# Monitoring Environmental Boundaries with a Robotic Sensor Network

Sara Susca Sonia Martínez Francesco Bullo

***Abstract*—In this paper we propose and analyze an algorithm to monitor an environmental boundary with mobile agents. The objective is to optimally approximate the boundary with a polygon. The mobile sensors rely only on sensed local information to position some interpolation points and define an approximating polygon. We design an algorithm that distributes the vertices of the approximating polygon uniformly along the boundary. The notion of uniform placement relies on a metric inspired by approximation theory for convex bodies. The algorithm is provably convergent for static boundaries and efficient for slowly-moving boundaries because of certain input-to-state stability properties.**

## I. Introduction

Much recent attention has been given to the problem of boundary estimation and tracking by means of robotic networks. The common goal is to design a distributed algorithm that allows a limited number of mobile agents to detect the boundary of a region of interest and estimate it as it evolves. Boundary estimation and tracking is useful in numerous applications such as the detection of harmful algae bloom [1], [2], oil spill [3], and fire containment [4]. In [1], Marthaler and Bertozzi adopt the so-called "snake algorithm" (from the computer vision literature) to detect and track the boundary of harmful algae bloom. Each agent is equipped with a chemical sensor that is able to measure the concentration gradient and with a communication system that is able to exchange information with a data fusion center. In [2], Bertozzi *et al.* suggest an algorithm that requires only a concentration sensor: the agents repeatedly cross the region boundary using a bang-bang angular velocity controller. In [3], Clark and Fierro use a random coverage controller, a collision avoidance controller and a bang-bang angular velocity controller to detect and surround an oil spill. In [4], Casbeer *et al.* describe an algorithm that allows Low Altitude Short Endurance Unmanned Vehicles (LASEUVs) to closely monitor the boundary of a fire. Each of the LASEUVs has an infrared camera and a short range communication device to exchange information with other agents and to download the information collected onto the base station. A different approach is considered by Zhang and Leonard in [5]. A formation of four robots tracks at unitary speed the level sets of a field. Their relative position changes so that they optimally measure the gradient and they estimate the curvature of the field in the center of the formation. Challenges in boundary estimation using motion-enabled sensors are discussed in [6].

In this paper we propose an algorithm to estimate and reconstruct the boundary of a region. The objective is for a group of mobile agents to optimally place some interpolation points on the boundary of a simply connected planar region. The boundary is then reconstructed by linear interpolation of the interpolation points. We assume that (i) at initial time the agents have an estimate of the boundary, (ii) each agent is equipped with a limited-footprint camera-like sensor and with algorithms to locally estimate the tangent and curvature of the boundary, and (iii) the agents exchange information through a ring-topology communication network. An example scenario for these assumptions is a situation where a group of Unmanned Air Vehicles (UAVs) with an on-board camera are tasked to reconstruct the boundary of an oil spill or of a forest fire.

The contribution of this paper is twofold. First, we propose a criterion to optimally place interpolation points to reconstruct a planar boundary. The criterion requires that the interpolation points are uniformly distributed according to a curvature-weighted distance function defined along the boundary; this function is inspired by the literature on optimal approximation of convex bodies, e.g., see the survey by Gruber [7]. Second, combining the optimal distribution for the interpolation points with a data structure generated by the mobile agents, we present one of the first provably convergent algorithms to reconstruct a planar boundary. The presented algorithm has the following two properties: (i) our algorithm is provably convergent for static boundaries and efficient for slowly-moving boundaries, (ii) our algorithms leads the interpolation points to a locally asymptotically optimal distribution along the boundary. The optimality is local along each convex arc of the boundary and asymptotic in the number of interpolation points. Our convergence analysis relies upon and extends known results from the theory of consensus algorithms; a necessarily incomplete list of references about this subject includes [8] and [9].

The paper is organized as follows. In Section II we review some mathematical literature on approximation theory. In Section III we introduce an algorithm to jointly update an environment boundary and deploy the agents uniformly along the boundary estimate. In Section IV we present our final concluding remarks.

## II. Approximation theory for convex bodies

In this section we review some known useful results from the literature on approximation of strictly convex bodies; e.g., see the extensive survey [7]. In the standard literature on

Sara Susca and Francesco Bullo are with the Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, Santa Barbara, CA 93106, sara@ece.ucsb.edu, bullo@engineering.ucsb.edu

Sonia Martínez is with the Department of Mechanical and Aerospace Engineering, University of California at San Diego, San Diego, CA 92093, soniamd@ucsd.edu

convex bodies approximations, the symmetric difference $\delta^S$ between two compact, and strictly convex bodies $C$, $B \in \mathbb{R}^d$ is defined by $\delta^S(C,B) = \mu(C \cup B) - \mu(C \cap B)$, where $\mu$ is the Lebesgue measure on $\mathbb{R}^d$. If $Q$ is the body to be approximated by an inscribed $n$-vertices polygon $P_n$, then $\delta^S(Q,P_n) = \mu(Q) - \mu(P_n)$. Let $\partial Q$ be the boundary of $Q$, $\ell$ be the arc length along $\partial Q$, and $\theta$ be the angular position in a polar variable parameterization of $\partial Q$. Let $\rho$ and $\kappa = \rho^{-1}$ be the curvature radius and curvature of the boundary, respectively. For $n$ sufficiently large, McLure and Vitale [10] show that $\delta^S(Q,P_n^*) \approx \frac{1}{12n^2}\left(\int_0^{2\pi}\rho(\theta)^{2/3}d\theta\right)^3 = \frac{1}{12n^2}\left(\int_{\partial Q}\kappa(\ell)^{1/3}d\ell\right)^3$, where $P_n^*$ is the best approximating polygon with $n$ vertices inscribed in $Q$. To construct the best approximating polygon $P_n^*$ for a strictly convex body McLure and Vitale in [10] suggest the *method of empirical distributions*. According to this method, the positions $\theta_i$, $i \in \{1,\dots,n\}$, of the vertices along $\partial Q$ have the property that $D^S(i) = \int_{\theta_i}^{\theta_{i+1}}\rho(\theta)^{2/3}d\theta$ has the same value for every consecutive pair of vertices $(i, i+1)$. (Here and in what follows, we adopt the convention that $n+1 = 1$.) Interpolating polygons computed according to the method of empirical distributions converge to $P_n^*$ as $n \to +\infty$.

For smooth nonconvex bodies with a finite number of inflection points, the method of empirical distributions will also yield a nearly optimal distribution as $n \to +\infty$ because of the local convexity of the body away from inflection points. We show how to do this in what follows. Since the curvature radius may be unbounded at some point of a nonconvex boundary, the integral $D^S(i)$ may be unbounded for some $i$. We avoid this problem by considering the following general notion of distance along a boundary. For $\lambda \in [0,1]$, we define the pseudo-distance $D_\lambda$ between vertices $(i, i+1)$ by:

$$D_\lambda(i) = \lambda\int_{\ell_i}^{\ell_{i+1}}\kappa(\ell)^{1/3}d\ell + (1-\lambda)(\ell_{i+1}-\ell_i).$$

This definition is inspired by the fact that, for convex bodies, we have $\int_0^{2\pi}\rho(\theta)^\alpha = \int_{\partial Q}\kappa(\ell)^{1-\alpha}d\ell$ for $\alpha > 0$, see [7]. Introducing the convex combination with arc length, we guarantee that $D_\lambda(i)$ is nonzero whenever the vertices $i$ and $i+1$ do not coincide. Note that changing the value of the parameter $\lambda$ has less noticeable impact in arcs of the boundary with high curvature and more in the arcs with low curvature. In what follows we develop a version of the method of empirical distributions in which consecutive vertices are uniformly distributed according to the pseudo-distance $D_\lambda$.

## III. Boundary estimation and agent pursuit algorithm

In this section we propose and analyze an algorithm that leads a group of $n_\mathrm{a}$ agents to compute and constantly update an estimate of a slowly moving boundary. The estimate is computed in the form of an interpolating polygon; the algorithm aims to place the interpolation points so that they are uniformly distributed according to the pseudo-distance $D_\lambda$ introduced in the previous section. As discussed in the Introduction, we assume that (i) at initial time the agents have an estimate of the boundary, (ii) each agent can locally estimate the tangent and curvature of the boundary, and (iii) the agents are able to exchange information according to a ring-topology communication service.

