Monitoring Environmental Boundaries with a Robotic Sensor Network

Sara Susca
Electrical & Computer Engineering
UC Santa Barbara
sara@ece.ucsb.edu

Sonia Martínez
School of Engineering
UC San Diego
soniamd@ucsd.edu

Francesco Bullo
Mechanical Engineering
UC Santa Barbara
bullo@engineering.ucsb.edu

***Abstract*— In this paper we propose and analyze an algorithm to monitor an environmental boundary with mobile sensors. 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 known results on approximation of convex bodies. The algorithm is provably convergent for static boundaries and also for slowly-moving boundaries because of certain input-to-state stability properties.**

I. Introduction

Recently much 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 sensors to detect the boundary of a region of interest and estimate it as it evolves. Boundary estimation and tracking is useful is numerous applications such as detection of harmful algae bloom [1], [2], oil spill [3], and fire containment [4], [5]. In [1], Bertozzi *et al.* adopt the so called "snake algorithm" (from the computer vision literature) to detect and track the boundary of harmful algae bloom. The agents are equipped with a chemical sensor that is able to measure concentration gradient and with a communication system that is able to exchange information with a data fusion center. In [2], Kemp *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] the authors use a random coverage controller, a collision avoidance controller and a bang-bang angular velocity controller to detect and surround an oil spill. In [5] Casbeer *et al.* describe an algorithm that allows LASE (Low Altitude Short Endurance) Unmanned Vehicles to closely monitor the boundary of a fire. The LASEs have 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 [6]. 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 estimate the curvature of the field in the center of the formation.

In this paper we propose an algorithm to estimate and reconstruct the boundary of a region. We require a group of Unmanned Air Vehicles (UAVs) to optimally place some interpolation points on the boundary of a region of interest. The boundary can then be reconstructed by linear interpolation of the interpolation points. We assume that (i) the UAVs do not have a priori knowledge of the boundary, (ii) they are equipped with a camera sensor and with algorithms to estimate the tangent and curvature of the boundary, and (iii) a wireless communication network provides the UAVs with the ability to download and upload the interpolation points from and to a data center. The algorithm is provably convergent for static boundaries and also for slowly-moving boundaries because of certain input-to-state stability properties.

The novelty of this paper is in the criterion used to optimally place interpolation points in such a way that they are uniformly distributed according to a curvature-weighted distance function defined along the boundary. The curvature-weighted distance function was inspired by the literature on optimal approximation of convex bodies by polygons e.g., see the survey [7], [8].

The convergence of the algorithm is proven using tools from the theory of consensus algorithms. An incomplete list of references includes [9], [10], and [11]. The effort of the authors is in proving that the infinite product of some matrices (that belong to a finite set, [9], or infinite set, [10], [11]) converges to a rank-one matrix.

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

II. Basic ideas in approximation of convex bodies

Here we review some known useful results from approximation of strictly convex bodies, e.g., see the extensive surveys [8], [7]. In the standard literature on convex bodies approximations, the symmetric difference δ^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 μ is the Lebesque measure on $\mathbb{R}^d$. If Q is the body to be approximated by an inscribed n-vertices polygon P_n, then $\delta^S(Q, P_n) = \mu(Q) - \mu(P_n)$. For n sufficiently

large, McLure and Vitale [12] 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, ∂Q is the boundary of Q, ρ and $\kappa = \rho^{-1}$ are the curvature radius and curvature of the boundary, respectively, ℓ is the arc length along ∂Q, and θ is the angular position in a polar variable parametrization of ∂Q. To construct the best approximating polygon P_n^* for a strictly convex body, McLure and Vitale in [12] suggest the *method of empirical distributions*. According to this method, the positions θ_i, $i \in \{1, \dots, n\}$, of the n vertices along ∂Q should be uniformly distributed according to $D^S(i) = \int_{\theta_i}^{\theta_{i+1}}\rho(\theta)^{2/3}d\theta$. Interpolating polygons computed according to the method of empirical distributions converge to P_n^* as $n \to +\infty$.

