Noname manuscript No.
(will be inserted by the editor)

Estimating Boundary Dynamics Using Robotic Sensor Networks with Pointwise Measurements

David Saldaña · Renato Assunção · M. Ani Hsieh · Mario F. M. Campos · Vijay Kumar

Received: / Accepted:

Abstract In this paper, we consider environmental boundaries that can be represented by a time-varying closed curve. We use n robots equipped with location sensors to sample the dynamic boundary. The main difficulty during the prediction process is that only n boundary points can be observed at each time step. Our approach combines finite Fourier series for shape-estimation and polynomial fitting for point tracking in time. This combination gives a continuous parametric function that describes the boundary shape and its dynamics. We validate our strategy in simulation and with experiments using actual robots. We tested on non-convex boundaries assuming noisy measurements and inaccurate motion actuators.

Keywords Boundary prediction · Decentralized estimation · Multi-Robot Systems

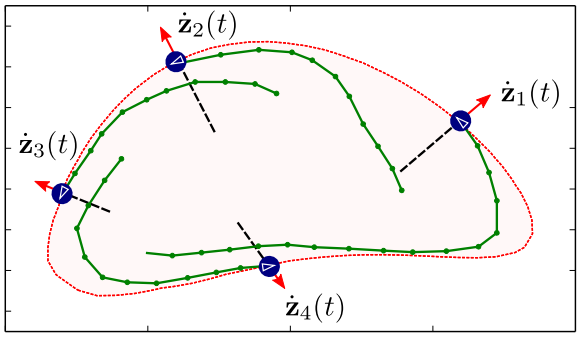


Fig. 1 Enclosing a boundary with four robots. The red dotted curve represents the recent shape of the boundary. The blue disks and their associated green lines represent the robots on the curve and their paths, respectively. The green points represent the pointwise samples. The arrows represent the velocity vector $\dot{\mathbf{z}}_i(t)$.

1 Introduction

Tracking and predicting the behavior of hazardous contaminants is a critical task to prevent catastrophes in natural or urban environments. A few examples are oil spills (Clark and Fierro, 2005; Fahad et al, 2015), forest fires (Casbeer et al, 2005, 2006), harmful algae blooms (Pettersson and Pozdnyakov, 2012), and radiation leaks (Bruemmer et al, 2002). These and other similar occurrences possess a common characteristic, which is a perimeter that borders the affected region.

The affected region can exhibit different behaviors, such as expansion, contraction, translation, and rotation. All these behaviors may happen at the same time generating anisotropic changes that depend on multiple external environmental factors.

Depending on the type, a phenomenon can be measured by a single sensor such as those aboard a satellite (Smith, 1997). However, there are other instances where a global observation is not possible, and only local sensors may be available to observe the phenomenon, e.g., temperature, radiation, salinity. Being able to detect the contaminant and to measure its location point, a mobile robot can identify and track the anomalous region in the environment. The main difficulty in this scenario is that only a finite number of points can be sam-

The authors acknowledge the support of ARL grant DCIST CRA W911NF-17-2-0181, NSF Grants CNS-1446592, and CNS-1521617, ARO grant W911NF-13-1-0350, and C-BRIC.

D. Saldaña is with the Intelligent and Autonomous Robotics Laboratory at Lehigh University, PA, USA. E-mail: saldana@lehigh.edu. R. Assunção, and M.F.M. Campos are with the Computer Science Department, Universidade Federal de Minas Gerais, MG, Brazil. E-mail: {assuncao, mario}@dcc.ufmg.br. M. Ani Hsieh, and V. Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA. E-mail: {m.hsieh, kumar}@seas.upenn.edu.

pled at each time step. These points only describe the location of the boundary in the environment, but they do not include their change in time. We illustrate an environmental boundary tracked by four robots in Figure 1. Anticipating the dynamics of these phenomena is a potentially life-saving indicator to support evacuations or to dispel the hazard.

1.1 Related Works

We present a literature review of the most relevant works that investigated this problem in the areas of robotic sensor networks, multi-robot systems, and related areas. We propose to cluster the most relevant works on monitoring environmental boundaries into categories according to four main facets of the problem.

1.1.1 Finding boundaries in the environment

When a team of robots starts exploring the environment, they persistently look for contaminated or affected regions. These robots can execute different boundary-search algorithms such as random walk (Bruemmer et al, 2002), spiral search (Clark and Fierro, 2005), cooperative exploration (Saldaña et al, 2015), they may follow an environmental function (Marthaler and Bertozzi, 2003a), or follow a gradient (Hsieh et al, 2008). Bruemmer et al (2002) proposes a bio-inspired algorithm using a robotic swarm to detect gradient-free chemical contamination areas. This algorithm is based on potential fields that allow the robots to search for multiple static boundaries in a finite area. Srinivasan et al (2008) propose an algorithm to move the robots toward a contour or a level curve in a scalar field. In (Cortés, 2012), the authors propose a cooperative algorithm to detect boundaries that surround areas with rapid changes based on statistical estimation. A hybrid approach to search and track the boundary with multiple robots is presented in (Clark and Fierro, 2005; Cruz et al, 2007). A simplified hybrid hierarchical control technique is proposed in (Zhang et al, 2013). In previous work, we proposed a coordination method to detect and to track multiple dynamic boundaries (Saldaña et al, 2015). Matveev et al (2015) extend the previous works for non-holonomic Dubins-car-like robots.

