, and another one is removed before step 740.

corners, while a robot near a corner has recent information about its position, but receives the information about the opposite one with the greatest delay, in some L/r steps: The number of hops is quite large when the region (hence the number of robots) is large, recall that L/r is in the order of $\sqrt{n}$. The maximum number of hops effective during each step is shown in the videos, and the maximum value, h, is recorded, and it is indicated in the legends of the plots below.

By the end of the experiment, it is simulated that some robots disappear, one by one, up to some 20% of the team, to illustrate the nice robustness of the swarm against individual failures. For instance, for the case of $n = 24$, several robots are removed, one by one, every $s12$ steps starting at $t = s126$.

The basis for the feedforward action is the estimation of the movement of the dynamic region, derived from the latest two known positions of the region. First-order approximations of the translation and rotation speeds are used: They are obtained from the change of the position of the mass center, and the change of orientation of the axis with minimal inertia, respectively, of m identical point masses placed at the region corners. Notice that this feedforward action is deliberately simple, to illustrate robustness against imperfect estimations. For instance, it does not include size changes, although these transformations/movements occur during the experiments. As with the use of a greater order estimation of the translation and rotation speeds, reasonable corrections accounting for size and shape changes could be introduced, to improve performance.

4.2. Time and length scales, and scalation in n

Time is measured in number of steps, of duration T. In order to make the movement slower, the number of steps required for the same movement is multiplied by a speed factor s. To compare performance with different speeds, the time axis will be scaled with sT. Recall that, in these simulations, at each step all robots perform one iteration of the algorithm (communications, Voronoi computations, and movement). The methods are expected to fail tracking a region that moves too fast (s too small), particularly when feedforward is not used.

It would be natural to fix r, which is the communication radius, hence the area would grow with n, and lengths with $\sqrt{n}$. Nevertheless, for graphic convenience, the area is fixed. Therefore, to investigate scalation, when n increases by 4, r is reduced to its half: with $n = \{6, 24, 96, 384\}$ robots, $r = \{1, 0.5, 0.25, 0.125\}$.

4.3. Performance figures: Displacements and distances

In Figs. 5, 6, and 8, a couple of performance measures are plotted. Time is always scaled by sT, and distances by r.

The top plot in each figure depicts the maximum (scaled) displacement of any robot at each step (dashed for the leaders): e/r. The typical condition to stop iterating static CVT algorithms, where "leaders" don't move, is that these displacements are very small, because for a static region these displacements are the actuating errors. Since the displacement e in one step of duration T is normalized by r, e/r can also be interpreted as the instantaneous velocity, in communication radius per control period. To give a physical impression, assuming $r = 100$ meters, and $T = 2$ seconds, a velocity of, say, 0.27 (27% of the communication radius r is traveled in T seconds) is 13.5 m/s, almost 50 km/h.

Naturally, the displacements at each step should be small compared to r ($e/r << 1$), because otherwise robots would attempt to move in one step to the area beyond their communication range. Therefore, reasonable velocities are bounded by $e/r < 1$. In the examples below, e/r reaches the maximum value 0.8 in some case, which is an extreme speed, practically unreasonable.

In these plots, the disappearance of robots is often revealed by narrow peaks in the (scaled) maximum distance traveled by any robot, e/r, compared to that of any leader, because when a robot disappears the neighbors must move to "fill the gap", and they do so in a few steps. This can be appreciated in the three figures, but it is particularly apparent in Fig. 6, at $t = s126$ and $t = s174$.

The plot below shows the minimum, mean, and maximum distances between connected robots. This performance measure has been selected because its evolutions are very intuitively informative of the behaviour of the system: Almost constant mean distances with small and constant deviations indicate that robots are deployed evenly and the configuration is kept. Large minimums ($d/r >> 0$) show that collisions are far to occur. The fact that d is not much smaller than r (notice that d/r is between 0.5 and 1 when the region is not shrunken) indicates that the density of communications is low, each robot barely reaches its Voronoi neighbors.

The different experiments (simulations) are identified by colors, and their main parameters are summarized in the legends: n, r, h, s, and also the maximum velocity of the region (of some vertex), in communication radius per control period.

