E. Teruel, R. Aragüés, G. López-Nicolás. A Practical Method to Cover Evenly a Dynamic Region with a Swarm. IEEE Robotics and Automation Letters 6(2): 1359-1366 (2021)

A Practical Method to Cover Evenly a Dynamic Region with a Swarm

Enrique Teruel Rosario Aragues Gonzalo López-Nicolás

***Abstract*— Many applications require exploring or monitoring a region. This can be achieved by a sensor network, a large team of robots which can cover only a very small fraction each. When the region is convex, small, and static, it suffices to deploy the robots as a Centroidal Voronoi Tessellation (CVT). Instead, we consider that the area to cover is wide, not necessarily convex, and complex. Then, a smaller simple region is maneuvered and deformed to rake the full area. A few waypoints describing the region along time are provided to the robots. The goal is that the robots coordinate to dynamically deploy over this region evenly, near a CVT. Unfortunately, the distributed CVT computation algorithm converges too slowly for such exploration method to be practical. In this work, CVT computation is complemented with *feedback* and *feedforward* based control techniques, and *dynamic consensus*, to adjust the robot speeds so that they coordinate to cover the dynamic region. We demonstrate in simulation that the proposed method succeeds to achieve the goal of tracking the region, with the robots evenly deployed, while keeping the connectivity and avoiding collisions. We also compare the performance of the proposed method versus other alternatives.**

I. INTRODUCTION

In the taxonomy proposed by [1] we are interested both in mobile search and monitoring problems, with similar applications as those that motivate, e.g., [2]–[10], and references therein. In this work we propose a method to cover evenly a dynamic region. This region, and its changes, is given by the (human or intelligent) leader or operator, to remove from the team difficult decisions about its borders, while the operator cognitive effort to control n robots is reduced from something above O(n) to O(1). To make the method practical, the allowable velocity of region's changes is an issue. The proposed method has immediate applications in search and rescue, 3-D mapping, and monitoring of protected environments. 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. For such a mission, it is required that the swarm is kept together, to receive new commands and to take almost simultaneous photos of the same scene from different points of view.

In these scenarios, it is interesting to use ideas from swarm robotics, that achieve useful high level behaviors through the cooperation of relatively simple autonomous robots situated in the environment, with local sensing and communication capabilities, intending to be inherently reliable [11]. Single robots are interchangeable and expendable. In order to distribute them evenly on the region to cover, they can be placed near a *Centroidal Voronoi Tessellation (CVT)* [12]. Distributed implementations of the CVT algorithm [13] have been applied to cover a convex region [2], [10]. However, when the region to cover changes along time, classical CVT methods are impractical due to their slow convergence speed.

In our previous work [14], this problem is faced exploiting the past information about the region pose to incorporate a feedforward action. There, it is shown that the method can successfully track a region moving much faster than with the basic CVT algorithm (*Lloyd's method*) alone. We consider that robots *successfully track* the region if the network stays continuously connected, agents do not collide, and they are deployed so that the coverage of the region is high. However, [14] required that the region to be covered was communicated to the robots step by step, and it had to be guessed a priori how fastly the region should change from one step to the next. Here, we introduce a practical method to adapt the speed of the region movement to what the swarm can correctly follow. From a few sparse specified *waypoints*, the robots cooperate to decide if they can move at maximum speed from one to the next, or they have to slow down.

A. Related Work

In some methods to tackle a similar problem, the geometry of the formation is determined, allowing some flexibility [7]. On the contrary, in our proposal robots behave as a swarm: Each robot decides its movement autonomously, after the information received from its current neighbors, irrespective of their number or identity, and the place of each individual robot within the region is not important.

In [6], the problem of region-based shape control for swarms of robots is approached by maintaining minimum relative distances between them. In that approach, following the research line [4], [5], the shape of the desired region needs to be defined with an appropriate objective function that guarantees convergence through Lyapunov analysis. However, the flexibility is limited since each desired shape requires a particular objective function definition, whereas our proposal is general for any convex shape, and able to directly deal with shape transitions. Unlike [6], our proposal provides scalability in the number of robots and homogeneous distribution of the swarm within the working area, which may be necessary in many applications.

In [9], the problem of making a swarm to acquire a desired shape is addressed. Given the desired shape, they use auction mechanisms to assign individual robots to individual terminal positions. Then, robots move between these configurations according to collision free paths. There are

The authors are with the Inst. Investigación en Ingeniería, Universidad de Zaragoza, Spain, e-mail: {eteruel,raragues,gonlopez}@unizar.es. This work was supported by the Spanish Government/European Union through project PGC2018-098719-B-I00 (MCIU/AEI/FEDER, UE), and project COMMANDIA SOE2/P1/F0638 (Interreg Sudoe Programme and European Regional Development Fund, ERDF).

several differences between [9] and our proposal. First, we are not interested in particular robots achieving particular destinations. Instead, our mission is satisfied if the whole desired shape is uniformly covered by our robots, regardless of their identities. This makes our method lighter in terms of the computation required to solve the assignment problem. In addition, our method copes in a more natural way with the sudden disappearance of robots, due to ocassional robot (or sensor) failures, since we do not need to re-compute the number and positions of targets. Second, [9] deals with sequential reconfigurations (transitions between a set of static configurations), while we are focused on the dynamic case, where the region to be covered changes all along the execution. Thus, we address the problem of correctly tracking and adapting to these variations.

In [10], coverage of a long region by deploying and moving the sensors, is achieved maximizing a sensing function. The followed strategy uses two virtual guidance points, located in the rear and front areas, to control the dynamics of the moving region. It is first assumed in [10] that all agents communicate with each other, whereas we focus in scalability of the system, with a fully distributed approach.