1.1.2 Tracking static and dynamic boundaries

When the robots are already on the boundary, their main task is to follow a static or time-varying boundary accurately. In most of the tracking approaches, the robots are driven to circumnavigate the boundary in a counterclockwise manner (Kemp et al, 2004; Marthaler and Bertozzi, 2003b; Bertozzi et al, 2005). Some works use gradient information to control the tracking action (Kemp et al, 2004; Marthaler and Bertozzi, 2003b; Zhang and Leonard, 2010), while other works use gradient-free approaches. Especially in the gradient-free case, each robot can only sense if it is inside or outside the contaminated area (Barat and Rendas, 2003; Casbeer et al, 2005; Joshi et al, 2009). The most common and straightforward approach is implemented by the bang-bang algorithm (Kemp et al, 2004; Hsieh et al, 2005; Marthaler and Bertozzi, 2003b; Joshi et al, 2009) in which a robot keeps persistently switching steering angles to change directions while circulating the perimeter. Additionally, a desired objective in the tracking task is to distribute the robots evenly along with the boundary shape (Kemp et al, 2004; Marthaler and Bertozzi, 2003b). In this way, the sampling points will be evenly distributed to improve the estimation task. The specific case of tracking static boundaries is studied in (Hsieh et al, 2005; Marthaler and Bertozzi, 2003b).

Barat and Rendas (2003) presented an experimental work to track multiple benthic zones using autonomous underwater vehicles. In (Zhang and Leonard, 2005), a cooperative method is proposed to track level curves using the gradient information and to minimize the square error between the robot location and the level curve. The robots move in a pre-defined configuration and cooperatively estimate the gradient of the centroid by combining their measurements at each time step. Menon et al (2014) presents a method to track a dynamic boundary with a single robot using a suboptimal sliding mode algorithm. The same authors also presented an extension of the tracking algorithm for three-dimensional boundaries in (Menon and Ghose, 2013). Matveev et al (2017) proposes a multi-target tracking approach for a single robot model where a conglomerated set of targets is modeled as a dynamic environmental frontier. In (Casbeer et al, 2005, 2006), the authors propose an approach for monitoring the fringe of a forest fire. Instead of sampling pointwise data, this work uses the information from an infrared camera to sample sections of the boundary. As a result, the aerial robots can observe and approximate local sections of the boundary and perform smooth trajectories during the tracking process.

Recent work by Li et al (2014) proposes a control method to track dynamic plumes (pollutants released at a point source) based on the advection-diffusion model. Their robot controller is analytically constructed with proved convergence for chemicals or liquid substances poured in a marine environment. The method of Turgeman and Werner (2017) allocates robots to track the boundary and to find the plume source simultaneously.

In this type of scenario, there is an interest in deploying robust robots in real environments (Mellucci et al, 2017).

1.1.3 Estimating the boundary shape

Once the robots are sampling the boundary using a tracking algorithm, an important task is to estimate the boundary shape using the collected pointwise measurements. Usually, these pointwise measurements come from the location sensor of the robot, i.e., a GPS sensor. One of the main problems of mapping a time-varying boundary is that only a small number of boundary points can be sampled at each time step. This is due to the number of mobile sensors is finite, and for this reason, mapping a dynamic boundary as a closed curve remains a challenge. Kemp et al (2004) made the first attempt to solve this problem. They represent the boundary curve with n points (where n is the number of robots) and use a *snake*-based algorithm (a well-known technique in computer vision) to drive the robots towards the perimeter in a distributed manner. In (Susca et al, 2008, 2009), approximation theory of convex bodies is used to estimate slow-moving boundaries with a polygon with a fixed number of vertices. This algorithm is said to be provably convergent by increasing the number of interpolation points.