- $n = 24$, $r = 0.5$, $h = 0$ (leaders broadcast), $s = 16$, $v_{max} = 0.05$, without feedforward.
- $n = 24$, $r = 0.5$, $h = 0$ (leaders broadcast), $s = 1$, $v_{max} = 0.8$, with feedforward.

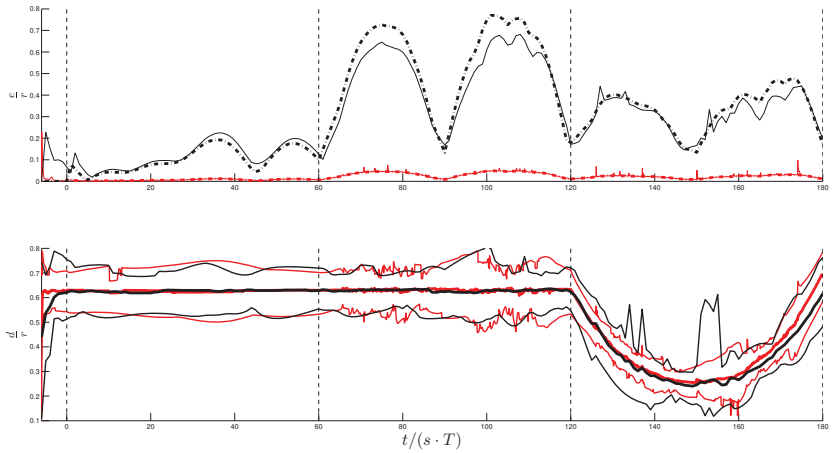

Figure 5: When the information about the region is broadcasted, hence no hops are required, performance is highly improved by the proposed method: Compare the case without feedforward and a very slow movement ($s = 16$, red) to the fast case with feedforward ($s = 1$, black). Notice that the movement couldn't be faster: maximum displacements e in one control period (of duration T) are almost as large as r. Top: the normalized maximum displacement of any robot at each step (e/r), dashed for the leaders, and bottom: normalized minimum, mean, and maximum distances between connected robots (d/r), see Subs. 4.3.

4.4. Discussion on the results

Firstly the necessity of the feedforward action is illustrated with the case $n = 24$ with broadcast. Figure 5 shows a relatively successful case without feedforward, that requires a very slow movement ($s = 16$, red), and a most fast case with feedforward ($s = 1$, black). Notice that speed (0.8 communication radius per control step) is extreme in the later case, because displacements approach r. A larger case with the same speed ($n = 96$, $s = 2$) is included in the video attachment. Observe, in the plot at the bottom of Fig. 5, that in both cases the distances between robots deviate in a similar amount, revealing similar difficulties to keep an even deployment.

Fortunately, the method with feedforward is able to perfectly keep the

- $n = 24$, $r = 0.5$, $h = 8$, $s = 16$, $v_{max} = 0.05$, without feedforward.
- $n = 24$, $r = 0.5$, $h = 8$, $s = 3$, $v_{max} = 0.27$, with feedforward.

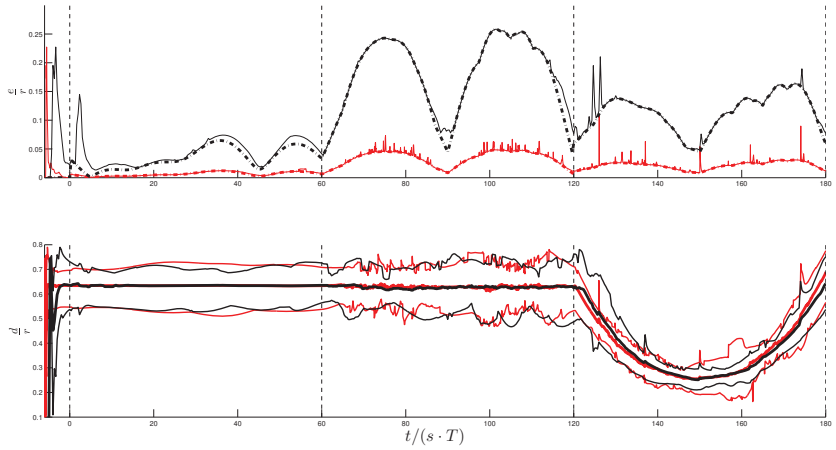

Figure 6: When the information about the region is flooded through the network, and several hops are required (up to 8 here), performance is still greatly improved by the proposed method: Compare the case without feedforward and a very slow movement ($s = 16$, red) to the fast case with feedforward ($s = 3$, black). Top: the normalized maximum displacement of any robot at each step (e/r), dashed for the leaders, and bottom: normalized minimum, mean, and maximum distances between connected robots (d/r), see Subs. 4.3.