Robustness against robots' failures is inherited from the CVT deployment strategy in static regions. Note that, as agents deploy over a region, they expand to cover larger regions. Thus, even if the agents start separated or disconnected, they eventually get closer to each other and establish communication links [17]. When the region is not static but it moves, in order to keep this property, robots must move at a very slow speed to avoid disrupting substantially the near-CVT formation. This problem is addressed in this work, in order to move the formation as fast as possible.

B. Contributions and Organization of the Paper

The main contribution of this paper is the introduction of a practical method to regulate automatically the speed of a swarm of robots to cover a dynamic region specified by a few waypoints in its trajectory. The problem statement and assumptions are collected in Section II. The robots deploy autonomously as a near-CVT formation within the region, using the method introduced in [14], recalled in Section III. The next waypoint is known in due time by every robot, which uses also the information received through the network from the rest of the swarm to decide each next step, not diverging from each other thanks to a dynamic consensus. The velocity regulation is presented in Section IV, leading to a distributed algorithm to control the swarm. Performance of the method and scalation are demonstrated in Section V through simulation examples.

II. Problem Statement and Assumptions

Consider n robots operating in a dynamic planar convex region, denoted by $Q(t) \subset \mathbb{R}^2$, with $t \in \mathbb{R}$. Robots sense with better quality the area nearby, so they should keep evenly deployed over the area for a good coverage of the region, while this region moves and changes.

In this paper, we make the following assumptions:

1) The region is a polygon inscribed within an ellipse. (The problem could be extended, and approached similarly, to more general regions, and also to the 3-D case.)
2) Each robot can obtain information from its current neighbors, within its perception/communication radius r, so the communication graph is undirected and time-varying. The communication radius r is enough for ensuring connectivity when the robots are uniformly distributed within the region. In the worst case, the number of communication links between neighbors, or hops, between the most distant robots is in the order of $\sqrt{n}$. With r or n larger than necessary, the region would be overpopulated with robots, and less hops would be required.
3) Direct communication between robots, within radius r, is assumed to be instantaneous. In other words, the dominant information delays (by far) are those caused by the flooding mechanism, due to the number of communication hops. Other delays are neglected.
4) Each robot knows its own pose in a global reference.
5) Robots are equipped with a suitable lower level local control, in which we are not interested here, so they are able to reach their successive targets specified by the high level control steps.
6) Although neither communication nor control actions over robots need to be synchronized, for simplicity they are considered to take place at discrete high level control steps of duration T. Hence, time can be measured in steps. We denote $Q_k = Q(kT)$ in discrete-time. The robot positions at step k are denoted by $p_{i,k}$, for $i = 1, \ldots, n$.

III. Basic Deployment Algorithm

Let us begin with a brief description of the coverage problem, and the basic solution strategy based on CVT computation plus feedforward action.

A. Centroidal Voronoi Tessellations

For a good coverage, the goal is to deploy robots evenly over the region Q_k. The ideal would be to partition Q_k into n disjoint cells $W_k = (W_{1,k}, \ldots, W_{n,k})$, and to place the robots in positions $P_k = (p_{1,k}, \ldots, p_{n,k})$ at the centroids of these cells, such that they form a CVT, defined as follows:

Definition 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 cells $W = (W_1, \ldots, W_n)$ with n robots in positions $P = (p_1, \ldots, p_n)$ within those cells.
2) A *Voronoi Tessellation* 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 Tessellation (CVT)* is a Voronoi Tessellation with each p_i at the centroid of W_i, see, e.g., [12].

B. Movements/Transformations

Although the movement of the dynamic region can be more general in practice, for some theoretical results in [14] to hold, it is required that the successive Q_k are related by affine transformations [18, Ch. 12], denoted as follows:

Definition 2: 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.) The resulting region from applying $\mathcal{A}$ to every point in Q, that is, Q after movement $\mathcal{A}$, is denoted by $Q \xrightarrow{\mathcal{A}} \mathcal{A}(Q)$. Successive movements are represented by the composition of transformations: $Q_k = \mathcal{A}_k \circ \mathcal{A}_{k-1} \circ \cdots \circ \mathcal{A}_1(Q_0)$.

In general, it cannot be assumed that robots know the "true" pose of the region at any time. Instead, they manage their own estimations of the region from the information they have. This is denoted and computed as follows:

Definition 3: Consider $Q_k \subset \mathbb{R}^2$ and $\mathcal{A}_k : \mathbb{R}^2 \to \mathbb{R}^2$, respectively a region and a movement at step k, and n robots. The estimation of the region Q_k, or the movement $\mathcal{A}_k$, by robot $i \in \{1, \ldots, n\}$ is denoted by $\widehat{Q}^i_k$, or $\widehat{\mathcal{A}}^i_k$, respectively.

Robot i uses the most recent information available about the region. If this information was h_i steps old, robot i would know up to Q_{k-h_i}, from which $\widehat{\mathcal{A}}^i_{k-h_i}$ could be estimated. Then, robot i would presume $\widehat{\mathcal{A}}^i_{k'} = \widehat{\mathcal{A}}^i_{k-h_i}$, for every $k' > k - h_i$, and $\widehat{Q}^i_k = (\widehat{\mathcal{A}}^i_{k-h_i})^{h_i}(Q_{k-h_i})$, where $(\widehat{\mathcal{A}}^i_{k-h_i})^{h_i}$ is the composition of h_i times $\widehat{\mathcal{A}}^i_{k-h_i}$.