1.1.4 Predicting the boundary behavior

A relevant task, in the aforementioned scenarios, is not only estimating the boundary shape but also predicting how the boundary changes in time. Jin and Bertozzi (2007) assume that an ellipse can approximate the boundary shape. In their centralized method, the team of robots acts as a single observer that estimates the boundary dynamics using a Bayesian filter. In a posterior work, the robustness of the technique is shown by implementation with actual robots (Joshi et al, 2009). However, elliptical shapes offer a very limited range of applications in the face of the complexity of natural boundaries. Another centralized method was proposed by White et al (2007). Their method uses multiple aerial vehicles to sample clouds at a fixed altitude. In a planar representation, the sampled points are used to define line segments of constant curvature, known as splinegons. Then, using the velocity of the measured points in the cloud and the cloud motion model, the system is able to generate a linear prediction of the boundary. This linear prediction is a feasible solution only for slow-moving boundaries or predictions for short-time intervals. The work of Duttagupta et al (2011) predicts the behavior of non-closed curves by using range sensors. Using this type of sensors, robots are able to sample multiple points of the boundary at the same time. However, they only work for some specific scenarios *e.g.* it is not suitable for salinity or temperature. The method due to Nagarathna et al (2014); Nagarathna and S. (2015) takes into account the communication issues for reporting the boundary estimation as a sink. Neighboring sensors communicate and exploit the spatio-temporal correlation using the parameters of the contour. As a result, the time to push the sampled data to the sink is predicted. The contour is modeled as a non-closed curve with drifted Brownian motion. They consider measurement errors in detecting the boundary, and the curve estimation is obtained by interpolating the filtered sampled points.

In a previous work (Saldaña et al, 2016), we proposed a method to predict the behavior of environmental boundaries without prior knowledge of the dynamics of the boundary using a single robot and its location measurements. Extending this centralized approach to multiple robots brings challenging issues such as: *i)* design a coordination method to distribute the robots along the dynamic curve; *ii)* integrate the sampled information from all robots without a central approach; *iii)* parameterize the curve without a global view of the boundary and the samples. In a centralized approach, all robots might send all their measurements to a central server. However, this would saturate the communication network and limit the scalability of the multi-robot system.

1.2 Contribution

The present work is related to the last three categories (tracking, estimating, and predicting) and is specifically focused on predicting dynamic boundaries. A preliminary portion of this manuscript was presented in (Saldaña et al, 2017). We present an extended analysis and performance evaluation with simulations and experiments with actual robots.

The main contributions of this work are *i)* We present a multi-robot controller to sample a dynamic boundary efficiently. We maximize the frequency that every point on the curve is visited by a robot. *ii)* We propose a mathematical framework to represent and estimate dynamic curves in the planar space. Our representation is based on the combination of polynomials and Fourier series that incorporates the variation in time. In contrast to related works, our method estimates not only the shape of the boundary but also its behavior. *iii)* We present a coordination method for collective estimation without requiring a central agent. Our decentralized approach enables robots to integrate their estimate using

a communication network in a ring topology. We highlight that our method does not require a dynamic model of the phenomenon.

2 Problem Statement

We are interested in estimating the behavior of a time-varying region in a planar environment, which is bounded by a well-delineated perimeter. A set of mobile sensors with motion and communication capabilities forms the robotic sensor network. Each robotic sensor can measure if it is either inside or outside the region of interest at its location. We study how to sample the boundary in order to estimate its dynamics.

2.1 Dynamic boundary

This region of interest is a connected set $\Omega_t \subset \mathbb{R}^2$ with finite area, indexed by time $t \in \mathbb{R}_{\geq 0}$, and enclosed by a boundary defined as

Definition 1 A *dynamic boundary* is a set of planar points $\partial\Omega_t$ such that for every point $\boldsymbol{z} \in \partial\Omega_t$, and any real value $\xi > 0$, the open disc centered at point $\boldsymbol{z}$ with radius ξ contains points in Ω_t and its complement $\Omega_t^{\complement}$.

The boundary $\partial\Omega_t$ can be modeled by a boundary function such that $\partial\Omega_t = \{\boldsymbol{\gamma}(t,s) \mid s \in [0,1]\}$.

Definition 2 A *boundary function* $\boldsymbol{\gamma} : \mathbb{R}_{\geq t_0} \times [0,1] \to \mathbb{R}^2$ describes a simple closed curve, mapped by the parameter $s \in [0,1)$, such that $\boldsymbol{\gamma}(t,0) = \boldsymbol{\gamma}(t,1)$ and the restriction that $\boldsymbol{\gamma}(t,s)$ is an injective function of $s \in [0,1)$ for a fixed time t. Hence, $\boldsymbol{\gamma}$ describes a continuous curve with no self-intersecting points.

The unitarian tangent vector, at any point, is

$$\mathbf{T}(t,s) = \frac{\partial\boldsymbol{\gamma}(t,s)/\partial s}{\|\partial\boldsymbol{\gamma}(t,s)/\partial s\|}, \tag{1}$$

where $\|.\|$ denotes the Euclidean norm of the vector. We develop our tracking and estimation method based on the following assumptions about the boundary.