even deployment with a reasonably fast movement, even without broadcast, see Fig. 6. Compared to the case without feedforward and a very slow movement ($s = 16$), depicted in red, feedforward, depicted in black, allows to track a quite faster movement ($s = 3$, shown also in the video attachment, after the case without feedforward and $s = 8$, which fails).

In order to illustrate the ability of the method to cope with the perturbation that a movement can be interpreted to be, the following experiment was set, for the case $n = 24$: Starting from a CVT for the original region, $P(0)$, the distances between the current configuration, P_k, and the original CVT transformed by the movement, $\mathcal{A}(P(0))$, which is the "ideal" or "non-perturbed" CVT in the sense of Prop. 3.5, Th. 3.6, and Cor. 3.7, are plotted in Fig. 7, for the cases with and without feedforward, and $s = 8$ (black and red, respectively). Observe that the case without feedforward is unsuccessful,

communication with some leader is lost eventually, before $t = 100s$, while feedforward succeeds to keep deviations in check, although, in the end, after the movement suddenly stops at $t = 180s$, the CVT that is reached is not $\mathcal{A}(P(0))$: CVT's are not unique (the energy function is nonlinear, CVT's are local minima), so when the original configuration is perturbed beyond some point, the method might converge to a different CVT, as it happens in this case. In terms of Cor. 3.7, it is expected that with a slower (smoother) movement, the original CVT is never lost. This is confirmed in Fig. 7 by the case $s = 16$ with feedforward (blue). Actually, with $s = 16$, the movement is sufficiently slow so that even the method without feedforward succeeds, although from the beginning the configurations are somehow distant from the original one, as illustrated bu the relatively high values in the magenta plot. It must be emphasized that, when feedforward is used, such a slow movement is not required in practice, as the case $n = 24$ and $s = 8$ demonstrates, not to mention the still faster case $s = 3$ in Fig. 6.

Finally, we illustrate scalation in the number of robots, hence the size of the region. The purpose is twofold: On the one hand, it is shown that the method scales well, and the addition of feedforward allows tracking larger regions. On the other hand, it illustrates the expected fact that with larger sizes, requiring larger number of hops in the communication, the allowable speeds are limited.

Figure 8 shows cases with $n = 6, 24, 96, 384$ robots, depicted in green, black, blue, and magenta, respectively. The case $n = 24$ and $s = 16$ without feedforward has been repeated from Fig. 6, depicted in red, for comparison. The speed factors are about the maximum ones that allow a successful tracking in each case. Actually, in all cases a value of $s/2$ results eventually in a loss of communication with some of the leaders, so the tracking is not successful, while the deployment is kept almost perfectly even with a value of $2s$. Concerning instantaneous velocities, recall that they are not only related to s, but also to the sizes: with four times as many robots to cover a four times wider area, lengths are double (but r is halved to keep the size of the region in videos, for graphic convenience). Observe that, compared to the case without feedforward (red), it is possible to successfully track an identical region that moves more than five times faster (black), a four times larger region, with four times more robots, moving four times faster (blue), or a sixteen times larger region that moves twice faster (magenta)!

Increasing the size it is expected that the performance degrades somehow, because more hops are required for flooding (up to $h = 3, 8, 15, 30,$

- $n = 24$, $r = 0.5$, $s = 16$, $v_{max} = 0.05$, with feedforward.
- $n = 24$, $r = 0.5$, $s = 8$, $v_{max} = 0.1$, with feedforward.
- $n = 24$, $r = 0.5$, $s = 16$, $v_{max} = 0.05$, without feedforward.
- $n = 24$, $r = 0.5$, $s = 8$, $v_{max} = 0.1$, without feedforward.