C. Basic Algorithm: Dynamic Distributed CVT

The algorithm proposed in [14] to deploy the robots in a dynamic region is based on the distributed version of the Lloyd's method for a static region [13]. Starting at a configuration (W_0, P_0), in a region Q_0, which moves according to successive $Q_{k-1} \xrightarrow{\mathcal{A}_k} Q_k$, at each step k, with robots at P_k, each robot i executes Alg. 1. When a robot receives the information on the region to cover through the network, at each step k, it has only information up to $k - h_i$, where h_i is the number of hops that the information traveled.

Each cell $W_{i,k}$ is computed in line 11 of Alg. 1 as the intersection between: (a) the boundaries of $\widehat{Q}^i_k$; (b) a circle, or approximate polygon, with center $p_{i,k}$ and radius $r/2$; and (c) its Voronoi cell $V_{i,k}$ based on the (known) positions of the robots up to distance r,

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

which is a relatively straightforward geometric problem, see, e.g., [13]. The Voronoi cell can be shrunk to prevent collisions between large robots moving within neighbor cells, see, e.g., [19].

The next robot position $p_{i,k+1}$ computed in line 15 of Alg. 1 consists of two terms: Its centroid ("feedback action") plus the next estimated displacement ("feedforward action").

Note that the computational and memory costs associated to the operations performed by this algorithm, within each robot at each iteration, are very light. They are actually linear with the number of robots placed within a distance r, which are very few (at most about six [20]) in non overpopulated swarms. This is one of the facts that makes distributed strategies such as this one so appealing.

In [14, Th. 3.6] it is proven that, thanks to the feedforward action, the swarm controlled by this algorithm maintains a near-CVT, for movements consisting of an adequate number of intermediate steps. It is always possible to slow down the region motion by oversampling a given trajectory, including more steps. However, it was not clear a priori the relation between the region movement dynamics versus the required number of steps that allows the swarm to *successfully track* the region. Moreover, the difficulty to track a practical movement is varying: a region that translates at uniform speed can be trivially tracked by using a common constant feedforward action. However, when the region movement changes abruptly, e.g., in a sharp turn, it may be required to slow down so that robots have enough time to rearrange. Although this was discussed, nothing was said about whether, and how, robots could compute automatically this speed. This is the problem that we tackle in this paper: we want to automatically control the speed of the movement, so that the swarm can autonomously track the moving region successfully. Essentially, it is required that the swarm feeds back some information about its current difficulties to follow the region, and that the velocity is adapted consequently. This is the main contribution of this paper, which is addressed in the next section.

Algorithm 1: Dynamic Distributed CVT Algorithm

Data: Current robot position: $p_{i,k}$
(Some) Previous known poses of the region
Result: Next robot position: $p_{i,k+1}$
Distance to nearest neighbor: n_i
Distance traveled: d_i
Effort to stay in region: e_i
1 Get positions $p_{j,k} \ni ||p_{j,k} - p_{i,k}|| \leq r$ (neighbors)
2 $n_i \leftarrow \min(||p_{j,k} - p_{i,k}||)$
3 Update, from neighbors, the most recent previous pose of the region Q_{k-h_i}
4 Estimate $\widehat{\mathcal{A}}^i_{k-h_i}$ from Q_{k-h_i} and previous poses
5 // The subsequent, unknown, movements are presumed to be similar (Def. 3). Note that when $h_i = 0$ then $\widehat{Q}^i_k = Q_k$.
6 $\widehat{Q}^i_k = (\widehat{\mathcal{A}}_{k-h_i})^{h_i}(Q_{k-h_i})$; $\widehat{\mathcal{A}}_{k+1} = \widehat{\mathcal{A}}_{k-h_i}$
7 **if** $p_{i,k} \notin \widehat{Q}^i_k$ **then**
8 $p_{i,k} \leftarrow$ A close position at the boundary of $\widehat{Q}^i_k$
9 $e_i \leftarrow$ Effort to stay in the region (the distance from the former to the new $p_{i,k}$)
10 **end**
11 Compute $W_{i,k}$ [13]
12 // Compute mass (area) and centroid of $W_{i,k}$ [2]:
13 $M_{W_{i,k}} = \int_{W_{i,k}} dq$
14 $C_{W_{i,k}} = M^{-1}_{W_{i,k}} \int_{W_{i,k}} q dq$
15 Move to $p_{i,k+1} = C_{W_{i,k}} + \widehat{\mathcal{A}}^i_{k+1}(C_{W_{i,k}})$
16 $d_i \leftarrow ||p_{i,k+1} - p_{i,k}||$

IV. Feedback Control to Prevent Overspeed

From a process control viewpoint, we consider the swarm as a plant to be controlled. Given the complexity of modeling

this plant for a practical distributed real-time control purpose, this calls for the application of feedback controllers, which do not rely on an accurate plant model. PID control has been the most relevant approach to successfully control such systems in industry for decades [15], [16].

The first step to apply feedback control in any system is to measure the deviation of its behavior with respect to the one desired. Our objective is to move as fast as possible while succesfully tracking the region. In Subsect. IV-A, we define a measure of overspeed, that should be kept small or null.

Once the controlled variable, overspeed, is defined, a control law is applied to decide a good value for the control action. The control action is chosen to be a virtual force (Subsect. IV-C), which induces a change in velocity (Subsect. IV-B), inversely proportional to the size of the swarm. The designed feedback control law is explained in Subsect. IV-D. Finally, the control algorithm for the swarm robots is given in Subsect. IV-E.

A. Overspeed

A most critical issue when conceiving a feedback control system is to select the appropriate controlled variable, which must be informative in accordance with the final purpose of the system. For the feedback control of physical magnitudes, it is usually a matter of selecting the appropriate sensors and signal processing. In our case, though, it is a matter of finding an appropriate definition of overspeed, which must be: 1) Precise and sensitive, meaning that it differs from zero if, and only if, the current velocity is actually too high, and it becomes gradually greater when the velocity becomes increasingly dangerous; 2) Simple to implement, taking into account that it must be computed by the distributed network of simple robots; and 3) Simple to interpret, so that a trace of its causes can be easily understood, particularly during the tuning and testing of the control.