Assumption 1 (Smooth boundary). — *The boundary function $\boldsymbol{\gamma}(t,s)$ changes smoothly with respect to the curve parameter s and time t, i.e. its first and second derivatives exist and are continuous.*

Since actual robots have speed limitations, tracking a completely arbitrary boundary dynamics is not always feasible. For example, the robot is not able to track a boundary that moves with ever-increasing speed. Therefore, we assume the following.

Assumption 2 (Bounded motion). — *The magnitude of the velocity of any point on the boundary, described by $\boldsymbol{\gamma}$, is upper bounded by*

$$\left\|\frac{\partial\boldsymbol{\gamma}(t,s)}{\partial t}\right\| \leq \epsilon_v. \tag{2}$$

2.2 Robotic sensor network

The robotic sensor network is composed of n robotic sensors, also called robots in this paper, that are initially distributed along the boundary. We constrain their motion to the perimeter of the dynamic boundary $\partial\Omega_t$. The location of ith robot along the curve at time t can be described by an ever-increasing curve parameter $s_i(t)$. The location of the robot $s_i(t)$ can be mapped to the plain using the boundary function $\boldsymbol{\gamma}(t, s_i(t))$. In our robot configuration, all robots have a cyclic counterclockwise identification order along the curve. It means that the i-th robot is behind its successor (robot $i+1$) and after its predecessor (robot $i-1$), i.e., $s_{i-1}(t) < s_i(t) < s_{i+1}(t)$.

As it was aforementioned, tracking controllers for dynamic boundaries have been widely studied in the literature (Kemp et al, 2004; Casbeer et al, 2005; Joshi et al, 2009). Using local sensing and one of these tracking controllers (e.g., a bang-bang controller), we assume that the robots always move on the curve. In this way, we denote the robot location in the Euclidean space $\mathbb{R}^2$ and its first order dynamics is given by

$$\begin{aligned} \mathbf{x}_i(t) &= \boldsymbol{\gamma}(t, s_i(t)) \\ \dot{\mathbf{x}}_i(t) &= \frac{d\boldsymbol{\gamma}(t, s_i(t))}{dt}. \end{aligned} \tag{3}$$

Each robot is always circulating around the boundary in a counterclockwise manner, i.e., $\dot{s}_i(t) > 0$, and we can control its speed

$$\|\dot{\mathbf{x}}_i(t)\| = u_i(t). \tag{4}$$

We say that a boundary is slow-moving if we count on sufficiently rapid robots. This leads us to the following assumption.

Assumption 3 (Slow-moving boundary). — *The robots move much faster than the boundary, i.e.*

$$\|\dot{\mathbf{x}}_i(t)\| \gg \left\|\frac{\partial\boldsymbol{\gamma}(t,s)}{\partial t}\right\|.$$

We can also relax this assumption taking into account the number of robots as

$$\|\dot{\mathbf{x}}_i(t)\| \gg \frac{1}{n}\left\|\frac{\partial\boldsymbol{\gamma}(t,s)}{\partial t}\right\|.$$

2.3 Available Information

Each robot i is equipped with sensors to obtain local information in discrete time. At each time step t_k, $k \in \mathbb{N}$, assuming that the robots always move on the curve, each robot i is able to measure one point of the curve and the velocity of the point, i.e.,

$$\mathbf{z}_i(t_k) = \boldsymbol{\gamma}(t, s_i(t_k)),$$
$$\dot{\mathbf{z}}_i(t_k) = \frac{\partial \boldsymbol{\gamma}(t, s_i(t_k))}{\partial t}.$$

We note that $\dot{\mathbf{z}}_i(t_k)$ is different from $\dot{\mathbf{x}}_i(t_k)$ in (3), since the first describes the velocity of the point, which can be arbitrary depending on the phenomenon, and the second describes the velocity of the robot moving around the curve. We illustrate a scenario with four robots and their sample vectors in Figure 1. The vector $\mathbf{T}_i$ is not always perpendicular to $\dot{\mathbf{z}}_i$ because $\mathbf{T}_i$ depends on the shape of the curve and $\dot{\mathbf{z}}_i$ depends on the motion of the curve. However, in relative slow-moving boundaries, it is also possible to use the vector perpendicular to the robot velocity vector $\dot{\mathbf{x}}_i(t_k)$, for cases where the vector $\dot{\mathbf{z}}_i(t_k)$ is not measurable. In a previous work (Saldaña et al, 2016), we presented this approximation with satisfactory results.

The robots can exchange messages using a ring topology. This communication topology offers a natural way to maintain network connectivity and to reduce the number of hops when the robots are distributed along a closed curve. In this topology, each robot i can communicate with its successor $i+1$ and its predecessor $i-1$, where $n+1 = 1$.

2.4 Objective

Our objective is to use a team of robots to predict the behavior of a time-varying boundary. This led us to two main problems to be solved. Initially, we want to track the behavior of every point in the curve.