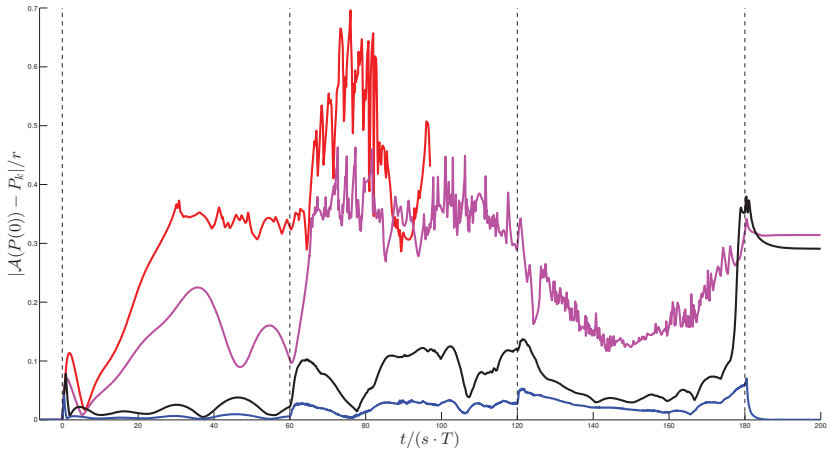

Figure 7: Feedforward reduces the effect of the perturbation due to the movement of the region. Starting from a CVT $P(0)$, these plots show the distance between the current configuration, and $\mathcal{A}(P(0))$, the original CVT transformed by the movement. Following the ideas in the proof of Th. 3.6, and the statement of Cor. 3.7, it is demonstrated that the original CVT is preserved using feedforward (blue) when the movement is smooth enough ($s = 16$). Naturally, the movement does not need to be so slow, when feedforward is used, although it might happen that coverage is achieved by a CVT which differs from the original one (black).

- $n = 24$, $r = 0.5$, $h = 8$, $s = 16$, $v_{max} = 0.05$, without feedforward.
- $n = 384$, $r = 0.125$, $h = 30$, $s = 32$, $v_{max} = 0.1$, with feedforward.
- $n = 96$, $r = 0.25$, $h = 15$, $s = 8$, $v_{max} = 0.2$, with feedforward.
- $n = 24$, $r = 0.5$, $h = 8$, $s = 3$, $v_{max} = 0.27$, with feedforward.
- $n = 6$, $r = 1$, $h = 3$, $s = 1$, $v_{max} = 0.4$, with feedforward.

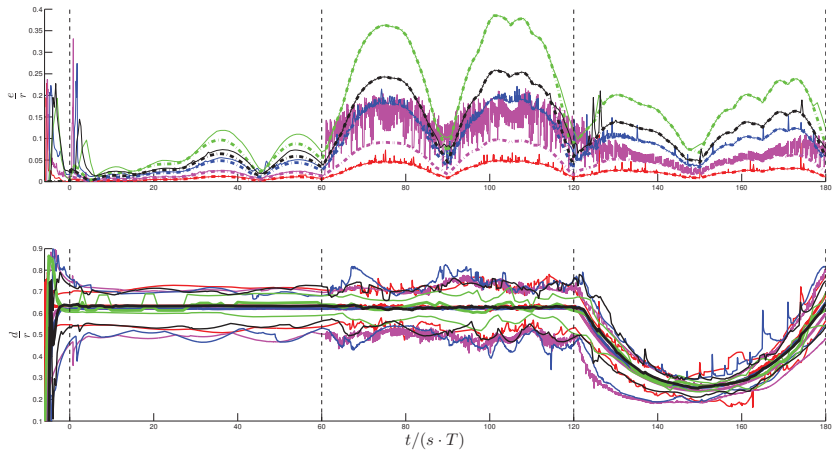

Figure 8: Case $n = 6, 24, 96, 384$ with feedforward, with $s = 1, 3, 8, 32$, in green, blue, black, and magenta, respectively. Case $n = 24$ without feedforward, with $s = 16$, in red. Top: the normalized maximum displacement of any robot at each step (e/r), dashed for the leaders, and bottom: normalized minimum, mean, and maximum distances between connected robots (d/r), see Subs. 4.3.

respectively). Notice that the maximum speed had to be limited (up to $v_{max} =$ 0.4, 0.27, 0.2, 0.1, respectively) and the distances between robots were not kept so even, with more erratic movements (compare from the green to the magenta plots).