Recall that we consider that robots successfully track the region if the network stays continuously connected, agents do not collide, and they are deployed so that the coverage of the region is high. An intuitive promising candidate for keeping the network connected would be the algebraic connectivity [21], [22], which can be computed by a decentralized algorithm [23]. However, this option was discarded, since preliminary experiments showed that even the centralized version of the algebraic connectivity may drop to zero from a non-alarming value in one single step. In our case, we propose using instead $e_{\max}$, which informs of the increasing difficulties for some robots to stay in the region, which is related to connectedness: When a robot estimates the region with the currently available information, it could be the case that the robot finds itself outside of such region, what indicates it is falling behind of the swarm, risking connectivity. Another good indicator of increasing difficulties for the swarm to keep a near-CVT formation is $n_{\min}$, related to collision avoidance: When the minimum distance to the closest neighbor approaches zero, the corresponding robots are risking collision. These situations are quantified by saturated linear functions, from 0 to 1, which start to be positive at the warning value of the corresponding variable, and which reach unity at the alarming value. Overspeed, o, from 0 to 1, is defined as the saturated sum of these functions:

Definition 4: The following swarm performance indices are defined, at each step k of the algorithm:
1) $n_{\min} = \min\{n_i\}$, where n_i is the minimum distance between robot i and a neighbor.
2) $e_{\max} = \max\{e_i\}$, where e_i is the distance robot i moved to re-enter its estimated region, $\widehat{Q}_k^i$ (Alg. 1, line 9).

Overspeed, $o \in [0, 1]$, is defined as follows:

$$o = \text{sat}\left(\text{sat}\left(0 \leq \frac{n_{\min}^{\downarrow} - n_{\min}}{n_{\min}^{\downarrow} - n_{\min}^{\Downarrow}} \leq 1\right) + \cdots + \text{sat}\left(0 \leq \frac{e_{\max} - e_{\max}^{\uparrow}}{e_{\max}^{\Uparrow} - e_{\max}^{\uparrow}} \leq 1\right) \leq 1\right) \quad (1)$$

The following parameter selection, used for the simulations in Sect. V, has been made with the Assumption 2 that the region is not overpopulated (Sect. II):

Variable	Warning Value	Alarming Value
$n_{\min}$	$< 0.2r = n_{\min}^{\downarrow}$	$< 0.1r = n_{\min}^{\Downarrow}$
$e_{\max}$	$> 0.1r = e_{\max}^{\uparrow}$	$> 0.2r = e_{\max}^{\Uparrow}$

Our selection of warning and alarming distances between neighbor robots, $n_{\min}^{\downarrow} = 0.2r$ and $n_{\min}^{\Downarrow} = 0.1r$, respectively, is prudent, and it makes good sense, taking into account that the usual distance between neighbor robots in a non overpopulated region would be about $0.5r$ to r, and, with a typical value of the perception ratio of $100\,m$ and a robot safety diameter of $0.5\,m$, the safety distance would be $0.005r$.

B. Velocity and Limit Velocity

The feedback mechanism based on overspeed measurement shall compute an action that induces the appropriate changes in the velocity, to adjust it. Notice that such velocity refers to the change of the region between successive *steps*, that is, the change from a Q_{k-1} to Q_k. Let us start by quantifying precisely such velocity of change of pose.

A trajectory is a sequence of (a few) poses or the region, or *waypoints*, that must be passed through: $Q_0, Q_1, \cdots, Q_w, Q_{w+1}, \cdots Q_f$. Waypoints are not too close, so the region advances (typically much) less than one waypoint per step. If, for instance, it takes 100 steps to advance from Q_w to Q_{w+1}, then the velocity is $v_k = 0.01$ between two successive intermediate poses Q_{k-1} and Q_k, in the direction from Q_w towards Q_{w+1}. At any time, the next pose of the region is obtained interpolating between the previous and next waypoints. The following definition summarizes these ideas, introducing a convenient notation:

Definition 5: Given a region $Q \subset \mathbb{R}^2$ and an affine transformation $\mathcal{A} : \mathbb{R}^2 \to \mathbb{R}^2$, to simplify the notation, obviating the involved transformations, if $Q' = \mathcal{A}(Q)$, then $Q' - Q$ reads $\mathcal{A}$. Let $\mathcal{A}$ be a movement between some Q_a and Q_b, and let $\alpha\mathcal{A}$ be a fraction of such movement, with $\alpha \in [0, 1]$. Then, if $Q_k = \alpha\mathcal{A}(Q_{k-1})$ this is denoted by $Q_k - Q_{k-1} = \alpha(Q_b - Q_a)$, or $Q_k = Q_{k-1} + \alpha(Q_b - Q_a)$.

Given a complete trajectory to follow, $Q_0, Q_1, \cdots, Q_w, Q_{w+1}, \cdots Q_f$, let Q_{k-1} be a pose between Q_w and Q_{w+1}, then:

$$Q_k = Q_{k-1} + v_k(Q_{w+1} - Q_w). \tag{2}$$

where $v_k \in [0, 1]$ is the velocity between Q_{k-1} and Q_k.