Problem 1 Design a multi-robot controller to distribute the robots along the curve and to efficiently sample the boundary.

In Section 3, we present a sampling strategy for Problem 1. Using this strategy, we can assign a curve parameter s_j to each sample $(t_j, \mathbf{z}_i(t_j))$. In order to improve the estimation, we describe a controller to efficiently sample the boundary in Section 4. After sampling, each robot i has a dataset $\mathcal{D}_i = \{(t_j, s_j, \mathbf{z}_i(t_j)) | j = 1, ..., k\}$ based on its sample points. We want to integrate all the datasets to estimate the dynamics of the curve.

Problem 2 Given a distributed dataset $\mathcal{D} = \{\mathcal{D}_i | i = 1, ..., n\}$ and local communication, estimate the function $\boldsymbol{\gamma}(t, s)$, which describes the dynamics of the environmental boundary.

We present an estimation method for Problem 2 in Section 5. Initially, a centralized approach is described, and then, a coordination method to integrate all the distributed datasets.

3 Sampling the Boundary

In order to track the motion of every point on the closed-curve, we propose a method to parameterize the curve by using the measured information and local communication. This method is composed of three stages. We initially introduce them in a centralized manner. Then, we describe a way to decentralize this process using the ring communication topology.

3.1 Enclosing the boundary

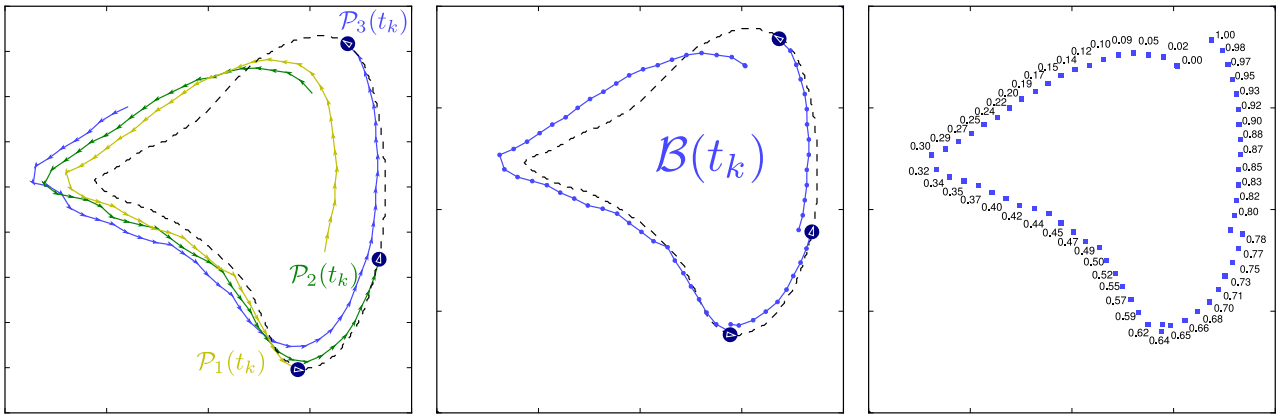
In the initial state, robots move along the curve in counterclockwise direction by $u_i = \Upsilon$, where $\Upsilon > 0$ is a desired constant speed. The boundary is enclosed when each robot i projects its vector $\dot{\mathbf{z}}_i(t)$ and this vector crosses the path or robot $i+1$. We illustrate the enclosing stage in Figure 1. We can see four robot paths intersect with the projection of vectors $\dot{\mathbf{z}}_i(t)$, for all $i = 1, ..., n$.

3.2 Initial parameterization of the curve

Once the boundary is enclosed, we proceed to assign a parameter value to each sample point. One way to enclose a boundary is by using polylines (Saldaña et al, 2015). A *polyline* defines a continuous line based on straight line segments, and it is composed by a set of sorted points. The path of robot i is modeled as a polyline $\mathcal{P}_i(t_k) = \{\mathbf{z}_i(t) | t = t_1, ..., t_k\}$ with sorted points by decreasing time values. Figure 2(a) illustrates three different robot paths after enclosing the boundary.

At time t_k, the robots enclose the boundary, and each robot i has a polyline $\mathcal{P}_i(t_k)$. We take the last sample point of the polyline $\mathbf{z}_{i-1}(t_k) \in \mathcal{P}_{i-1}(t_k)$ and trace a line that passes this point with direction $\dot{\mathbf{z}}_{i-1}(t)$. Since we assumed that the boundary is slow-moving, part of the polyline $\mathcal{P}_i(t_k)$ is close to $\mathcal{P}_{i-1}(t_k)$, and therefore, the line traced from crosses $\mathcal{P}_i(t_k)$ at point $\mathbf{p}_i$. Then, we remove the older points after the intersection and include the intersected point, the resultant polyline is

$$\mathcal{P}'_i(t_k) = \{\mathbf{z}_i(t) | t = t_1, ..., t_l\} \cup \{\mathbf{p}_i\},$$



(a) Initial paths after completing a cycle. The actual boundary and polylines are represented by the dashed line and solid line respectively.