For smooth non-convex bodies with a small number of saddle points, the method of empirical distributions also yields a nearly optimal distribution as $n \to +\infty$ because of the local convexity of the body. We show how to do this in what follows. Since the curvature radius ρ may be unbounded at some point of a non-convex 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_λ 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 $\int_0^{2\pi}\rho(\theta)^\alpha = \int_{\partial Q}\kappa(\ell)^{1-\alpha}d\ell$, see [8]. Introducing the convex combination with arc length, we guarantee that $D_\lambda(i)$ is non-zero whenever the vertices i and $i+1$ do not coincide.

III. Boundary estimation algorithm

In this section we propose and analyze an algorithm that uniformly distributes the interpolation according to the pseudo-distance D_λ introduced in the previous section.

We suppose that the sensing agents can locally estimate the tangent and the curvature of ∂Q. For the case of UAVs surveilling a visible boundary, this information could be provided by a camera and an edge detection algorithm. Another possibility is to substitute every agent with a formation of chemical sensors. In the recent work [6] the authors propose an optimal formation of four agents to estimate the gradient and the curvature of a given level set in a field. We assume that an initial estimate of the boundary is available so that the interpolation points can be distributed (possibly non uniformly) on the boundary and the estimated pseudo-distance $\widehat{D}_\lambda$ between any two neighbors is known. We assume also that every agent is equipped with wireless communication devices to communicate with a data fusion center.

To interpolate the unknown slowly time-varying boundary ∂Q, we introduce a counterclockwise ordered set of interpolation points $\{p_1, \dots, p_{n_{\text{ip}}}\}$ that are the vertices of the interpolating polygon. These are virtual positions stored in a data fusion center together with the tangent of ∂Q at all interpolation points, and the pseudo-distance between any interpolation point and its counterclockwise neighbor. The agents have two objectives: (i) update the interpolation points such that they are uniformly distributed along ∂Q according to the estimated pseudo-distance $\widehat{D}_\lambda$, (ii) be equally distributed along the boundary according to arc length distance.

To achieve these objectives we propose a novel ESTIMATE UPDATE AND PURSUIT ALGORITHM that can be summarized as follows. After reaching a point on ∂Q, the sensing agents move along ∂Q to collect the following data: (i) points belonging to ∂Q, and (ii) tangent and curvature of ∂Q at those points. Using these measurements and communicating with the data center, they complete the following three steps. In the first step, they determine which interpolation point p_i they are closer to and then project it onto the measured boundary. In the second step, they adjust p_{i-1} so that it is at the center of its Voronoi cell along ∂Q. In the third step, they estimate the arc length of distance between them and their immediate clockwise and counterclockwise neighbors and use this information to speed up or slow down. The first two steps have the combined effect of updating the local estimates of the boundary. Thanks to the third step, the agents distribute themselves uniformly along the boundary.

In what follows we present the ESTIMATE UPDATE AND PURSUIT ALGORITHM in some detail and we analyze its stability.

A. Problem setup and notation

Let $\|v\|$ denote the Euclidean norm of $v \in \mathbb{R}^n$. If v is a scalar, then $|v|$ denotes its absolute value. Let $\overline{\mathbb{R}}_+$ be the set of non negative real numbers and $\mathbb{N}_0 = \mathbb{N}\cup\{0\}$. Let $\mathbf{1}$ be the column vector in $\mathbb{R}^n$ with all entries equal to 1. Let ∂Q be the boundary of a connected, and possibly non-convex 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 at time 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 ∂Q is upperbounded and lowerbounded uniformly in t. We define the curvature $\kappa\colon [0, 1] \to \bar{\mathbb{R}}_+$ of the curve γ by: $\kappa(s) = \frac{\|\gamma'(s)\times\gamma''(s)\|}{\|\gamma'(s)\|^3}$.

Let $p_1, \dots, p_{n_{\text{ip}}} \in \mathbb{R}^2$ be the locations of n_{ip} ordered interpolation points. Let $P_i(t)$, with $i \in \{1, \dots, n_{\text{a}}\}$ and $n_{\text{ip}} \gg 3n_{\text{a}}$, be the positions of the sensing agents at a given time. Both the interpolation points and the sensing agents are ordered counterclockwise. We assume that the sensing agents move counterclockwise along the boundary, with speed v_i.