We let $\{P_i\}_{i\in\{1,\dots,n_\mathrm{a}\}}$ be the positions of the mobile agents and we let $\{p_\alpha\}_{\alpha\in\{1,\dots,n_\mathrm{ip}\}}$ be the vertices of the interpolating polygon; in a practical implementation, we assume that each agent maintains a copy of these virtual positions. Relying upon the initial estimate of the boundary, we make the following additional assumptions: at time $t = 0$, the agents have reached a point of $\partial Q$ and the interpolation points are distributed (possibly nonuniformly) on the estimated boundary. We assume that both the interpolation points and the agents are ordered counterclockwise, and that the agents move counterclockwise along the boundary with speed $v_i$, see Figure 1.

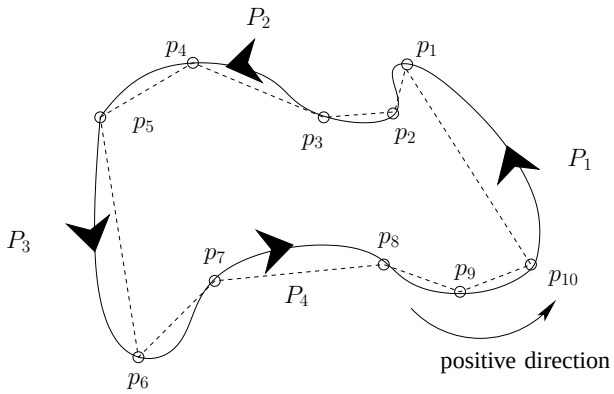


Fig. 1. In the figure the solid line is the boundary $\partial Q$, the triangles are the agents, the circles are the interpolation points, and the dotted line is the approximating polygon defined by the interpolation points.

The agents have two objectives: (i) update the interpolation points such that they are uniformly distributed along $\partial Q$ according to the estimated pseudo-distance $\widehat{D}_\lambda$, (ii) move along the boundary equally distributed according to arc length distance. To achieve these two objectives we propose a novel Estimate Update and Pursuit Algorithm that can be summarized as follows.

> Every agent moves counterclockwise along the time-varying $\partial Q$ and collects estimates of the curve $\partial Q$ and of its tangent and curvature. Using these estimates, the agent completes the following four actions: First, each agent updates the positions of the interpolation points so that they take value in $\partial Q$. In other words, as sufficient information is available, each interpolation point $p_\alpha$, $\alpha \in \{1,\dots,n_\mathrm{ip}\}$, is projected onto the measured boundary. Second, after an interpolation point $p_\alpha$ has been projected, the agent collects sufficient information so that it can locally optimize its position along the estimate of $\partial Q$. Third, every agent estimates the arc length distance between itself and its immediate clockwise and counterclockwise neighbors and uses this information to speed up or slow down. Fourth and last, the updated interpolation point $p_\alpha$ is transmitted to appropriate neighboring agents.

The first two steps have the combined effect of updating the local estimates of the boundary. The third step has the effect of distributing the agents uniformly along the boundary. The fourth step has the effect of maintaining correct distributed information about the boundary estimate.

### A. Algorithm description

In this section we present the ESTIMATE UPDATE AND PURSUIT ALGORITHM in some detail and we analyze its stability. We begin by introducing some basic geometric notions about curves and making some smoothness assumptions. In what follows, we let $\|v\|$ be the Euclidean norm of $v \in \mathbb{R}^n$, $\overline{\mathbb{R}}_+$ be the set of nonnegative real numbers, and $\mathbb{N}_0$ be the set of nonnegative integers. Let $\partial Q$ be the boundary of a simply connected, and possibly nonconvex set $Q$ in $\mathbb{R}^2$. Let $\gamma\colon \overline{\mathbb{R}}_+ \times [0,1] \to \mathbb{R}^2$ be a parametric representation of the time-varying boundary so that, at fixed $t \in \overline{\mathbb{R}}_+$ and for all $s \in [0,1]$, $\gamma(t,s)$ describes the boundary $\partial Q(t)$. We assume that $\frac{\partial \gamma(t,s)}{\partial s} = \gamma'(t,s) \neq 0$ for all $s \in [0,1]$ and for all $t$, that $\gamma(t,0) = \gamma(t,1)$, and that $s$ increases as we traverse the curve in the counterclockwise direction. We also assume that $\gamma(t,s)$ is smooth with respect to $s$ and $t$ and that the length of the boundary $\partial Q$ is upper and lower bounded uniformly in $t$. The curvature $\kappa\colon [0,1] \to \bar{\mathbb{R}}_+$ of the curve $\gamma$ is defined by $\kappa(s) = \frac{\|\gamma'(s) \times \gamma''(s)\|}{\|\gamma'(s)\|^3}$.

Now, we can begin our detailed description of our algorithm; we begin with the data structure. Each agent $i$ maintains the following the following variables in its memory.

- ***Variable #1:*** a counter NOW taking values in $\{1, \ldots, n_{\text{ip}}\}$, when necessary we will use $\text{NOW}^i$ to indicate the value of the counter NOW for agent $i$;
- ***Variable #2:*** a buffer BUFFERARC containing a collection of triplets $\{o_j, \widehat{\gamma}'(o_j), \widehat{\kappa}(o_j)\}$, where $o_j$ is a point on $\partial Q$, $\widehat{\gamma}'(o_j)$ and $\widehat{\kappa}(o_j)$ are tangent vector and curvature at the point $o_j$, respectively, and $j$ takes value in an index set $\{1, \ldots, n_o\}$. It is also convenient to let $\mathcal{O} = \{o_j\}_{j \in \{1,\ldots,n_o\}}$;
- ***Variable #3:*** a boundary estimate given by interpolation points $p_1, \ldots, p_{\text{ip}}$, tangent vectors at interpolation points $\gamma'_1, \ldots, \gamma'_{\text{ip}}$, and pairwise pseudo-distance between interpolation points $\widehat{D}_\lambda(p_\alpha, p_{\alpha+1})$, $\alpha \in \{1, \ldots, n_{\text{ip}}\}$.

These variables are initialized as follows: NOW is set equal to the index of the interpolation point that is immediately counterclockwise from $P_i(0)$, BUFFERARC is empty, and the boundary estimate is given by assumption.

*Remark 1 (Interpretation):* The positions $\mathcal{O}$ are points that an individual agent has recently visited while moving along $\partial Q$ and are an arbitrarily accurate discretization of a portion of $\partial Q$; these points reside in the memory of every individual agent. On the contrary, the interpolation points $p_1, \ldots, p_{\text{ip}}$ are a coarser discretization of a portion of $\partial Q$ and are communicated among agents. The idea is that the agent moves and gathers sufficient information to update the interpolation point $p_{\text{NOW}}$ with the set of observations in BUFFERARC, that is, to project $p_{\text{NOW}}$ onto the discretized representation BUFFERARC of $\partial Q$. •

Let us illustrate the meaning of the variables in Figure 2. The curve of points represents the approximation BUFFERARC of $\partial Q$ as seen by agent $i$, while the solid line represents $\partial Q$ as known through the interpolation points $p_1, \ldots, p_{\text{ip}}$ and the tangent vectors $\gamma'_1, \ldots, \gamma'_{\text{ip}}$ before any update takes place. The agent is represented by a triangle. The white circles are the interpolation points before the update, and the black circles represent the interpolation points after the update; the white arrows denote the projection of the interpolation points onto the recently measured boundary and the black arrow denotes the locally optimal repositioning of the interpolation points.

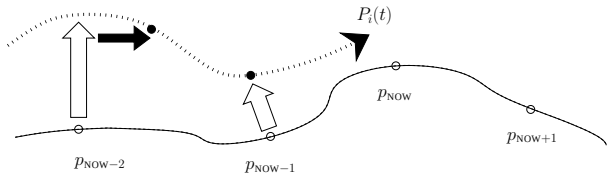


Fig. 2. Mobile agent moving along boundary, projecting (white arrow) and locally updating (black arrow) interpolation points.