In order to further illustrate that the speed limitation stems from the number of hops, three different cases tracking the same dynamic region moving at the same velocity with 96 robots are shown in Fig. 9. We start with the case already shown in Fig. 8 (blue), with communication radius $r = 0.25$, a value chosen to be just sufficient to reach the neighbors, which is a worst case in terms of density of robots covering the region. With the size of the

region, the number of hops h required to communicate the region position from the leaders is up to $h = 15$. In the second case shown in Fig. 9 (red) the communication radius is $r = 1$, which reduces the maximum number of hops to $h = 2$. This would correspond to applications where the sensing/acting radius of robots was smaller than the communication radius, so the region needs to be more densely populated than required to keep the network connected, with the side effect that communications are more dense. Finally, the third case (green) shows the behavior when the region position is immediately broadcasted to every robot from the leaders, so no hops are required. Notice that with less number of hops, the resulting deployment is more even, and less erratic robots' movements are performed (compare from the blue to the green plots), that is, robots fulfill the mission better and more easily. Actually, with $r = 1$ or when the region is directly broadcasted to all the robots ($h = 0$) it would be possible to track the same region moving quite faster than what can be allowed with $r = 0.25$ (see the video attachment).

5. CONCLUSION AND FUTURE WORK

We have introduced a practical method to make CVT affordable to distributed control of large swarms of robots that only communicate locally with their neighbors in mobile search and monitoring problems. It can be applied effectively to dynamic and deformable regions, where the arbitrary shape of the region is only required to be convex. We consider a distributed framework where the position of the region is flooded from the leaders to the robots. As expected, the number of hops to receive the information limits the allowable speed of change of the region. Nevertheless, due to the proposed feedforward-based approach, the system can cope with increasing speed of the covered region, it is scalable with the number of robots, and robustness to the loss of robots is provided. In fact, simulation results demonstrate that without the feedforward term the dynamic coverage task is unfeasible if the region is much larger or moves much faster, whereas our approach is able to perform well.

Besides addressing the 3-D case, further avenues of research include the evaluation of the approach considering practical variations of the proposed method, such as allowing transformations which are not affine, asynchronous communications with occasional failures, sensors with noise, or robots with detailed dynamics. Also, in order to alleviate the effect of flooding delays, information about the vertices could be used as soon as possible, as opposed

- $n = 96$, $r = 0.25$, $h = 15$, $s = 8$, $v_{max} = 0.2$, with feedforward.
- $n = 96$, $r = 1$, $h = 2$, $s = 8$, $v_{max} = 0.2$, with feedforward.
- $n = 96$, $r = 0.25$, $h = 0$ (leaders broadcast), $s = 8$, $v_{max} = 0.2$, with feedforward.

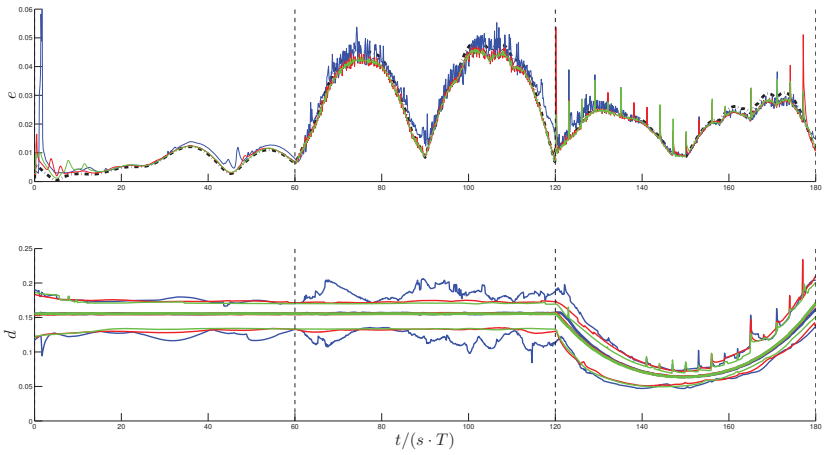

Figure 9: Case $n = 96$ with feedforward, with $h = 15, 2, 0$, in blue, red, and green, respectively. Top: the maximum displacement of any robot at each step (e), dashed black for the leaders, and bottom: minimum, mean, and maximum distances between connected robots (d).

to each robot waiting to receive the complete information of the region before using it, as required for the theoretical analysis of the method. From some preliminary tests of these practical variations, the method appears to be very robust.