We assume that each agent maintains some variables in its memory that are described as follows. The state of the sensing agent i is:

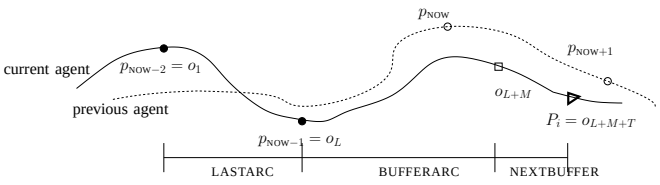
$$\{\text{NOW}^i, \text{LASTARC}^i, \text{BUFFERARC}^i, \text{NEXTBUFFER}^i\},$$

where the first variable is a counter, and the other three are a discrete representation of the subset of ∂Q the sensing agent is flying over. For simplicity we will omit the upperscript and lowerscript i and we will introduce them when necessary.

Let NOW $\in \{1, \dots, n_{\text{ip}}\}$ be the next point to be projected onto ∂Q. Let $\mathcal{O} = \text{LASTARC} \cup \text{BUFFERARC} \cup \text{NEXTBUFFER} \subset \partial Q$ be the set of observations collected by the sensing agent up to time t while going from $p_{\text{NOW}-2}$ towards p_{NOW} along ∂Q defined as follows:

$$\begin{aligned} \text{LASTARC} &= \{o_1, \dots, o_L\}, \\ \text{BUFFERARC} &= \{o_{L+1}, \dots, o_{L+M}\}, \\ \text{NEXTBUFFER} &= \{o_{L+M+1}, \dots, o_{L+M+T}\}, \end{aligned} \tag{1}$$

where $L, M \in \mathbb{N}$, $T \in \mathbb{N}_0$, $o_1 = p_{\text{NOW}-2}$, $o_L = p_{\text{NOW}-1}$, and $o_{L+M+T} = P(t)$. The following figure illustrates these notation and quantities. The solid line represents ∂Q as seen by the agent i, while the dashed line represents ∂Q as seen by the agent $i-1$. The sensing agent is represented by a triangle. The white circles represent the interpolation points before the agent updates them, whereas the black circles represent the interpolation points after the agent has updated them. The square represents the last point belonging to BUFFERARC. The first and the last point of the three data structures LASTARC, BUFFERARC, and NEXTBUFFER are shown, the others are omitted for clarity.



Before defining the point o_{L+M} and the index M we introduce the set of estimated tangent vectors to ∂Q, $\widehat{\gamma}' \colon \mathcal{O} \to \mathbb{R}^2$, and the set of estimated curvature of ∂Q, $\widehat{\kappa} \colon \mathcal{O} \to \bar{\mathbb{R}}_+$. In other words, $\widehat{\gamma'}(o_j)$ and $\widehat{\kappa}(o_j)$ are estimated tangent vector and curvature at the point o_j, for $j \in \{1, \dots, L+M+T\}$. We can now define $\widehat{D}_\lambda \colon \mathcal{O} \times \mathcal{O} \to \bar{\mathbb{R}}_+$ as the discretized pseudo-distance between two observations o_j and o_h, with $h, j \in \{1, \dots, L+M+T\}$, and $h > j$. We shall characterize implicitly the observation o_{L+M} as follows:

$$\widehat{D}_\lambda(o_{L+1}, o_{L+M}) = 2\widehat{D}^-_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}}),$$

where $\widehat{D}^-_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$ is the estimated pseudo-distance between $p_{\text{NOW}-1}$ and p_{NOW} when an agent updated for the last time p_{NOW}. This information is assumed to be stored in the data center.

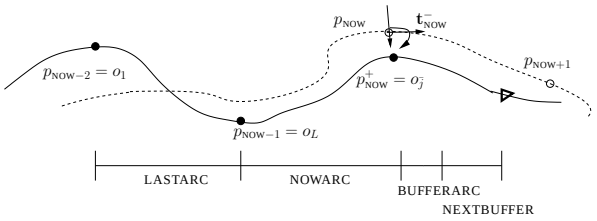
The sets LASTARC, and BUFFERARC will be used by the sensing agents to define the projection onto ∂Q of p_{NOW} and the Voronoi center $\widehat{C}_{\text{NOW}-1}$ of the interpolation point $p_{\text{NOW}-1}$. We recall that the positions $o_1, \dots, o_{L+M+T}$ are points on the plane that the sensing agent has visited in previous instants while moving along the boundary ∂Q, i.e., $o_j = P(\tau)$ for some $\tau < t$ and for all $j \in \{1, \dots, L+M+T\}$. We can say that the points $o_1, \dots, o_{L+M+T}$ are a fine discretization of the portion of ∂Q from $p_{\text{NOW}-2}$ to the current position of the sensing agent P, while the indices $p_{\text{NOW}-2}$, $p_{\text{NOW}-1}$, and p_{NOW} are a coarser discretization of the same arc.