(b) Piecewise boundary, represented by the solid line. The boundary at time t_k is represented by the dashed line.

(c) Parameterization of the sample points (blue dots) by using the piecewise boundary.

Fig. 2 Distributed parameterization of the closed-curve. It is computed in three steps: (*i*) enclosing the boundary as presented in Panel (a), (*ii*) computing a piecewise boundary as presented in Panel (b), and (*iii*) the parameterization of the piecewise boundary based on the arc-length is illustrated in Panel (c).

where l is the last point before the intersection. There are two special cases to take into account: first, if the line intersects multiple points in $\mathcal{P}'_i(t_k)$, we remove all the points after the most recent intersecting point; second, if $\dot{\mathbf{z}}_{i-1}(t) = \mathbf{0}$, it means that the boundary is static at that point, then the point $\mathbf{z}_{i-1}(t_k)$ is on the polyline $\mathcal{P}_i(t_k)$ and the intersecting point is $\mathbf{p}_i = \mathbf{z}_{i-1}(t_k)$. Now, we proceed to use all the new computed polylines.

Definition 3 A *piecewise boundary* is a set of n polylines that surrounds a boundary,

$$\mathcal{B}(t_k) := \{\mathcal{P}'_1(t_k), ..., \mathcal{P}'_n(t_k)\}.$$

Each of these polylines comes from a robot path, and they can be connected to form a closed loop by projecting the velocity vector $\dot{\mathbf{z}}$ of the first point of each polyline.

Figure 2(b) illustrates the resultant piecewise boundary after processing the paths from Figure 2(a). We highlight that the piecewise boundary is a fundamental part of our method because it is used for the robot distribution and the boundary estimation. In the next sections, we will describe our approach to control the robots in order to increase the updating frequency of every point in the curve.

We now proceed to parameterize the piecewise boundary $\mathcal{B}(t_k)$. The length of the polyline, associated with robot i, is denoted by $\ell_i := \text{length}(\mathcal{P}'_i(t_k))$. The total length of the piecewise boundary is $\ell = \sum_{i=1}^{n} \ell_i$. The number of points in a polyline is denoted by $|\mathcal{P}'_i(t_k))|$. We can assign a parameter $s \in [0, 1]$ to each point of the piecewise boundary based on the arc-length. Therefore, the parameter of the jth sample point of the ith robot is

$$s_{ij} = \frac{1}{\ell}\Big(\sum_{l=1}^{i-1} \ell_l + \sum_{k=1}^{j-1} \|\mathbf{z}_i(t_{k+1}) - \mathbf{z}_i(t_k)\|\Big). \quad (5)$$

Figure 2(c) shows the parameterization of the piecewise boundary for the aforementioned example.

3.3 Decentralized coordination

In the initial stage, the robot with an identification number equal to n serves as a temporary leader in charge of identifying if the boundary is enclosed, and creating the initial parameterization of the curve. After the initial parameterization, the robots will continue working without the need for a leader and estimate the boundary using only local information and communication with their neighbors in the ring topology.

Robots can check if they enclose the boundary if the projection of the vector $\dot{\mathbf{z}}_i(t)$ crosses the path of the robot $i+1$. Following the ring communication topology, each robot i shares its sample $(t, \mathbf{z}_i(t))$ to the robot $i-1$ at each time step t. Robot $i-1$ can identify the path of robot i using only the received values. The global task of enclosing the boundary is completed when all the local tasks are completed. The robot team can check the achievement of the global task by periodically exchanging messages. Each robot shares with its neighbors a boolean flag representing whether its local task has been completed or not. When it receives the flags from its neighbors, it applies a logic *and* comparator to its flag, and the received flags. Then it shares the new

resulting flag. After exchanging n messages, all robots will know whether the task was completed or not. This checking process is periodically repeated until all the robots complete their own local tasks, and as a consequence, all flags will be *true*.

The robot team can also coordinate to parameterize the curve. Every robot is able to locally compute $\{\mathcal{P}'_i(t_k)\}$. Robot n sends $\{\mathcal{P}'_n(t_k)\}$ to robot $n-1$. When robot $n-1$ sends $\{\mathcal{P}'_n(t_k), \mathcal{P}_{n-1}(t_k)\}$ to robot $n-2$. The same process is repeated n times through the network until the temporal leader receives the set $\{\mathcal{P}'_n(t_k), ..., \mathcal{P}'_1(t_k)\}$. Then, the leader applies the parameterization using (5) and shares the result.