References

C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5):865–875, 2004.

O. Byer, F. Lazebnik, and D. Smeltzer. *Methods for Euclidean Geometry*. Mathematical Association of America, 2010.

M. Cao and C. Hadjicostis. Distributed algorithms for Voronoi diagrams and application in ad-hoc networks. Technical report, 2003. URL `http://www.eng.ucy.ac.cy/hadjicostis/JournalPapers/voronoi_1.pdf`.

C. Cheah, S. Hou, and J. Slotine. Region following formation control for multi-robot systems. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3796–3801, 2008.

J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage Control for Mobile Sensing Networks. *IEEE Transactions on Robotics and Automation*, 20(2): 243–255, 2004.

Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999.

R. Freeman, P. Yang, and K. Lynch. Distributed estimation and control of swarm formation statistics. *Proceedings of the 2006 American Control Conference Minneapolis, Minnesota, USA, June 14-16, 2006*, pages 749–755, 2006.

J. Hateley, H. Wei, and L. Chen. Fast Methods for Computing Centroidal Voronoi Tessellations. *Journal of Scientific Computing*, 63(1):185–212, 2015.

S. Hou, C. Cheah, and J. Slotine. Dynamic region following formation control for a swarm of robots. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1929–1934, 2009.

Y. Kantaros and M. Zavlanos. Distributed communication-aware coverage control by mobile sensor networks. *Automatica*, 63:209–220, 2016.

A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis. Human Interaction with Robot Swarms: A Survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2016.

S. Lee, Y. Diaz-Mercado, and M. Egerstedt. Multirobot Control Using Time-Varying Density Functions. *IEEE Transactions on Robotics*, 31(2):489–493, 2015.

W. Li and Y. Liu. Dynamic coverage control for mobile robot network with limited and nonidentical sensory ranges. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 775–780, 2017.

Y. Liu, W. Wang, B. Lévy, F. Sun, D. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4):1–17, 2009.

B. Pendleton and M. Goodrich. Scalable Human Interaction with Robotic Swarms. In *AIAA Infotech@Aerospace (I@A) Conference*, pages 1–13, 2013.

C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.

M. Schwager, D. Rus, and J. Slotine. Decentralized, Adaptive Coverage Control for Networked Robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.

Y. Song, B. Wang, Z. Shi, K. Pattipati, and S. Gupta. Distributed algorithms for energy-efficient even self-deployment in mobile sensor networks. *IEEE Transactions on Mobile Computing*, 13(5):1035–1047, 2014.

W. Sun, L. Dou, and H. Fang. Cooperative pollution supervising and neutralization with multi-actuator-sensor network. *Acta Automatica Sinica*, 37(1):107–112, 2011.

J. Svennebring and S. Koenig. Building terrain-covering ant robots: A feasibility study. *Autonomous Robots*, 16(3):313–332, May 2004.

J. Tardós, R. Aragues, C. Sagüés, and C. Rubio. Simultaneous Deployment and Tracking Multi-Robot Strategies with Connectivity Maintenance. Sensors 18(3), 927, 2018.

I. Wagner, Y. Altshuler, V. Yanovski, and A. Bruckstein. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research*, 27(1):127–151, 2008.

P. Walker, S. Amraii, M. Lewis, N. Chakraborty, and K. Sycara. Human control of leader-based swarms. In *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pages 2712–2717, 2013.

L. Wang, F. Hétroy-Wheeler, and E. Boyer. A hierarchical approach for regular centroidal voronoi tessellations. *Computer Graphics Forum*, 35(1): 152–165, 2016.

K. Yoshida, H. Fukushima, K. Kon, and F. Matsuno. Control of a group of mobile robots based on formation abstraction and decentralized locational optimization. *IEEE Transactions on Robotics*, 30(3):550–565, 2014.