In what follows, we need to provide rules to perform the various data management tasks:

- ***Rule #1:*** how to maintain the data in BUFFERARC, i.e., how long should the buffer be;
- ***Rule #2:*** when and how to project onto $\partial Q$ the next outstanding interpolation point $p_{\text{NOW}}$;
- ***Rule #3:*** when and how to locally optimize the updated interpolation point $p_{\text{NOW}-1}$; and
- ***Rule #4:*** when and what to communication and to whom.

*Rule #1:* If agent $i$ is in the process of projecting interpolation point $p_{\text{NOW}}$, then BUFFERARC must contain information about $\partial Q$ starting from interpolation point $p_{\text{NOW}-2}$ up to the agent position.

*Rule #2:* In most cases, the projection takes place when the agent crosses the line $\ell_{\text{NOW}}$ that passes through $p^-_{\text{NOW}}$ and is perpendicular to $\widehat{\gamma}'(p^-_{\text{NOW}})$. To be specific, $p^-_{\text{NOW}}$ denotes the interpolation point about to be updated, and $\widehat{\gamma}'(p^-_{\text{NOW}})$ denotes the corresponding tangent vector. We can therefore define $p^+_{\text{NOW}}$ to be the point where the mobile agent trajectory $P_i(t)$ crosses $\ell_{\text{NOW}}$, and $\widehat{\gamma}'(p^+_{\text{NOW}})$ to be the tangent to $\partial Q$ at $p^+_{\text{NOW}}$. This is indeed the correct definition if the agent does cross this $\ell_{\text{NOW}}$. This projection operation is illustrated in Figure 3.

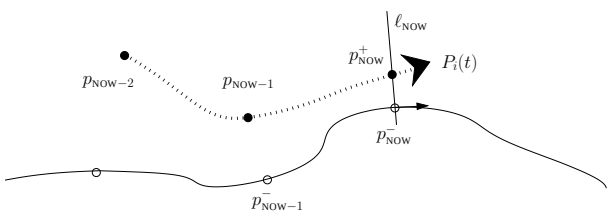


Fig. 3. Mobile agent projecting interpolation point onto the observed boundary

We therefore amend the algorithm to act as follows. If sufficient time has elapsed without the agent crossing $\ell_{\text{NOW}}$, e.g., if no crossing has happened at time $t$ such that $\widehat{D}_\lambda(p_{\text{NOW}-1}, P(t)) = 2\widehat{D}_\lambda(p^-_{\text{NOW}-1}, p^-_{\text{NOW}})$, then $p^+_{\text{NOW}}$ is set

equal to the point on $\mathcal{O}$ that is closest to $p^-_{\text{NOW}}$. The corresponding definition is also employed for $\widehat{\gamma}'(p^+_{\text{NOW}})$. In both cases, this projection is well defined and has the following properties. If $\partial Q$ is time-invariant, then $p^-_{\text{NOW}} = p^+_{\text{NOW}}$, if $\partial Q$ is slowly time-varying, then $p^+_{\text{NOW}}$ is close to the orthogonal projection of $p^-_{\text{NOW}}$ onto $\partial Q$.

*Rule #3:* The local optimization of $p_{\text{NOW}-1}$ takes place immediately after the update of $p_{\text{NOW}}$. Using the data in BUFFERARC, the agent computes the Voronoi cell inside $\mathcal{O}$ of the interpolation point $p_{\text{NOW}-1}$ and moves $p_{\text{NOW}-1}$ to the center of this cell. This operation is illustrated in Figure 4.

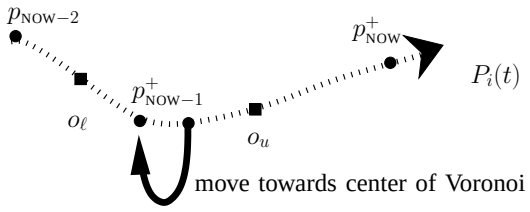


Fig. 4. Mobile agent locally optimizing interpolation point $p_{\text{NOW}-1}$ along the observed boundary, after projecting $p_{\text{NOW}}$

To describe this local optimization accurately, let us introduce some notation. The Voronoi cell $\{o_\ell, \ldots, o_u\} \subset \mathcal{O}$ of the interpolation point $p_{\text{NOW}-1}$ is defined implicitly by

$$\widehat{D}_\lambda(p_{\text{NOW}-2}, o_\ell) = \widehat{D}_\lambda(o_\ell, p_{\text{NOW}-1}) = \frac{\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})}{2},$$
$$\widehat{D}_\lambda(p_{\text{NOW}-1}, o_u) = \widehat{D}_\lambda(o_u, p^+_{\text{NOW}}) = \frac{\widehat{D}_\lambda(p_{\text{NOW}-1}, p^+_{\text{NOW}})}{2}.$$

In other words, the point $o_\ell$ is the midpoint between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$, while $o_u$ is the midpoint between $p_{\text{NOW}-1}$ and $p^+_{\text{NOW}}$ after the latter was projected on $\partial Q$. We now implicitly define the center $o_k \in \mathcal{O}$ of the Voronoi cell by

$$\widehat{D}_\lambda(o_\ell, o_k) = \widehat{D}_\lambda(o_k, o_u) = \frac{\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1}) + \widehat{D}_\lambda(p_{\text{NOW}-1}, p^+_{\text{NOW}})}{4}. \quad (1)$$

Thus, the new position of $p_{\text{NOW}-1}$ is $p^+_{\text{NOW}-1} = o_k$. As a consequence $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$ and $\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$ have changed, but we can easily calculate their new values:

$$\widehat{D}_\lambda(p_{\text{NOW}-2}, p^+_{\text{NOW}-1}) = \widehat{D}_\lambda(p_{\text{NOW}-2}, o_\ell) + \widehat{D}_\lambda(o_\ell, p^+_{\text{NOW}-1})$$
$$= \frac{\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})}{2} + \frac{\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1}) + \widehat{D}_\lambda(p_{\text{NOW}-1}, p^+_{\text{NOW}})}{4},$$

similarly, the value for $\widehat{D}_\lambda(p^+_{\text{NOW}-1}, p_{\text{NOW}})$ can be calculated.

*Rule #4:* Transmission rule: after locally optimizing the position of the interpolation point $p_{\text{NOW}-1}$ and updating the corresponding data $\widehat{\gamma}'(p_{\text{NOW}-1})$ and $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$, agent $i$ transmits this information to agent $i-1$. We assume the transmission is reliable. After this local optimization is performed, the counter NOW is updated to NOW $+ 1$ and Rule #1 is applied again, i.e., the buffer BUFFERARC is updated by dropping all observations $o_j$ between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$.

*Remark 2 (Synchronization assumption):* We assume that when agent $i$ is relocating and transmitting information about $p_{\text{NOW}-1}$ agent $(i-1)$ has not yet projected NOW $- 2$. If this assumption does not hold, i.e., if agent $i-1$ is ready to apply Rule #2 before agent $i$ has applied Rule #4, then agent $i-1$ will have to keep collecting data in its buffer BUFFERARC until agent $i$ transmits the new position of $p_{\text{NOW}-1}$. •

*Remark 3 (Extensions):* In the interest of simplicity, we have omitted two possible generalization that might be useful in practice. First, each agent does not need to know all interpolation points; it would suffice for it to know only the interpolation points located ahead of its position and before the position of the preceding agent. Second, each agent could locally optimize not only a single interpolation point, but it could store a longer buffer and locally optimize arrays of interpolation points. •

This completes our description of the estimate update algorithm and we now focus on the pursuit objective. To uniformly distribute the agents along the boundary $\partial Q$ according to arc length, we use the following update law for their velocities:

$$v_i(t) = v_0 + k_{\text{prop}}(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i)),$$

with $k_{\text{prop}}$, $v_0 > 0$ and $\widehat{L}(P_n, P_m) = \sum_{\alpha=\text{NOW}^n+1}^{\text{NOW}^m}(\|p_{\alpha-1} - p_\alpha\|)$, for all $n, m \in \{1, \ldots, n_{\text{a}}\}$. Here, recall that $p_{\text{NOW}^n}, p_{\text{NOW}^{n+1}}, \ldots, p_{\text{NOW}^m}$ are the interpolation points separating agent $n$ and agent $m$, with $n < m$, and therefore $\widehat{L}$ is the estimated arc length of the portion of $\partial Q$ that has to be traversed to go from the agent $n$ to the agent $m$. The agents have only local information of $\partial Q$ but still they have to estimate the distance, along $\partial Q$, from their clockwise and counterclockwise neighbors in order to calculate their speed. The estimate $\widehat{L}(P_n, P_m)$ is obtained from the approximating polygon formed by the interpolation points. In practice, any agent will speed up if it is closer to the agent behind it, and slow down if closer to the agent in front of it. With a saturation-like function: $\text{sat}(v_i(t)) = \max\{v_{\min}, \min\{v_i(t), v_{\max}\}\}$, we additionally impose that $0 < v_{\min} \leq v_i(t) \leq v_{\max}$ for all $t$.