Clearly, the region should not move so fast that some of its points move anything near one perception radius r in one step. This motivates the definition of a limit velocity, v_{lim}:

Definition 6: Given two regions $Q, Q' \subset \mathbb{R}^2$ such that $Q' = \mathcal{A}(Q)$, their distance is defined to be the maximum Euclidean distance between two corresponding points:

$$\|Q' - Q\| = \max_{q \in Q} \{\|\mathcal{A}(q) - q\|\} \tag{3}$$

The limit velocity to move in the direction from one waypoint, Q_w, towards the next, Q_{w+1}, is

$$v_{\text{lim}} = \eta \frac{r}{\|Q_{w+1} - Q_w\|} \tag{4}$$

where $\eta \in (0, 1]$ is a *prudency parameter*.

First of all, notice that the above notions of velocity and limit velocity are relative to the distance between waypoints. The absolute velocity is the displacement in one control step divided by the control step duration. This velocity is tightly related to η. With $r = 100\,m$ and $T = 1\,s$, the absolute velocity of a robot recklessly moving $0.6r$, approaching the border of its perception area, in one step would be $60\,m/s$ ($216\,km/h$), but, even with such an imprudent $\eta = 0.6$, if waypoints were 10 *km* away from each other, the limit velocity v_{lim} would be as low as 0.006 (0.6% of the distance between the waypoints in one step). Note that η represents the fraction of r that the region is allowed to move in one step at v_{lim}, so v_{lim} can be interpreted as the maximum absolute velocity of the region (in r/T).

Concerning the tuning of the prudency parameter, we have found that a good choice is between $\eta = 0.2$ and 0.4. This means that the maximum displacement of the region in one control step should be between 20% and 40% of r. Notice that robots would move at this pace when they are keeping the near-CVT formation, but some might have to move faster when they have to rearrange, so, with higher values of η, some robot might have to move near r, which would not be admissible in practice.

C. Lead and Drag Forces. Density.

We propose to use a virtual force to induce the velocity changes according to Newton's second law, using a virtual mass proportional to the swarm size: $\delta \cdot n$, where δ is a virtual density parameter. The value used for the simulations in Sect. V is $\delta = 1$, which is conservative (with less density the control action is more aggressive). The virtual force has two components, a constant lead force, F_L, and a drag force F_D that depends on the overspeed. For normalization, F_D ranges from 0 to 1. Mimicking actual vehicles, where their ability to brake should be larger than their ability to accelerate, a reasonable choice, used for the simulations in Sect. V, is $F_L = 0.25$, so the net force ranges from -0.75 to 0.25. The combined effect of the force and the limit velocity might be a larger or smaller number of steps from one waypoint to the next one, affecting the total time required to fulfill the mission.

D. Feedback Control Law

We want the control to be simple and robust, and easy to tune. We propose a proportional-plus-derivative (PD) control, with proportional gain K_p and derivative time constant T_d, that manipulates the virtual force, $F_L - F_D$, that drives the movement. The integral mode of control is not required, because once the overspeed, o, is small or zero there is a positive (lead) force, F_L, that increases the velocity up to its maximum prudent value, given by the limit velocity, v_{lim}. The inertial behavior, due to the virtual mass $\delta \cdot n$ of the swarm, smooths the changes in the velocity, v, although changes in the drag force, F_D, are often abrupt.

The difference equations, including the saturation limits in F_D and v, are the following:

$$F_{Dk} = \text{sat}\left(0 \le K_p\left(o_k + T_d\left(o_k - o_{k-1}\right)\right) \le 1\right) \tag{5}$$

$$v_k = \text{sat}\left(0 \le v_{k-1} + \frac{F_L - F_{Dk}}{\delta \cdot n} \le v_{\text{lim}}\right) \tag{6}$$

In Sect. V we adjusted $K_p = 1.2F_L$, and $T_d = \sqrt{n}/5$.

E. Algorithm for Automatic Region Movement

Algorithm 2: Automatic region movement

Data: Current robot position: $p_{i,k}$
Current estimated region pose: $\widehat{Q_k^i}$
Most recent n_j, d_j, and e_j of every robot $j \neq i$
Local values of o_k and v_k
Result: Next target position: $p_{i,k+1}$
Next target region pose: $\widehat{Q_{k+1}^i}$

1 $p_{i,k+1}$, n_i, d_i, $e_i \leftarrow$ Alg. 1 (Note that $h_i = 0$.)
2 Update, through neighbors, the values of n_i, d_i, e_i from $j \neq i$
3 Promediate o_k and v_k with the mean values updated from neighbors (dynamic average consensus [24])
4 Compute $n_{\min}$, $e_{\max}$, $o_{k+1} \leftarrow$ Def. 4
5 Compute v_{lim}, F_D, $v_{k+1} \leftarrow$ (4), (5), (6)
6 Estimate $\widehat{Q_{k+1}^i} = \widehat{Q_k^i} + v_{k+1}(Q_{w+1} - Q_w) \leftarrow$ (2)
7 // where Q_w and Q_{w+1} are the previous and next waypoints

It is assumed that, at any time, every robot knows the next waypoint. In the case of a fully autonomous mission, every robot would know all the waypoints from the start. In the case of a teleoperated mission, it suffices that the following waypoint is sent to one or some of the robots well before the swarm reaches the next waypoint, so it can be communicated to the others in time, through the network. In any case, the next pose of the region is *not* communicated through the network, but it is computed by each robot i, at each step k, and consensuated with the others, applying Alg. 2.

Note that the proposed method is distributed: For the CVT and feedforward computation (Alg. 1), and for the overspeed regulation and velocity consensus, only data from the robot and from its one-hop neighbors are required, although part of these data refers to more distant robots.