To calculate the Voronoi cell $\widehat{V}_{\text{NOW}-1}$ along ∂Q of the interpolation point $p_{\text{NOW}-1}$, we first need to project p_{NOW} on ∂Q. Let $o_{\bar{j}}$ be the projection of p_{NOW}, defined by:

$$p^+_{\text{NOW}} := o_{\bar{j}} = \text{argmin}_{o_j \in \text{BUFFERARC}} \|(o_j - p_{\text{NOW}}) \cdot \mathbf{t}^-_{\text{NOW}}\|,$$

where $\mathbf{t}^-_{\text{NOW}} = \frac{\widehat{\gamma'}(p_{\text{NOW}})}{\|\widehat{\gamma'}(p_{\text{NOW}})\|}$ is the unit-length tangent vector at $\partial Q(t^-)$ at the interpolation point p_{NOW} last time the interpolation point was updated. In other words the projection of p_{NOW} at time t^+ on $\partial Q(t^+)$ is the intersection of $\partial Q(t^+)$ with the normal vector to $\partial Q(t^-)$ at p_{NOW} at time t^-. This projection is univocally defined and has the following properties. If ∂Q is time-invariant then $p^-_{\text{NOW}} = p^+_{\text{NOW}}$, if ∂Q is slowly time-varying then p^+_{NOW} is close to the orthogonal projection of p^-_{NOW} onto $\partial Q(t^+)$.

We can now define the set NOWARC and update BUFFERARC as follows: NOWARC $= \{o_{L+1}, \dots, o_{\bar{j}}\}$, BUFFERARC $=$ BUFFERARC $\setminus \{o_{L+1}, \dots, o_{\bar{j}}\}$. In the following figure, the agent (i) projects the interpolation point p_{NOW} onto $\partial Q(t)$, (ii) update the state variable BUFFERARC and generate the variable NOWARC.



Using the collected data, the sensing agents can numerically evaluate the pseudo-distances $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$ and $\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$ between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$, and between $p_{\text{NOW}-1}$ and p_{NOW}. Recall that $p_{\text{NOW}-2} = o_1$, $p_{\text{NOW}-1} = o_L$. Let then $\widehat{V}_{\text{NOW}-1} = \{o_\ell, \dots, o_u\}$, where $o_\ell \in$ LASTARC and $o_u \in$ NOWARC are implicitly defined as:

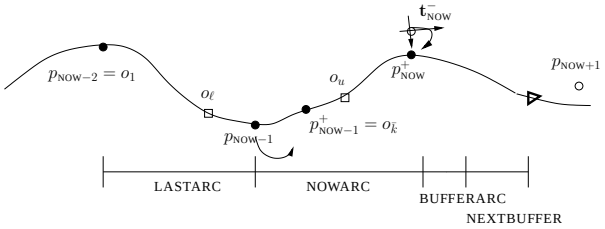
$$\widehat{D}_\lambda(o_1, o_\ell) = \widehat{D}_\lambda(o_\ell, o_L) = \frac{\widehat{D}_\lambda(o_1, o_L)}{2},$$

$$\widehat{D}_\lambda(o_L, o_u) = \widehat{D}_\lambda(o_u, o_{\bar{j}}) = \frac{\widehat{D}_\lambda(o_L, o_{\bar{j}})}{2}.$$

The point o_ℓ 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_{NOW} after the latter was projected on ∂Q. We can implicitly define the Voronoi center $\widehat{C}_{\text{NOW}-1} := o_{\bar{k}} \in$ LASTARC $\cup$ NOWARC by:

$$\widehat{D}_\lambda(o_\ell, o_{\bar{k}}) = \widehat{D}_\lambda(o_{\bar{k}}, o_u) = \frac{\widehat{D}_\lambda(o_1, o_L) + \widehat{D}_\lambda(o_L, o_{\bar{j}})}{4}.$$

In the following figure, the agent (i) calculates the Voronoi cell $\widehat{V}_{\text{NOW}-1}$, and (ii) updates the interpolation point $p_{\text{NOW}-1}$ to lie optimally between $p_{\text{NOW}-2}$ and p_{NOW}.