*Remark 4 (Partial knowledge):* The pursuit objective of the proposed algorithm requires more knowledge than the boundary estimation objective. In fact, to calculate $v_i(t)$, agent $i$ needs to know the position not only of all the interpolation points between itself and $P_{i+1}$, but also of the ones between itself and $P_{i-1}$. Therefore, in addition to the data transmitted according to Rule #4, we require that agent $i$ transmits $p^+_{\text{NOW}}$ and $p_{\text{NOW}+1}$ to $i-1$ and the counter NOW $+ 1$ to $i+1$. •

Now we summarize the discussion in this section with a pseudo-code description of the algorithm in Table I.

### B. Algorithm analysis

Some steps of the algorithm are affected by noise and error: (i) $\widehat{\gamma}'$ and $\widehat{\kappa}$ are only estimate of the true values, (ii) $\widehat{L}$ is an approximation of $L$, (iii) the set $\mathcal{O}$ is a discretization of the subset of $\partial Q$ that agent $i$ is visiting, therefore, the center of the Voronoi cell of the interpolation point $p_{\text{NOW}^i-1}$ might not be calculated exactly. Let $\widehat{\mathbf{D}}(t)$ and $\mathbf{L}(t)$ be the column vectors:

$$\widehat{\mathbf{D}}(t) = \big[\widehat{D}_\lambda(p_1(t), p_2(t)), \ldots, \widehat{D}_\lambda(p_{n_{\text{ip}}-1}(t), p_{n_{\text{ip}}}(t)), \widehat{D}_\lambda(p_{n_{\text{ip}}}(t), p_1(t))\big]^T,$$

TABLE I
ESTIMATE UPDATE AND PURSUIT ALGORITHM

| | |
|---|---|
| **Goal:** | Uniformly distribute the interpolation points according to the pseudo-distance $\widehat{D}_\lambda$, and the agents according to the arc length $\widehat{L}$. |
| **Data:** | Location of the interpolation points, unitary tangent vector at $\partial Q$ at those points, last value of $\widehat{D}_\lambda$ between any two consecutive interpolation points, local tangent and local curvature of the boundary $\partial Q$. |
| **Requires:** | At $t_0 = 0$, $p_i$ lie on $\partial Q$, $\widehat{D}_\lambda$ between any two interpolation points is known, and $o_q = \emptyset$. |

At every sensing instant, the agent at position $P_i(t) = P(t)$ performs:

1: update BUFFERARC$^+$ := BUFFERARC $\cup \{o_{n_o+1}, \widehat{\gamma}'(o_{n_o+1}), \widehat{\kappa}(o_{n_o+1}), \widehat{D}_\lambda(o_{n_o}, o_{n_o+1})\}$
2: **if** $o_q = \emptyset$, **then**
3: **if** $\overline{o_{n_o} o_{n_o+1}} \cap \ell_{\text{NOW}} \neq \emptyset$, **then**
4: $o_q := \text{argmin}_{o_j \in \{o_{n_o}, o_{n_o+1}\}} \|\overline{o_{n_o} o_{n_o+1}} \cap \ell_{\text{NOW}} - o_j\|$
5: **else**
6: **if** $\overline{o_{n_o} o_{n_o+1}} \cap \ell_{\text{NOW}} = \emptyset$ and $\widehat{D}_\lambda(p_{\text{NOW}-1}, o_{n_o+1}) > 2\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$, **then**
7: $o_q := \text{argmin}_{o_j \in \{p_{\text{NOW}-1}, \ldots, o_{n_o+1}\} \subset O} \|o_j - p_{\text{NOW}}\|$
8: **end if**
9: **end if**
10: **end if**
11: **if** $o_q \neq \emptyset$ and $p_{\text{NOW}^i} \neq p_{\text{NOW}^{i+1}-2}$, **then**
12: update the interpolation point $p_{\text{NOW}}$ by projecting it onto $\partial Q$: $p^+_{\text{NOW}} := o_q$
13: calculate $o_k$ as in (1) and update $p_{\text{NOW}-1}$ by: $p^+_{\text{NOW}-1} := o_k$
14: communicate to agent $i-1$: transmit $p_{\text{NOW}+1}$, $p^+_{\text{NOW}}$, $p^+_{\text{NOW}-1}$, $\widehat{\gamma}'(p^+_{\text{NOW}-1})$, $\widehat{D}_\lambda(p_{\text{NOW}-2}, p^+_{\text{NOW}-1})$
15: communicate with agent $i+1$: transmit NOW $+ 1$
16: update the set BUFFERARC and the counter NOW:

$$\text{BUFFERARC}^+ := \{o_k, \ldots, o_{n_o+1}\}, \quad \text{NOW}^+ := \text{NOW} + 1$$

17: **end if**
18: calculate $v_i(t)$: $v_i(t) := \text{sat}(v_0 + k_{\text{prop}}(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i)))$

$$\mathbf{L}(t) = \big[L(P_1(t), P_2(t)), \ldots, L(P_{n_a-1}(t), P_{n_a}(t)), L(P_{n_a}(t), P_1(t))\big]^T.$$

Consider the disagreement vectors $\mathbf{d}(t)$ and $\delta\mathbf{L}(t)$ defined by:

$$\mathbf{d}(t) = \widehat{\mathbf{D}}_\lambda(t) - \frac{\mathbf{1}^T \widehat{\mathbf{D}}_\lambda(t)}{n_{\text{ip}}} \mathbf{1}, \tag{2}$$

$$\delta\mathbf{L}(t) = \mathbf{L}(t) - \frac{\mathbf{1}^T \mathbf{L}(t)}{n_{\text{a}}} \mathbf{1}. \tag{3}$$

Note that they are orthogonal to the vector $\mathbf{1}$, i.e., the column vector in $\mathbb{R}^n$ with all entries equal to 1.

We now establish that the dynamics of $\mathbf{d}$ and $\delta\mathbf{L}$ are input-to-state stable (ISS) where the inputs are the errors and noises above discussed. Because of the ISS property we can conclude that, if the errors are bounded, then the states $\mathbf{d}$ and $\delta\mathbf{L}$ are within a bounded distance from the origin.

*Theorem 1: (ISS of the dynamics of the interpolation points distances)* If the boundary is slowly time-varying and if $t \mapsto L(\partial Q(t))$ is upper and lower bounded uniformly in $t$, then, under the ESTIMATE UPDATE AND PURSUIT ALGORITHM, there exists a sequence of instants $\tau_k$, for $k \in \mathbb{N}_0$, and a sequence of ergodic and doubly stochastic matrices $\mathcal{A}(k)$, for $k \in \mathbb{N}_0$, such that

$$\mathbf{d}(\tau_{k+1}) = \mathcal{A}(k)\mathbf{d}(\tau_k) + \delta\mathbf{u}(\tau_k), \quad k \in \mathbb{N}_0,$$

where $\delta\mathbf{u}(\tau_k) = \mathbf{u}(\tau_k) - \frac{\mathbf{1}^T\mathbf{u}(\tau_k)}{n_{\text{ip}}}\mathbf{1}$, and $\mathbf{u}(\tau_k)$ is a bounded vector taking into account the effect of the estimation errors and of the boundary deformation during the interval $[\tau_k, \tau_{k+1}]$. Furthermore, the dynamics of $\mathbf{d}$ are input-to-state stable with input $\delta\mathbf{u}$.