V. Simulation Examples

The behavior of the method will be analyzed by simulation examples of a case adapted from [3]. A polygonal region (defined by 16 vertices, inscribed in an ellipse) translates, rotates, and changes its shape and size to pass through a narrow corridor. A tenth of the robots are removed suddenly by the middle of the simulation, little later than the sharp change of direction, to simulate a massive failure, a most demanding severe perturbation happening in the worst moment. The perception and communication adimensional radius r is chosen to be $r = 3\sqrt{A/(n\pi)}$ so that the region is not overpopulated with $A = 1$ being the largest area to cover[1]. Assuming that the communication radius is always the same, e.g., $100\,m$, with (adimensional) $r = 0.24\,(0.14)$ for $n = 50\,(150)$ robots, respectively, the scale is $399\,(691)\,m$, in the snapshots of Figs. 1 and 2.

Figs. 1 and 2 show snapshots of the swarm in a few cases, which are also included in the video attachment, in motion. They are accompanied by plots with the evolution of the following variables, related to the quality of the area coverage, the controlled velocity, and overspeed:

- Coverage ratio, the fraction of the area of the current region covered by at least one robot (black). The purpose of the mission is to keep the region covered, so this is an important performance index.
- Maximum distance d_i (scaled by r) traveled by a robot in each step (blue), and distance traveled by the region in each step (dotted bue). They differ when the robots have to rearrange to keep the near-CVT formation. These distances can be translated to actual velocities, for given values of r and T. For instance, a 0.4 value, for technologically feasible $r = 100\,m$ and $T = 1\,s$, corresponds to $40\,m/s$ ($144\,km/h$).
- The two magnitudes taken into account to compute the overspeed: the minimum distance n_i to the nearest neighbor (red), and the maximum distance e_i traveled to re-enter the region (magenta), both scaled by r. When their values are too low, respectively too high, they trigger the mechanism based on PD, force, and inertia to change the actual velocity of the region (dotted blue).

In Fig. 3, the results of these cases, and many others, are summarized. There are simulations with different number of agents (n, from 10 to 300, in the horizontal axis), and different methods: plain Lloyd's method (red), our previous work [14] (magenta), and the method proposed here, with three different settings of the prudency parameter, η, limiting the velocity, to 0.2 (prudent, green), 0.4 (medium, cyan), or 0.6 (reckless, blue). The thick plots indicate the mean approximate coverage ratio, and the dotted plots are one standard deviation above and below. The boxed integer numbers indicate the number of steps required to complete the mission, which starts once the robots are deployed as a CVT formation over the initial region, and ends when they are deployed as a CVT formation over the final region. For the previous methods, the displayed figures correspond to the fastest trials that succeeded, with a constant velocity that had to be guessed by trial-and-error (faster cases failed). For the proposed method, all the cases succeeded, even with the extreme $\eta = 0.6$. In fact, $\eta = 0.6$ is too reckless to be used in practice, because, while rearranging, some robots would be required to move about their perception ratio in one step.

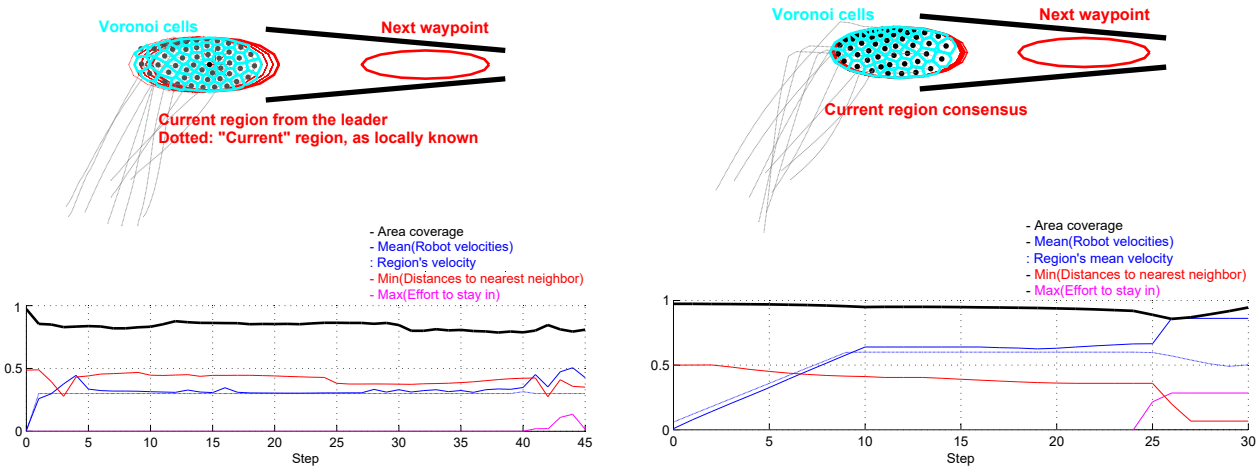
With the basic Lloyd's method, which works well with static regions [2], the region movement needs to be extremely slow for the mission to succeed, even if the region pose is broadcasted to every robot. In Fig. 3 some cases with up to 50 robots are reported. For instance, with 50 robots, 1290 steps were required. In the video attachment, a failure case, when the speed is not low enough, is shown. (We considered needless to try this method with more robots.)

For the feedforward, both in our previous and current methods, robots estimate the region movement from the last information about its translation and rotation, that is, changes in size and shape are ignored, so they can be regarded as unmodeled disturbances. Higher order estimations could be used, and future poses could be taken into account, when they were already known, but this is not done, for simplicity, and to illustrate robustness against some uncertainty.