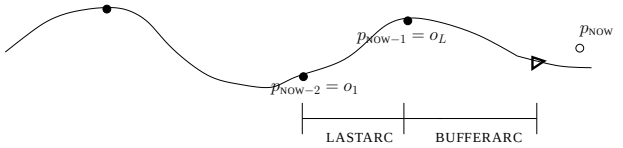
The sensing agent can now update once more the state variables as follows:

$$\begin{aligned}
\text{NOW}^+ &= \text{NOW} + 1,\\
\text{LASTARC}^+ &= \{o_{\bar{k}}, \ldots, o_{\bar{j}}\},\\
\text{BUFFERARC}^+ &= \{o_{\bar{j}+1}, \ldots, o_q\},\\
\text{NEXTBUFFER}^+ &= \{o_{q+1}, \ldots, o_{L+M+T}\},
\end{aligned}$$

where $o_q \in$ BUFFERARC $\cup$ NEXTBUFFER is implicitly defined by: $\widehat{D}_\lambda(o_{\bar{j}+1}, o_q) = 2\widehat{D}^-_\lambda(p_{\text{NOW}^+-1}, p_{\text{NOW}^+})$. If $\widehat{D}_\lambda(o_{\bar{j}+1}, o_{L+M+T}) < 2\widehat{D}^-_\lambda(p_{\text{NOW}^+-1}, p_{\text{NOW}^+})$ then

$$\begin{aligned}
\text{BUFFERARC}^+ &= \text{BUFFERARC} \cup \text{NEXTBUFFER}\\
\text{NEXTBUFFER}^+ &= \emptyset.
\end{aligned}$$

The following figure shows the state variables update as just described, in the case that NEXTBUFFER $= \emptyset$.



This completes our description of the estimate update algorithm and we now focus on the pursuit objective. To uniformly distribute the sensing agents along the boundary ∂Q according to arc length, we will use the following update law for their velocities: $v_i(t) = v_0 + k(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i))$, with $k, v_0 > 0$ and $\widehat{L}(P_n, P_m) = \sum_{j=\text{NOW}^n+1}^{\text{NOW}^m}(\|p_{j-1} - p_j\|)$, 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 $n < m$. $\widehat{L}$ is the estimated arc length of the portion of ∂Q that has to be traversed to go from the sensing agent n to the sensing agent m. The sensing agents have only local information of ∂Q but still they have to estimate the distance, along ∂Q, from their clockwise and counterclockwise neighbors in order to calculate their speed. The estimate $\widehat{L}(P_n, P_m)$ is obtained by 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: $\text{sat}(v_i(t)) = \max\{v_\text{min}, \min\{v_i(t), v_\text{max}\}\}$, we will impose though that $0 < v_\text{min} \le v_i(t) \le v_\text{max}$ for all t.

B. Estimate Update and Pursuit Algorithm

In this section we present an algorithm that allows n_a sensing agents to equally distribute the n_ip interpolation points along ∂Q, according to the pseudo-distance $\widehat{D}_\lambda$. Also the algorithm uniformly distributes the n_a sensing agents along ∂Q, according to the arc length. The algorithm is summarized in the following table.

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 sets LASTARC, BUFFERARC, and NEXTBUFFER are discretization of the subset of ∂Q that agent i is visiting, therefore, the center of the Voronoi cell

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

Assume data is as stated in (1). At every sensing instant, the agent at position $P_i(t) = P(t)$ performs:

1: **if** $\widehat{D}_\lambda(o_{L+1}, P(t)) > 2\widehat{D}^-_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$, **then**
2: update observations NEXTBUFFER$^+$:= NEXTBUFFER $\cup \{P(t)\}$,
3: **else**
4: update observations BUFFERARC$^+$:= BUFFERARC $\cup \{P(t)\}$.
5: **end if**
6: estimate $\widehat{\gamma'}(P(t))$, $\widehat{\kappa}(P(t))$, and $\widehat{D}_\lambda(o_{L+M+T}, P(t))$.
7: **if** NEXTBUFFER $\neq \emptyset$ and $p_{\text{NOW}^i} \neq p_{\text{NOW}^{i+1}-2}$ **then**
8: update the interpolation point p_NOW by projecting it onto ∂Q:

$$p^+_\text{NOW} := o_{\bar{j}}, \quad o_{\bar{j}} = \text{argmin}_{o_j \in \text{BUFFERARC}} \|(o_j - p_\text{NOW}) \cdot \mathbf{t}^-_\text{NOW}\|,$$