4 Distributing the robots along the curve

Our objective is to control the robots in order to visit every point in the curve as frequently as possible. By design, the robots move in a counterclockwise manner, and we control the speed of the robots (from (4)). If the robots circulate with speed Υ around a slow-moving boundary, our problem becomes a distribution problem. Since the boundary is a closed-loop, every point on the boundary is going to be visited periodically by each of the robots. If the robots move with constant speed, the longest time that a point is going to be visited is proportional to the longest distance between a pair of neighbor robots. Therefore, we minimize the longest distance between every pair of robots by evenly distributing them along the curve. In this way, in a static boundary, every point will be visited with frequency

$$F = \frac{\Upsilon}{s_{i+1} - s_{i-1}} = \frac{n\Upsilon}{\ell}.$$

Franchi et al (2010) designed a control law to move and distribute a robot team around a circular shape. Since a simple closed curve is topologically the same as a circle, we can apply a similar approach to evenly distribute the robots along the boundary based on the arc-length parameterization. We extend the results from Franchi et al. for slow-moving closed curves.

In order to ensure the robots are evenly distributed along the curve, we apply the following control law for the i-th robot,

$$u_i = \Upsilon + K(\ell_{i+1} - \ell_i), \tag{6}$$

where $K > 0$ is a gain constant. It means that we want each length of the updated paths to be the same, $\ell_i \to \bar{\ell}_i$, where $\bar{\ell}_i = (\ell_{i+1} + \ell_{i-1})/2$. Note that using (6), the ith robot only requires the length of the polyline of the robot $i+1$ and the length of its own polyline. Those polylines are computed by simply connecting the sample points. A prediction of the boundary shape is not required at this stage. This behavior will homogeneously distribute the robots as long as they move on slow-moving boundaries (see Assumption 3).

Proposition 1 *If the boundary is slow-moving, and each robot follows the control rule* (6)*, the robots asymptotically converge to an equidistant distribution along the curve (by arc-lenght). Additionally, the robots will keep moving with constant speed* Υ*.*

Proof. Initially, we re-parameterize the curve by arc-length. The arc-length parameter is given by

$$r(s) = \int_0^s \left\| \frac{\partial \boldsymbol{\gamma}(t,u)}{\partial u} \right\| du.$$

Since $\|\partial r/\partial s\| > 0$ for all s, it has a differentiable inverse function $s = s(r)$, we can consider a new curve function

$$\boldsymbol{\alpha}(t,r) = \boldsymbol{\gamma}(t, s(r)),$$

where the parameter r is bounded by $0 \le r \le L(t)$, and $L(t)$ is the arc-length of the curve at time t. The location of the ith robot in the curve $\boldsymbol{\alpha}(t,r)$ is denoted by $r_i(t)$.

We can decompose the robot dynamics from (3) by using the chain rule in the new parameterization,

$$\begin{aligned}\dot{\mathbf{x}}_i(t) &= \frac{\partial \boldsymbol{\alpha}(t, r_i(t))}{\partial t} + \frac{\partial \boldsymbol{\alpha}(t, r_i(t))}{\partial r}\dot{r}_i(t) \\ &= \frac{\partial \boldsymbol{\alpha}(t, r_i(t))}{\partial t} + \dot{r}_i(t)\mathbf{T}_i.\end{aligned}$$

Then, its magnitude satisfies,

$$\|\dot{\mathbf{x}}_i(t)\| \le \left\| \frac{\partial \boldsymbol{\alpha}(t, r_i(t))}{\partial t} \right\| + \dot{r}_i(t)\|\mathbf{T}_i\|.$$

Since the curve is parameterized by arc-lenght, the tangent vector is $\|\mathbf{T}_i\| = 1$, and based on (4), we obtain

$$u_i(t) \le \left\| \frac{\partial \boldsymbol{\alpha}(t, r_i(t))}{\partial t} \right\| + \dot{r}_i(t).$$

Based on $\frac{\partial \boldsymbol{\alpha}}{\partial t} = \frac{\partial \boldsymbol{\gamma}}{\partial t}$ and (2),

$$u_i(t) \le \epsilon_v + \dot{r}_i(t). \tag{7}$$

The path of the robot from the last point that intersected the previous robot at time t_{i-1} and the current position is

$$\begin{aligned}\ell_i(t) &= \int_{t_{i-1}}^{t} \|\dot{\mathbf{x}}_i(t)\| dt \\ &\le \int_{t_{i-1}}^{t} (\epsilon_v + \dot{r}_i(t)) dt \\ &\le \epsilon_v(t - t_{i-1}) + r_i(t) - r_i(t_{i-1}).\end{aligned}$$