In Fig. 1a, a snapshot after 45 steps of the method in [14] is shown at the highest constant speed for which the mission ends successfully. The current region position is flooded through the network from a central leader, so the outermost robots try to cover a region that is some steps outdated (shown dotted), and therefore they are falling behind, in spite of the feedforward action. In Fig. 1b, a snapshot after 30 steps of the proposed method with $\eta = 0.6$ is shown. Note that, at this step, the swarm is slowing down the movement (dotted blue) because both $e_{\max}$ (magenta) and $n_{\min}$ (red) have alarming values, signalling overspeed. While the dynamic consensus is being reached, each robot has slightly different versions of the region to cover, but nevertheless the area coverage ratio of the "true" (mean) region, yet unknown/consensuated by the robots, is still very good. Note that the mission is accomplished faster, not only better: in the end, it results in 71 vs. 89 steps, with a mean coverage ratio of 95% vs. 83% (see Fig. 3, for $n = 50$).

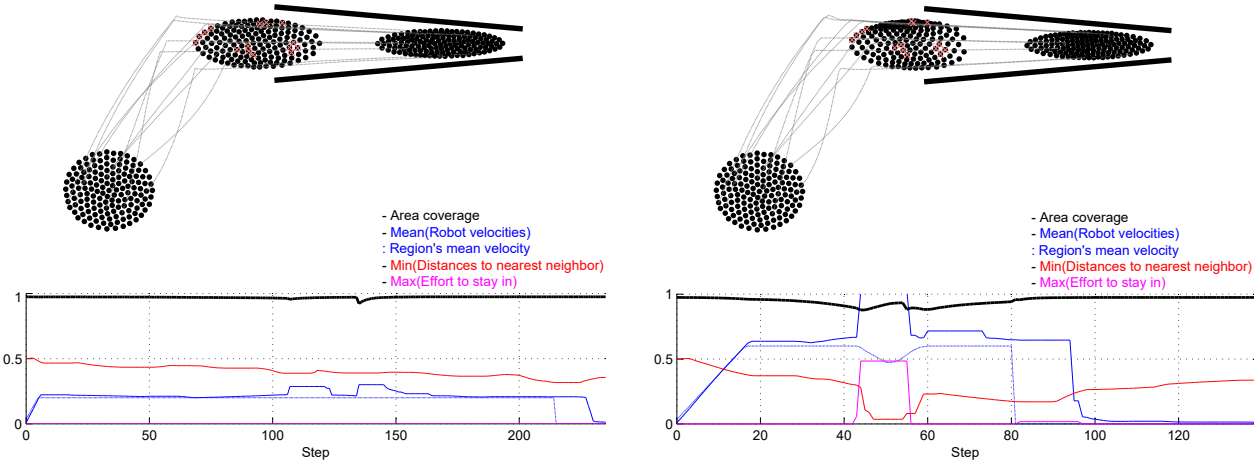
To demonstrate the good scalation of the method, two cases with $n = 150$ initial robots are shown, with $\eta = 0.2$ and 0.6, in Fig. 2a and 2b, respectively. Notice that when the reason for a larger swarm is to cover a wider area with the same kind of robots, then it is also the case that distances are greater (they are roughly proportional to $\sqrt{n}$), hence, with the same velocities, the time to complete the mission (the number of steps) grows. Three snapshots are shown in the scenes: the initial and final ones, and one at the step when 15 robots are forced to disappear (they are marked with

[1]To satisfy Assumption 2 (Sect. II): Notice that $A \approx n\pi(r_{\min}/2)^2$, when all robots are separated $r_{\min}$.



(a) $n = 50$, *Lloyd+FF, constant speed [14]* (b) $n = 50$, $\eta = 0.6$

Fig. 1. Rather than receiving the pose of the region to cover at each step from (or through) some leaders, it is more practical regulating plus consensuating the velocity between waypoints known in advance. This also leads to a better coverage ratio even when the prudency parameter η is adjusted to a reckless value.



(a) $n = 150$, $\eta = 0.2$ (b) $n = 150$, $\eta = 0.6$

Fig. 2. With many more robots, to cover a larger area, the behavior is similar, although it naturally takes more time to complete the mission, since distances are greater. The parameter η allows to priorize either coverage quality and orderly movement or speed.

red $\times$). It is apparent, here and in Fig. 3, that lower, more prudent, values of η result consistently in a better quality of the coverage, practically independent of the swarm size. The more ordered evolution of the swarm comes at the expense of a longer time to complete the mission.

The behavior is better appreciated in the video attachment, where, for a final practical demonstration, an additional simulation experiment is shown: Two rooms connected by a corridor have to be explored by a team of robots uniformly deployed over a square, then a diamond, and finally a circle. (The last transformation of the region is not affine, a restriction that is only needed for some theoretical results to hold.) In this experiment, which serves as a validation and illustration, all the parameters are the same as in all the other simulations, with $\eta = 0.4$.

VI. Conclusion

A practical method to cover uniformly a dynamic region has been proposed. This method is based on Centroidal Voronoi tessellations, as a method to deploy the robots uniformly, which works perfectly to cover static regions with a sort of flexible formation, and which is well adapted to swarms. To cope with movements, deformations, and other disturbances, simple control principles are applied. The set up used in the simulation experiments was somehow ideal, to concentrate on the basic properties of the method. Once it has demonstrated to work well and to be scalable, even in difficult situations, with reckless speeds, we plan to investigate its performance and robustness further, in more realistic simulations of the perception and control of the agents, with relative positions, uncertainties, communication latencies and random losses, and asynchronicity, or in real experiments.