*Proof:* In what follows we identify $\tau_0 \equiv 0$ and $\tau_k \equiv k$. Let us suppose that $\partial Q(t)$ is time-invariant, and that $\mathcal{O}$ is a continuous (and not discrete) representation of $\partial Q$, i.e., $\mathbf{u}(k) = \mathbf{0}$ for all $k \in \mathbb{N}_0$. Then, because of Rule #2, $p_{\text{NOW}}$ is projected onto itself and, because of Rule #3, $p_{\text{NOW}-1}$ is moved exactly to the center if its Voronoi cell. Suppose that an agent has passed by the point $p_{\text{NOW}}$, and then it can optimally place $p_{\text{NOW}-1}$. As a consequence, the pseudo-distances $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$ and $\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$ will take new values that can be expressed as follows, (recall Figure 4):

$$\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})^+ = \frac{3}{4}\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1}) + \frac{1}{4}\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}}),$$

$$\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})^+ = \frac{1}{4}\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1}) + \frac{3}{4}\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}}),$$

where the superscript $^+$ indicates the new values of the pseudo-distances after $p_{\text{NOW}-1}$ has been optimally placed. For $\alpha \in \{1, \ldots, n_{\text{ip}}\}$, define the doubly stochastic matrix $A_\alpha \in \mathbb{R}^{n_{\text{ip}} \times n_{\text{ip}}}$ by

$$(A_\alpha)_{jh} = \begin{cases} 3/4, & \text{if } j = h = \alpha, \text{ or } j = h = \alpha - 1, \\ 1/4, & \text{if } j = \alpha - 1 \text{ and } h = \alpha, \text{ or viceversa,} \\ \delta_{jh}, & \text{otherwise.} \end{cases}$$

Therefore, $A_\alpha$, for $\alpha \in \{1, \ldots, n_{\text{ip}}\}$, are the matrices determining the dynamic system $\widehat{\mathbf{D}}_\lambda(t_2) = A_\alpha \widehat{\mathbf{D}}_\lambda(t_1)$, where $t_2 > t_1$ is the time when the interpolation point $\alpha$ is moved by an agent to its new Voronoi center and where we are assuming that between $t_1$ and $t_2$ no other interpolation point has been moved. If at the same instant more interpolation points are relocated, then the matrix describing the dynamics is the product of all the $A_\alpha$ that correspond to the relocated interpolation points. The order of the matrix multiplication is irrelevant as one can show that these matrices commute. Let us now derive the dynamics of $\widehat{\mathbf{D}}_\lambda$ when $\partial Q$ is slowly time-varying, while $\mathcal{O}$ is still a continuous representation of $\partial Q$. By assumption $\gamma(t, s)$ is smooth in both its arguments and, as argued above, the projection of the interpolation points is well defined and unique.

Let $t_\alpha^{k+1}$ be the $k+1$-th time that $p_\alpha$ is optimally placed by an agent. Before optimally placing $p_\alpha$, the agent will project $p_{\alpha+1}$. It can be proved that right before placing $p_\alpha$, because the boundary has changed, the pseudo-distance $\widehat{D}_\lambda(p_\alpha, p_{\alpha+1}, t_\alpha^{k+1})$ will differ from $\widehat{D}_\lambda(p_\alpha, p_{\alpha+1}, t_{\alpha+1}^k)^+$ by some noise $g(t_\alpha^{k+1} - t_{\alpha+1}^k)$ which is a continuous function of $t_\alpha^{k+1} - t_{\alpha+1}^k$ and $g(0) = 0$. With $\widehat{D}_\lambda(p_\alpha, p_{\alpha+1}, t_{\alpha+1}^k)^+$ we denote the pseudo-distance between $p_\alpha$ and $p_{\alpha+1}$ right after $p_{\alpha+1}$ has been optimally placed for the $k$th time. Therefore,

the system is evolving according to:

$$\widehat{\mathbf{D}}_\lambda(p_\alpha, p_{\alpha+1}, t_\alpha^{k+1})^+ = A_\alpha\left(\widehat{\mathbf{D}}_\lambda(p_\alpha, p_{\alpha+1}, t_{\alpha+1}^k) + \mathbf{e}_{\alpha+1} g(t_\alpha^{k+1} - t_{\alpha+1}^k)\right), \quad (4)$$

where $\mathbf{e}_\alpha$ is the column vector with null entries but the $\alpha$-th component that is equal to 1, and the subscript $^+$ indicates that the interpolation point $p_\alpha$ has just been optimally placed. Let $\Delta T = \sup_{t\in\overline{\mathbb{R}}_+} \frac{L(\partial Q(t))}{v_{\min}}$. Note that $\Delta T < +\infty$ since by assumption the length of the boundary $\partial Q(t)$ is uniformly upperbounded. This means that at most after $\Delta T$ any interpolation point is updated at least once. Any time that an agent updates any interpolation point $p_\alpha$ the vector $\widehat{\mathbf{D}}_\lambda$ evolves according to (4), where $t_\alpha^{k+1} - t_{\alpha+1}^k$ is upperbounded by $\Delta T$. Since $\Delta T$ is finite, there exists a sequence of instants $\tau_k$, with $k \in \mathbb{N}_0$, such that across the interval $[\tau_k, \tau_{k+1}]$ every interpolation point has been updated at least once by an agent, and:

$$\widehat{\mathbf{D}}_\lambda(k+1) = \mathcal{A}(k)\widehat{\mathbf{D}}_\lambda(k) + \mathbf{u}(k), \quad k \in \mathbb{N}_0. \quad (5)$$

The matrix $\mathcal{A}(k)$ is the product of a finite number $M(k)$ of matrices $A_\alpha$, i.e., $\mathcal{A}(k) = \Pi_{\beta=1}^{M(k)} A_{\alpha_\beta}$, $\alpha_\beta \in \{1, \ldots, n_{\text{ip}}\}$. The value of the index $\alpha_\beta$ depends on the order in which the interpolation points are updated. It is easy to see that $n_{\text{ip}} \leq M(k) \leq n_{\text{a}} n_{\text{ip}}$. Note that $\mathcal{A}(k)$ is doubly stochastic because it is the product of doubly stochastic matrices. Since across the interval $[\tau_k, \tau_{k+1}]$ every interpolation point has been updated at least once by an agent, the graph associated with $\mathcal{A}(k)$ is connected and therefore $\mathcal{A}(k)$ is ergodic (see [11]). Furthermore, $\sup(\tau_{k+1} - \tau_k) \leq \Delta T < +\infty$, and (by [8]) we claim that, if $\mathbf{u}(k) \equiv 0$, then $\widehat{\mathbf{D}}_\lambda(k)$ converges exponentially fast to $\frac{\mathbf{1}^T \widehat{\mathbf{D}}_\lambda(k)}{n_{\text{ip}}}\mathbf{1}$. Consider now the disagreement vector $\mathbf{d}(k)$ defined in (2). Recalling (5), and that $\mathcal{A}(k)$ is doubly stochastic, we can derive the update law of $\mathbf{d}(k)$:

$$\mathbf{d}(k+1) = \mathcal{A}(k)\mathbf{d}(k) + \delta\mathbf{u}(k), \quad k \in \mathbb{N}_0, \quad (6)$$

where $\delta\mathbf{u}(k) = \mathbf{u}(k) - \frac{\mathbf{1}^T\mathbf{u}(k)}{n_{\text{ip}}}\mathbf{1}$. Given the properties of the matrix $\mathcal{A}(k)$, the origin of the unforced system (6) is exponentially stable. Since the input $\delta\mathbf{u}$ enters linearly, we conclude that the system is input-to-state stable ([12]). This implies that $\widehat{\mathbf{D}}_\lambda$ will asymptotically reach a ball centered at $\frac{\mathbf{1}^T \widehat{\mathbf{D}}_\lambda(k)}{n_{\text{ip}}}\mathbf{1}$ (the equilibrium of the unforced system) and with radius that is a class $K$ function of the input, see [12]. This also holds if we now relax the assumption that no errors affect the calculation of the Voronoi centers, because this error enters linearly in the system (6). ■