9: update the set BUFFERARC and generate the set NOWARC by:

$$\begin{aligned}
\text{BUFFERARC}^+ &:= \text{BUFFERARC} \setminus \{o_{L+1}, \ldots, o_{\bar{j}}\},\\
\text{NOWARC}^+ &:= \{o_{L+1}, \ldots, o_{\bar{j}}\},
\end{aligned}$$

10: calculate $\widehat{C}_{\text{NOW}-1} := o_{\bar{k}}$ and update $p_{\text{NOW}-1}$ by $p^+_{\text{NOW}-1} := o_{\bar{k}}$,
11: communicate with data center: transmit $p_{\text{NOW}-1}$, p_NOW, $\widehat{\gamma'}(p_{\text{NOW}-1})$, $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$, $\widehat{D}_\lambda(p_{\text{NOW}-1}, p_\text{NOW})$ and receive $p_{\text{NOW}+1}$, $\widehat{\gamma'}(p_{\text{NOW}+1})$, $\widehat{D}_\lambda(p_\text{NOW}, p_{\text{NOW}+1})$,
12: update the counter NOW and the set LASTARC by

$$\text{NOW}^+ := \text{NOW} + 1, \text{ LASTARC}^+ := \{o_{\bar{k}}, \ldots, o_{\bar{j}}\},$$

13: update the sets BUFFERARC and NEXTBUFFER as follows:
14: **if** $\exists o_q \in$ BUFFERARC $\cup$ NEXTBUFFER s.t. $\widehat{D}^-_\lambda(o_{\bar{j}+1}, o_q) \ge 2\widehat{D}_\lambda(p_{\text{NOW}^+-1}, p_{\text{NOW}^+})$, **then**
15:

$$\begin{aligned}
\text{BUFFERARC}^+ &:= \{o_{\bar{j}+1}, \ldots, o_q\},\\
\text{NEXTBUFFER}^+ &:= \{o_{q+1}, \ldots, o_{L+M+T}\},
\end{aligned}$$

16: **else**
17:

$$\begin{aligned}
\text{BUFFERARC}^+ &:= \text{BUFFERARC} \cup \text{NEXTBUFFER},\\
\text{NEXTBUFFER}^+ &:= \emptyset.
\end{aligned}$$

18: **end if**
19: **end if**
20: communicate with P_{i+1} and P_{i-1}: receive NOW^{i+1}, NOW^{i-1}, transmit NOW^i. Communicate with the data center: receive the interpolation points with id between NOW^{i-1} and NOW^{i+1}.
21: calculate $v_i(t)$: $v_i(t) = \text{sat}(v_0 + k(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i)))$.

of the interpolation point p_{NOW^i-1} might not be calculated exactly. Let $\widehat{\mathbf{D}}(t)$ and $\mathbf{L}(t)$ be the column vectors:

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

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

Consider now the disagreement vectors $\mathbf{d}(k)$ and $\delta\mathbf{L}(t)$

defined as follows:

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

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

note that they are orthogonal to the vector $\mathbf{1}$.

Theorem 1: The evolution of the disagreement vectors defined by (2) and by (3) under the ESTIMATE UPDATE AND PURSUIT ALGORITHM is input-to-state stable with respect to estimation noise and deformation of the boundary $\partial Q(t)$. We omit the proof for lack of space, the interested reader is referred to [13].

Because of the ISS property we can conclude that as long as the errors are small, the states $D_\lambda(p_i, p_{i+1})$ and $L(P_i, P_{i+1})$ will be close to the equilibrium of the unperturbed system, i.e., $D_\lambda(p_i, p_{i+1}) = D_\lambda(p_{i+1}, p_{i+2})$ for all $i \in \{1, \ldots, n_{\text{ip}}\}$ and $L(P_i, P_{i+1}) = L(P_{i+1}, P_{i+2})$ for all $i \in \{1, \ldots, n_{\text{a}}\}$.

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 ∂Q is time invariant, while in the second is time varying.