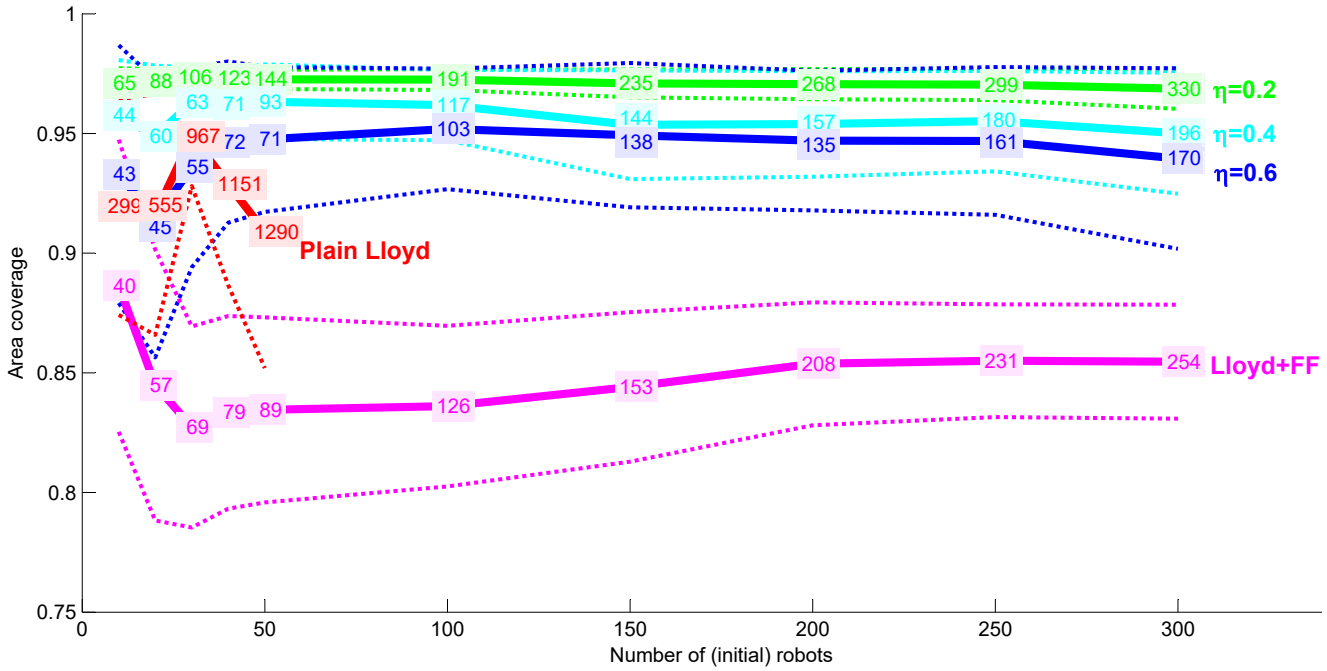


Fig. 3. Summary of results. The new proposed method clearly outperforms the existing alternatives to cover a dynamic region with a near-CVT formation. Thanks to the distributed nature of the algorithm, it scales well. The prudency parameter η allows to priorize order and quality or speed. The fact that the method succeeds even with such a reckless adjustment as $\eta = 0.6$ is a good indication of the robustness of the method.

REFERENCES

[1] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[2] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage Control for Mobile Sensing Networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[3] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 865–875, 2004.

[4] C. C. Cheah, S. P. Hou, and J. J. Slotine, "Region following formation control for multi-robot systems," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 3796–3801.

[5] S. P. Hou, C. C. Cheah, and J. J. Slotine, "Dynamic region following formation control for a swarm of robots," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 1929–1934.

[6] C. C. Cheah, S. P. Hou, and J. J. E. Slotine, "Region-based shape control for a swarm of robots," *Automatica*, vol. 45, no. 10, pp. 2406 – 2411, 2009.

[7] 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*, vol. 30, no. 3, pp. 550–565, 2014.

[8] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human Interaction with Robot Swarms: A Survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.

[9] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[10] F. Abbasi, A. Mesbahi, and J. Mohammadpour Velni, "A new voronoi-based blanket coverage control method for moving sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 409–417, 2019.

[11] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

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

[13] M. Cao and C. Hadjicostis, "Distributed algorithms for Voronoi diagrams and application in ad-hoc networks," Tech. Rep., 2003. [Online]. Available: http://www.eng.ucy.ac.cy/hadjicostis/JournalPapers/voronoi_1.pdf

[14] E. Teruel, R. Aragues, and G. López-Nicolás, "A distributed robot swarm control for dynamic region coverage," *Robotics and Autonomous Systems*, vol. 119, pp. 51 – 63, 2019.

[15] K. J. Astrom and T. Hägglund, *Advanced PID control*. IEEE Control Systems, 2006.

[16] Smith, C.A. and A. Corripio, *Principles and Practice of Automatic Process Control*. Wiley, 2005.

[17] Y. Song, B. Wang, Z. Shi, K. R. Pattipati, and S. Gupta, "Distributed algorithms for energy-efficient even self-deployment in mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1035–1047, 2014.

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

[19] Y. Kantaros and M. M. Zavlanos, "Distributed communication-aware coverage control by mobile sensor networks," *Automatica*, vol. 63, pp. 209–220, 2016.

[20] L. Wang, F. Hétroy-Wheeler, and E. Boyer, "A Hierarchical Approach for Regular Centroidal Voronoi Tessellations," *Computer Graphics Forum*, vol. 35, no. 1, pp. 152–165, 2016.

[21] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390 – 396, 2010.

[22] P. R. Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff, "A passivity-based decentralized strategy for generalized connectivity maintenance," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 299–323, 2013.

[23] E. Montijano, J. I. Montijano, and C. Sagues, "Fast distributed algebraic connectivity estimation in large scale networks," *Journal of the Franklin Institute*, vol. 354, no. 13, pp. 5421 – 5442, 2017.

[24] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms," *IEEE Control Systems*, vol. 39, no. 3, pp. 40–72, 2019.