*Theorem 2 (ISS of the dynamics of the interagent distances):* If the boundary is slowly time-varying and if $t \mapsto L(\partial Q(t))$ is upper and lower bounded uniformly in $t$, then, under the ESTIMATE UPDATE AND PURSUIT ALGORITHM,

$$\delta\dot{\mathbf{L}}(t) = k_{\text{prop}} A(c_1(t), \ldots, c_{n_a}(t))\Big(\delta\mathbf{L}(t) + \delta\mathbf{w}_1(t)\Big) + \delta\mathbf{w}_2(t),$$

where $\delta\mathbf{w}_1(t) = \mathbf{w}_1(t) - \frac{\mathbf{1}^T\mathbf{w}_1(t)}{n_{\text{a}}}\mathbf{1}$, $\delta\mathbf{w}_2(t) = \mathbf{w}_2(t) - \frac{\mathbf{1}^T\mathbf{w}_2(t)}{n_{\text{a}}}\mathbf{1}$, and

$$(A(c_1(t), \ldots, c_{n_a}(t)))_{jh} = \begin{cases} -c_i(t) - c_{i+1}(t), & \text{if } j = h = i, \\ c_{i+1}(t), & \text{if } j = i \text{ and } h = i+1, \\ c_i(t), & \text{if } j = i \text{ and } h = i-1, \\ 0, & \text{otherwise.} \end{cases}$$

with $c_i(t) \in [\beta(t), 1]$ for all $i \in \{1, \ldots, n_{\text{a}}\}$, and $\beta(t) = \frac{\min\{v_{\max} - v_0, v_0 - v_{\min}\}}{k_{\text{prop}} L(\partial Q(t))}$. The variable $\mathbf{w}_2(t) \in \mathbb{R}^{n_{\text{a}} \times 1}$ expresses the change in the arc length distance between any two consecutive agents due to the deformation of $\partial Q(t)$, while $\mathbf{w}_1(t) = \mathbf{L}(t) - \widehat{\mathbf{L}}(t) \in \mathbb{R}^{n_{\text{a}} \times 1}$ is the error due to the fact that the agents do not know exactly $\mathbf{L}$, the arc length distance between them and their neighbors, but only an estimate through the interpolation points $\widehat{\mathbf{L}}$. Furthermore, the dynamics of $\delta\mathbf{L}$ is input-to-state stable with $t \mapsto \delta\mathbf{w}_1(t)$ and $t \mapsto \delta\mathbf{w}_2(t)$ as inputs.

*Proof:* Let us suppose that the $\partial Q(t)$ is time-invariant and that the agents can actually compute without error the arc length distance between them and their clockwise and counterclockwise neighbors, i.e., $\mathbf{w}_1(t) = \mathbf{w}_2(t) = \mathbf{0}$ for all $t \geq 0$. The dynamics for $\mathbf{L}(t)$ can be written as $\dot{L}(P_i(t), P_{i+1}(t)) = v_{i+1} - v_i$, where $v_{i+1} = \text{sat}(v_0 + k_{\text{prop}}(L(P_{i+1}, P_{i+2}) - L(P_i, P_{i+1})))$ and $v_i = \text{sat}(v_0 + k_{\text{prop}}(L(P_i, P_{i+1}) - L(P_{i-1}, P_i)))$. Therefore, if the saturation on the speeds is not active, we have:

$$\dot{L}(P_i(t), P_{i+1}(t)) = k_{\text{prop}}\Big(L(P_{i+1}, P_{i+2}) - 2L(P_i, P_{i+1}) + L(P_{i-1}, P_i)\Big),$$

which in matrix form becomes:

$$\dot{\mathbf{L}}(t) = k_{\text{prop}} \begin{bmatrix} -2 & 1 & 0 & \ldots & 1 \\ 1 & -2 & 1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 1 & -2 & 1 \\ 1 & 0 & \ldots & 1 & -2 \end{bmatrix} \mathbf{L}(t) = k_{\text{prop}} A_L \mathbf{L}(t).$$

If, for agent $i$, the saturation is active, then we have that $v_i = v_0 + k_i'(L(P_i, P_{i+1}) - L(P_{i-1}, P_i))$, where $k_i' = k_{\text{prop}} c_i$, $c_i \leq 1$. In other words, we can think of the saturation function as a change in the gain in the control law. If $v_i = v_{\max}$, then

$$k_i' = \frac{v_{\max} - v_0}{(L(P_i, P_{i+1}) - L(P_{i-1}, P_i))} \geq \frac{v_{\max} - v_0}{L(\partial Q(t))}$$
$$\implies \quad c_i = \frac{k_i'}{k_{\text{prop}}} \geq \frac{1}{k_{\text{prop}}} \frac{v_{\max} - v_0}{L(\partial Q(t))} > 0.$$

If $v_i = v_{\min}$, then

$$k_i' = \frac{v_0 - v_{\min}}{(L(P_i, P_{i+1}) - L(P_{i-1}, P_i))} \geq \frac{v_0 - v_{\min}}{L(\partial Q(t))}$$
$$\implies \quad c_i = \frac{k_i'}{k_{\text{prop}}} \geq \frac{1}{k_{\text{prop}}} \frac{v_0 - v_{\min}}{L(\partial Q(t))} > 0,$$

in any case $c_i \in [\beta(t), 1]$ and $\beta(t) = \frac{\min\{v_{\max} - v_0, v_0 - v_{\min}\}}{k_{\text{prop}} L(\partial Q(t))}$.

Clearly then, if we introduce the saturation on the speeds $v_i$, the dynamics of $\mathbf{L}$ becomes:

$$\dot{\mathbf{L}}(t) = k_{\text{prop}} A(c_1(t), \ldots, c_{n_\text{a}}(t))\mathbf{L}(t). \tag{7}$$

Note that $\beta(t)$ is constant because we are still considering $\partial Q$ time-invariant. Using the properties of Metzler matrices, it can be proved (see Lemma 2 in [13]) that the new matrices $A(c_1(t), \ldots, c_{n_\text{a}}(t))$, like $A_L$, are negative semidefinite. The unique zero eigenvalue is associated with the eigenvector $\mathbf{1}$.

Consider the disagreement $\delta\mathbf{L}$ as described by (3), then

$$\delta\dot{\mathbf{L}}(t) = k_{\text{prop}} A(c_1(t), \ldots, c_{n_\text{a}}(t))\delta\mathbf{L}(t). \tag{8}$$

Let $V(\delta\mathbf{L}(t)) = \delta\mathbf{L}^T(t)\delta\mathbf{L}(t)$ be the candidate Lyapunov function for the system (8), then we have that $\dot{V}(\delta\mathbf{L}(t)) = 2k_{\text{prop}}\delta\mathbf{L}(t)A(c_1(t), \ldots, c_{n_\text{a}}(t))\delta\mathbf{L}(t) \leq 0$, where the equality holds only if the entries of $\delta\mathbf{L}(t)$ are all zero. Since $c_i$ belong to a compact set, the matrices $A(c_1(t), \ldots, c_{n_\text{a}}(t))$ belong to a compact set, and since the eigenvalues of a matrix are continuous functions of its entries (see [14]), then there exists an upperbound $-\rho < 0$ for the eigenvalues that are different from zero and as a consequence $\dot{V}(\delta\mathbf{L}(t)) \leq -\rho\|\delta\mathbf{L}(t)\|^2$. We can then conclude that for the system (8) the origin is exponentially stable and therefore, for the system (7), the equilibrium $\frac{\mathbf{1}^T\mathbf{L}(t)}{n_\text{a}}\mathbf{1}$ is exponentially stable.