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

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

The outcome is shown in Figure 1. In order to calculate their speeds, the sensing agents use $v_0 = 1$, and $k = 0.05$. 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 1 corresponds to the positions of the interpolation points and the sensing agents at the initial and final configurations. The interpolation points p_{NOW^i} for $i \in \{1, \ldots, n_{\text{a}}\}$ coincide with the positions of the sensing 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_λ and the arc length L can be calculated after the simulation is completed, we use D_λ and L instead of their estimate $\widehat{D}_\lambda$ and $\widehat{L}$ to show the algorithm performance. Figure 2 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 $\max_{i\in\{1,\ldots,n_{\text{ip}}\}} D_\lambda(p_i, p_{i+1}) - \min_{i\in\{1,\ldots,n_{\text{ip}}\}} D_\lambda(p_i, p_{i+1})$ 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.

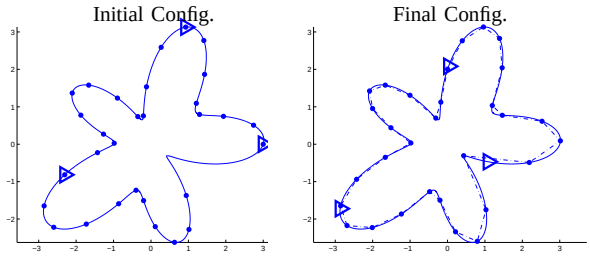


Fig. 1. This figure shows initial and final configuration after 50 seconds simulation obtained by the implementation of the ESTIMATE UPDATE AND PURSUIT ALGORITHM with $n_{\text{a}} = 3$, $n_{\text{ip}} = 30$, $v_0 = 1$, $k = 0.05$, $\lambda = \frac{10}{11}$. ∂Q is time invariant. The sensing agents'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

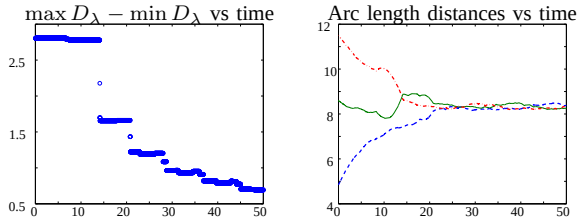


Fig. 2. ESTIMATE UPDATE AND PURSUIT ALGORITHM This plots refers to the case of ∂Q being time-invariant. In the first plot from right it is shown the error $\max_{i\in\{1,\ldots,n_{\text{ip}}\}} D_\lambda(p_i, p_{i+1}) - \min_{i\in\{1,\ldots,n_{\text{ip}}\}} D_\lambda(p_i, p_{i+1})$ vs time. In the second plot we show the arc length distances between the three sensing agents.

2) Slowly time-varying boundary: In this simulation we used $n_{\text{a}} = 3$ sensing agents to have an approximation of the non-convex boundary $\partial Q(t)$ described by:

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

with $\theta \in [0, 1)$, $t_f = 200$ seconds as shown in Figure 3. The values of v_0, k, $v_{\min}$, $v_{\max}$ and λ 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 3 correspond to the positions of the interpolation points and the sensing 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_{NOW^i} coincide with the positions of the sensing 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 3 it is clear that the sensing agents can adapt as ∂Q changes.

The pseudo-distance D_λ is well defined only if the interpolation points belong to the boundary ∂Q. Since the boundary changes with time, the interpolation points are only for some time on the boundary after a sensing agents has projected them. So, we consider as pseudo-distance between any two consecutive interpolation points in a certain time τ the pseudo-distance between their radial projection