Let us now assume that the boundary is slowly-varying and that instead of $L(P_i, P_{i+1})$ the agents use only the approximation $\widehat{L}(P_i, P_{i+1})$, i.e., $\mathbf{w}_1(t)$, $\mathbf{w}_2(t) \neq \mathbf{0}$. Then, the variation in time of the vector $\mathbf{L}(t)$ is due, not only to the fact that the agents speed up and slow down, but also to the deformation of $\partial Q$:

$$\begin{aligned}\dot{\mathbf{L}}(t) &= k_{\text{prop}} A(t)\widehat{\mathbf{L}}(t) + \mathbf{w}_2(t)\\ &= k_{\text{prop}} A(t)(\mathbf{L}(t) + \mathbf{w}_1(t)) + \mathbf{w}_2(t),\end{aligned}$$

where $A(t) = A(c_1(t), \ldots, c_{n_\text{a}}(t))$, $\mathbf{w}_1(t) = \mathbf{L} - \widehat{\mathbf{L}}$, and $\mathbf{w}_2(t)$ expresses the deformation of $\partial Q(t)$. In particular the $i$-th entry of $\mathbf{w}_2(t)$ is equal to $\frac{\partial}{\partial t}\int_{s_i}^{s_{i+1}} \|\gamma'(s,t)\|ds - k_{\text{prop}} A_i(t)(\mathbf{L}(t) + \mathbf{w}_1(t))$, where $A_i(t)$ is the $i$-th row of $A(t)$. Note that $c_i(t) \in [\beta(t), 1]$ for all $i \in \{1, \ldots, n_\text{a}\}$, and that $\beta(t) = \frac{\min\{v_{\max} - v_0, v_0 - v_{\min}\}}{k_{\text{prop}} L(\partial Q(t))}$ is indeed uniformly upper bounded even when $\partial Q$ is time-varying because we assumed that $L(\partial Q(t))$ is upper and lower bounded uniformly in $t$. The vector $\mathbf{w}_2$ is bounded because by assumption the boudary is smooth and slowly time-varying. Using the change of variables in equation (3), and recalling that $A(t)\mathbf{1} = \mathbf{0}$, for all $t$, we have:

$$\dot{\mathbf{L}}(t) = k_{\text{prop}} A(t)\delta\mathbf{L}(t) + k_{\text{prop}} A(t)\delta\mathbf{w}_1(t) + \delta\mathbf{w}_2(t) + \frac{\mathbf{1}^T\mathbf{w}_2(t)}{n_\text{a}}\mathbf{1},$$

where $\delta\mathbf{w}_2(t) = \mathbf{w}_2(t) - \frac{\mathbf{1}^T\mathbf{w}_2(t)}{n_\text{a}}\mathbf{1}$. It is easy to see that we can write $\dot{\mathbf{L}}(t) = \delta\dot{\mathbf{L}}(t) + \frac{\mathbf{1}^T\dot{\mathbf{L}}(t)}{n_\text{a}}\mathbf{1}$ and therefore:

$$\begin{aligned}\delta\dot{\mathbf{L}}(t) + \frac{\mathbf{1}^T\dot{\mathbf{L}}(t)}{n_\text{a}}\mathbf{1} = k_{\text{prop}} A(t)\delta\mathbf{L}(t) + k_{\text{prop}} A(t)\delta\mathbf{w}_1(t) +\\ + \delta\mathbf{w}_2(t) + \frac{\mathbf{1}^T\mathbf{w}_2(t)}{n_\text{a}}\mathbf{1}.\end{aligned}$$

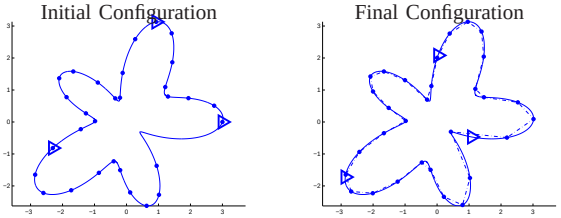


Fig. 5. This figure shows initial and final configuration after 50 seconds simulation obtained by the implementation of the ESTIMATE UPDATE AND PURSUIT ALGORITHM with $n_\text{a} = 3$, $n_{\text{ip}} = 30$, $v_0 = 1$, $k_{\text{prop}} = 0.05$, $\lambda = \frac{10}{11}$. $\partial Q$ is time invariant. The agents position is represented by the triangles and are initialized to be on the boundary $\partial Q$. In the last frame also the approximating polygon is shown.

Since $\mathbf{1}$ is orthogonal to $\delta\mathbf{L}(t)$, $\delta\mathbf{w}_1(t)$, and $\delta\mathbf{w}_2(t)$ we have:

$$\delta\dot{\mathbf{L}}(t) = k_{\text{prop}} A(t)\delta\mathbf{L}(t) + k_{\text{prop}} A(t)\delta\mathbf{w}_1(t) + \delta\mathbf{w}_2(t). \tag{9}$$

The system described by equation (9) is input-to-state stable with inputs $\delta\mathbf{w}_1(t)$ and $\delta\mathbf{w}_2(t)$ because (i) the origin of the unforced system is exponentially stable, and (ii) the right-hand-side of (9) is differentiable and uniformly globally Lipschitz in $\delta\mathbf{L}$, $\delta\mathbf{w}_1(t)$ and $\delta\mathbf{w}_2(t)$, (see [12]).

The ISS property guarantees that if $\mathbf{w}_1$ and $\mathbf{w}_2$ are bouded, then $\mathbf{L}$ will asymptotically reach a ball centered at $\frac{\mathbf{1}^T\mathbf{L}(t)}{n_\text{a}}\mathbf{1}$ (the equilibrium of the unforced system) and with radius that is a class $K$ function of the input, see [12]. The larger $n_{\text{ip}}$ is and the slower the deformation of $\partial Q$ is, then the smaller $\mathbf{w}_1$ and $\mathbf{w}_2$ are, and the closer to $\mathbf{0}$ the disagreement $\delta\mathbf{L}$ will be asymptotically. ■

### C. Simulations

In this section we present results of two different simulations obtained with the implementation of the ESTIMATE UPDATE AND PURSUIT ALGORITHM. In the first simulation the boundary $\partial Q$ is time invariant, while in the second is time varying.

*1) Time-invariant boundary:* In this simulation we use $n_\text{a} = 3$ agents to have an approximation of the nonconvex boundary $\partial Q$ described by:

$$\gamma(\theta) = \left(2 + \cos(10\pi\theta) + 0.5\sin(4\pi\theta)\right)\begin{bmatrix}\cos(2\pi\theta)\\ \sin(2\pi\theta)\end{bmatrix}.$$

The outcome is shown in Figure 5. In order to calculate their speeds, the agents use $v_0 = 1$, and $k_{\text{prop}} = 0.05$. The saturation function for the speed has lower limit $v_{\min} = 0.5$ and upper limit $v_{\max} = 2$. The number of interpolation points is $n_{\text{ip}} = 30$, while $\lambda = \frac{10}{11}$. The simulation time is 50 seconds and the sampling time 0.01 seconds. The plots in Figure 5 corresponds to the positions of the interpolation points and the agents at the initial and final configuration. The interpolation points $p_{\text{NOW}^i}$ for $i \in \{1, \ldots, n_\text{a}\}$ at time $t = 0$ coincide with the positions of the agents. The other interpolation points are randomly distributed on the boundary. In the last frame one can also see the approximating polygon and how close it is to the actual boundary.

Since the pseudo-distance $D_\lambda$ and the arc length $L$ can be calculated after the simulation is completed, we use $D_\lambda$ and

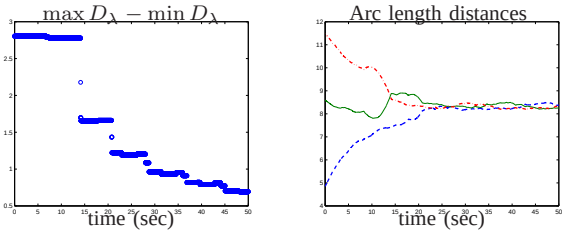


Fig. 6. ESTIMATE UPDATE AND PURSUIT ALGORITHM This plots refers to the case of $\partial Q$ being time-invariant. In the first plot from right it is shown the error $\max_{i\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_i,p_{i+1}) - \min_{i\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_i,p_{i+1})$ vs time. The second plot shows the arc length distances between the three agents.

$L$ instead of their estimate $\widehat{D}_\lambda$ and $\widehat{L}$ to show the algorithm performance. Figure 6 does indeed show the convergence of the algorithm. In the first plot we can see that the consensus on the pseudo-distance $D_\lambda(p_i,p_{i+1})$, between any two consecutive interpolation points, is reached. The quantity

$$\max_{\alpha\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_\alpha,p_{\alpha+1}) - \min_{\alpha\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_\alpha,p_{\alpha+1})$$

does not vanish because of numerical errors in the estimate $\widehat{D}_\lambda$. The second plot shows how the agents get uniformly spaced along the boundary. The steady state values of the arc length distances oscillates around $8.3$ which is the target value. The noise is again due to the fact that the agents only estimate the arc length using the positions of the interpolation points.