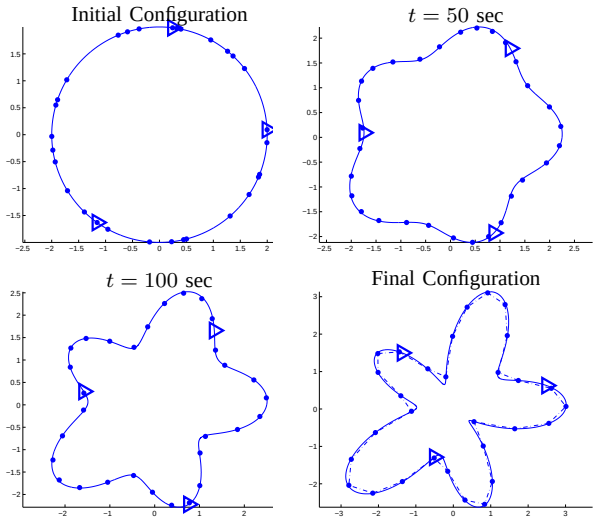


Fig. 3. This figure shows four different instants of the 200 seconds simulation obtained by implementing the ESTIMATE UPDATE AND PURSUIT ALGORITHM with $n_{\mathrm{a}} = 3$, $n_{\mathrm{ip}} = 30$, $v_0 = 1$, $k = 0.05$, $\lambda = \frac{10}{11}$. The boundary ∂Q is slowly time-varying in this case. The sensing agents positions are represented by triangles and initialized to be on the boundary ∂Q. The last frame also shows the approximating polygon.

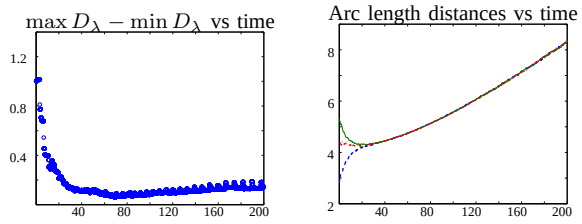


Fig. 4. ESTIMATE UPDATE AND PURSUIT ALGORITHM. This figure refers to the case of ∂Q being slowly time-varying. In the first plot from the right we shown the error $\max_{i\in\{1,\ldots,n_{\mathrm{ip}}\}} D_\lambda(p_i, p_{i+1}) - \min_{i\in\{1,\ldots,n_{\mathrm{ip}}\}} D_\lambda(p_i, p_{i+1})$ vs time. The second plot shows the arc length distances between the three sensing agents.

onto $\partial Q(\tau)$. The disagreement in the placement of the interpolation points, where D_λ is redefined as just explained, is shown in the first plot of Figure 4.

The arc length between any two consecutive sensing agents is shown in the second plot of Figure 4. The three distances increase with time because $L(\partial Q)$, the total length of the boundary, increases with time.

IV. CONCLUSIONS

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 the boundary in such a way as to obtain an approximating polygon with some optimality features.

The mobile agents are equipped with sensors that provide local information on the tangent and curvature of the boundary. The algorithm allows the robots to place a set of interpolation points uniformly spaced according to the estimate of the pseudo-distance D_λ. The position of the interpolation points is stored in a data fusion center and is available on-demand to the agents. The vertices of the approximating polygon are the interpolation point positions. The algorithm is proven to converge even if the boundary is slowly-moving. Tools from consensus analysis allow us to prove the correctness of the algorithm. The existence of a central data fusion center is not a critical ingredient in the design of the algorithm. Indeed, one can envision the following equivalent scenario: the agents communicate the updated interpolation points to their clockwise neighbor, instead of exchanging them with the data fusion center. In such a distributed setting, a stationary user could reconstruct the approximating polygon by communicating to all the agents as they pass by a fixed spatial location. Future research will explore this idea more in detail.

ACKNOWLEDGMENTS

This material is based upon work supported in part by ONR YIP Award N00014-03-1-0512 and NSF SENSORS Award IIS-0330008.

REFERENCES

[1] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Proc. of the Conference on Cooperative Control and Optimization*, (Gainesville, FL), Dec. 2002.

[2] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: asynchronous communication and physical scales," in *Proceedings of the 2003 Block Island Workshop on 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, New York: Springer Verlag, 2004.

[3] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *American Control Conference*, (Portland, OR), pp. 3500–3505, June 2005.

[4] D. W. Casbeer, S.-M. Li, R. W. Beard, R. K. Mehra, and T. W. McLain, "Forest fire monitoring with multiple small UAVs," in *American Control Conference*, (Portland, OR), pp. 3530–3535, June 2005.

[5] 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*, 2005. To appear.

[6] 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.

[7] P. M. Gruber, "Aspect of approximation of convex bodies," in *Handbook of Convex Geometry* (P. M. Gruber and J. M. Willis, eds.), vol. A, pp. 319–345, Oxford, UK: Elsevier, 1993.

[8] 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.

[9] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[10] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *IEEE Conf. on Decision and Control*, (Seville, Spain), pp. 2996–3000, Dec. 2005.

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

[12] 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.

[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.