*2) Slowly time-varying boundary:* In this simulation we used $n_{\text{a}} = 4$ agents to have an approximation of the nonconvex boundary $\partial Q(t)$ described by:

$$\gamma(\theta,t) = \left(2 - \frac{2t}{t_f} + \left(2+\cos(10\pi\theta)+0.5\sin(4\pi\theta)\right)\frac{t}{t_f}\right)\begin{bmatrix}\cos(2\pi\theta)\\ \sin(2\pi\theta)\end{bmatrix},$$

with $\theta \in [0,1)$, $t_f = 200$ seconds as shown in Figure 7. The values of $v_0$, $k_{\text{prop}}$, $v_{\text{min}}$, $v_{\text{max}}$ and $\lambda$ are respectively: 1, 0.05, 0.5, 2, and $\frac{10}{11}$. The simulation time is 200 seconds, the sampling time $0.01$ seconds. The plots in Figure 7 correspond to the positions of the interpolation points and the agents at four different instants, $t = 0$, $t = 50$, $t = 100$, and $t = 200$ seconds respectively. The algorithm is initialized with the agents on the boundary. The interpolation points $p_{\text{NOW}^i}$ at time $t = 0$ coincide with the positions of the agents. The other interpolation points are randomly distributed. In the last frame we can also see the approximating polygon and how close to the actual boundary is. From the frames in Figure 7 it is clear that the agents can adapt as $\partial Q$ changes.

The pseudo-distance $D_\lambda$ is well defined only if the interpolation points belong to the boundary $\partial Q$. Since the boundary changes with time, the interpolation points are only for some time on the boundary after an agents has projected them. So, we consider as pseudo-distance between any two consecutive interpolation points in a certain time $\tau$ the pseudo-distance between their radial projection onto $\partial Q(\tau)$. The disagreement in the placement of the interpolation points, where $D_\lambda$ is redefined as just explained, is shown in the first plot of Figure 8. The arc length between any two consecutive agents is shown

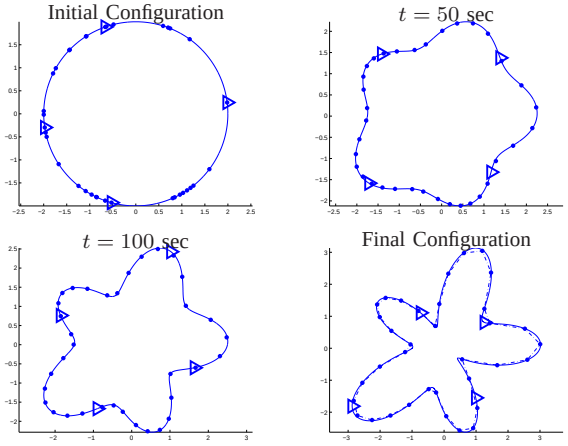


Fig. 7. This figure shows four different instants of the 200 seconds simulation obtained by implementing the ESTIMATE UPDATE AND PURSUIT ALGORITHM with $n_{\text{a}} = 4$, $n_{\text{ip}} = 35$, $v_0 = 1$, $k_{\text{prop}} = 0.05$, $\lambda = \frac{10}{11}$. The boundary $\partial Q$ is slowly time-varying in this case. The agents positions are represented by triangles and initialized to be on the boundary $\partial Q$. The last frame also shows the approximating polygon.

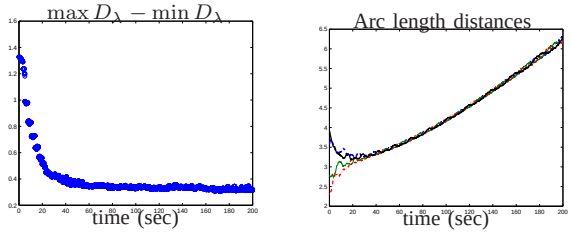


Fig. 8. ESTIMATE UPDATE AND PURSUIT ALGORITHM. This figure refers to the case of $\partial Q$ being slowly time-varying. In the first plot from the right we shown the error $\max_{i\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_i,p_{i+1}) - \min_{i\in\{1,\dots,n_{\text{ip}}\}} D_\lambda(p_i,p_{i+1})$ vs time. The second plot shows the arc length distances between the four agents.

in the second plot of Figure 8. The four distances increase with time because $L(\partial Q)$, the total length of the boundary, increases with time. Clearly the variables $n_{\text{a}}$ and $n_{\text{ip}}$ are design variables and are results of two different trade-offs. In deciding the number of agents, the speed with which the boundary changes and the maximum speed at which the robots can move play an important role. If the boundary changes very fast and the maximum speed of the robots is fixed, then a larger network will guarantee better performances. If the boundary changes slowly, then few robots (1 or 2) might suffice. In deciding the number of interpolation points, on the other hand, the most important role is played by the complexity of the boundary measured by the number of inflection points. To have good performance, $n_{\text{ip}}$ should increase as the number of the inflection points of the boundary increases.

## IV. CONCLUSIONS AND FUTURE WORK

In this paper we have addressed the problem of boundary estimation and tracking by means of robotic sensors. We have presented an algorithm to position interpolation points along a time-varying boundary in such a way as to obtain an approximating polygon with some optimality features.

Each mobile agent is equipped with (1) a sensor that provides only local information about the tangent and curvature of the boundary, and (2) a communication device that enables information exchanges between clockwise and counterclockwise neighbors along the boundary. The algorithm allows the agents to place a set of interpolation points uniformly spaced according to the estimate of the pseudo-distance $D_\lambda$. The vertices of the approximating polygon are the interpolation point positions. The algorithm is proved to converge even if the boundary is slowly-moving. Tools from consensus analysis allow us to prove the correctness of the algorithm.

An important problem for future research concerns the possible and realistic occurrence of boundary splits. In other words, it would be of interest to consider problems where the region enclosed in the boundary can split into two or more separate regions. Therefore, it will be valuable, for real-time implementation, to investigate how to enable the agents to detect splitting events and how to modify the ESTIMATE UPDATE AND PURSUIT ALGORITHM so that the agents can estimate the boundaries of the regions. Experimental work could also suggest further modification to the algorithm to improve its performances. Additionally, we plan to devise algorithms to monitor 3-dimensional regions, such as clouds of chemical pollutants and to extend the presented algorithm to monitor buildings with camera-like networks.

## ACKNOWLEDGMENTS

This material is based upon work supported in part by ARO MURI Award W911NF-05-1-0219 and NSF SENSORS Award IIS-0330008.

## REFERENCES

[1] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization* (S. Butenko, R. Murphey, and P. M. Pardalos, eds.), pp. 317–330, Kluwer Academic Publishers, 2003.

[2] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: Asynchronous communication and physical scales," in *Cooperative Control* (V. Kumar, N. E. Leonard, and A. S. Morse, eds.), vol. 309 of *Lecture Notes in Control and Information Sciences*, pp. 25–42, Springer Verlag, 2004.

[3] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Transactions*, vol. 46, no. 1, pp. 3–13, 2007.

[4] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Sciences*, vol. 37, no. 6, pp. 351–360, 2006.

[5] F. Zhang and N. E. Leonard, "Generating contour plots using multiple sensor platforms," in *IEEE Swarm Intelligence Symposium*, (Pasadena, CA), pp. 309–316, June 2005.

[6] A. Savvides, J. Fang, and D. Lymberopoulos, "Using mobile sensing nodes for boundary estimation," in *Workshop on Applications of Mobile Embedded Systems (Held in conjunction with MobiSys 2004)*, (Boston, MA), June 2004.

[7] P. M. Gruber, "Approximation of convex bodies," in *Convexity and its Applications* (P. M. Gruber and J. M. Willis, eds.), pp. 131–162, Birkhäuser Verlag, 1983.

[8] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[9] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[10] D. E. McLure and R. A. Vitale, "Polygonal approximation of plane convex bodies," *Journal of Mathematical Analysis and Applications*, vol. 51, no. 2, pp. 326–358, 1975.

[11] M. Fielder, *Special Matrices and their Applications in Numerical Mathematics*. Martinus Nijhoff Publishers, 1986.

[12] H. K. Khalil, *Nonlinear Systems*. Englewood Cliffs, NJ: Prentice Hall, 2 ed., 1995.

[13] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," Tech. Rep. CCDC-06-0125, Center for Control, Dynamical Systems and Computation. University of California at Santa Barbara, 2006. Available electronically at `http://ccdc.mee.ucsb.edu`.

[14] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.