Since robot $i-1$ intersects the path at location $r_{i-1}(t)$, the location of the robot is $r_{i-1}(t) = r_i(t_{i-1})$. Based on Assumption 3, if we increase the speed of the robot then $u_i(t)$ approximates to $\dot{r}_i(t)$ in (7). Therefore, for the quasi-static case, we can say that the updated path of the ith robot is approximately equal to a section of the curve, i.e., $\ell_i = r_i - r_{i-1}$. The controller from (6) can be expressed as,

$$u_i = \Upsilon + K(r_{i+1} + r_{i-1} - 2r_i). \tag{8}$$

We rewrite (8) for all robots in a compact form

$$\dot{\mathbf{r}} = \Upsilon\, \mathbf{1} + K\mathbf{A}\mathbf{r},$$

where $\mathbf{A}$ is a circular matrix with its first row equal to [−2 1 0 ... 0 1], with $\mathbf{1} = [1, ..., 1]^T$, and with $\mathbf{r} = [r_1, ..., r_n]^T$ as a vector of robot locations.

We say that robot i is equidistant to robot $i+1$ and robot $i-1$, if robot i is in the middle of the other two robots. That is $\bar{r}_i = r_i$, where $\bar{r}_i = (r_{i+1} + r_{i-1})/2$. Then, the error is given by $e_i = \bar{r}_i - r_i$. In a general form, for all robots,

$$\mathbf{e} = \frac{1}{2}\mathbf{A}\mathbf{r}.$$

The dynamics of the error is

$$\begin{aligned}\dot{\mathbf{e}} &= \frac{1}{2}\mathbf{A}\dot{\mathbf{r}} \\ &= \frac{1}{2}\mathbf{A}(\Upsilon\, \mathbf{1} + K\mathbf{A}\mathbf{r}) \\ &= \frac{\Upsilon}{2}\mathbf{A}\, \mathbf{1} + K\mathbf{A}\mathbf{e} \\ &= K\mathbf{A}\mathbf{e}.\end{aligned}$$

Since the matrix $\mathbf{A}$ is positive semidefinite and its eigenvalues are non-positive, the linear system yields the error $\mathbf{e}$ equals to the average of its initial values $\mathbf{e}(0)$. This average is zero because $\mathbf{1}^T\mathbf{e}(0) = \mathbf{1}^T\mathbf{A}\mathbf{r}(0)/2 = 0$. □

5 Cooperative prediction

In this section, we present a parametric function that describes the dynamics of the boundary, a centralized way to estimate the parameters, and how to integrate distributed information.

5.1 Modeling the curve function

We can approximate the trajectory of a point on the curve in a finite time interval using an n-degree polynomial. The trajectory of a point with a parameter $s_0 \in [0, 1]$ can be approximated by

$$\hat{\boldsymbol{\gamma}}(t, s_0) = \underbrace{\begin{bmatrix}\beta_{00}\ \beta_{01}\ \dots\ \beta_{0n} \\ \beta_{10}\ \beta_{11}\ \dots\ \beta_{1n}\end{bmatrix}}_{\boldsymbol{\beta}(s_0)} \underbrace{\begin{bmatrix}1 \\ t \\ \vdots \\ t^n\end{bmatrix}}_{\mathbf{f}(t)}. \tag{9}$$

Letting $\boldsymbol{\beta}(s_0)$ be the $2 \times (n+1)$ constant matrix for s_0, and $\mathbf{f}(t)$ the polynomal terms that depend on t. Since the curvature changes smoothly, we can generalize this trajectory for every point on the curve by making the matrix $\boldsymbol{\beta}$ a function of $s \in [0, 1]$,

$$\hat{\boldsymbol{\gamma}}(t, s) = \boldsymbol{\beta}(s)\, \mathbf{f}(t). \tag{10}$$

Each entry of $\boldsymbol{\beta}$ can be described by a periodic function with $\boldsymbol{\beta}(0) = \boldsymbol{\beta}(1)$, which is continuous, differentiable, and injective for every $s \in [0, 1]$. Therefore, we can approximate every entry of $\boldsymbol{\beta}$ with the truncated Fourier series as

$$\hat{\beta}_{ij}(s) = a_0^{(ij)} + \sum_{k=1}^{m} a_k^{(ij)}\, \sin(2\pi k s) + \sum_{k=1}^{m} b_k^{(ij)}\, \cos(2\pi k s).$$

We arrange the terms of the Fourier series in a vector $\mathbf{g}(s) = [1, \sin(2\pi s), \dots, \sin(2\pi m s), \cos(2\pi s), \dots, \cos(2\pi m s)]^\top$. We join the Fourier vector $\mathbf{g}(s)$ and the polynomial vector $\mathbf{f}(t)$ by using the Kronecker product as

$$\mathbf{h}(t, s) = \mathbf{f}(t) \otimes \mathbf{g}(s) = [f_1 g_1,\ f_1 g_2, \dots,\ f_{n+1}\, g_{2m+1}]^\top.$$

With some algebraic manipulation (details in (Saldaña et al, 2016)), we can separate the curve function (10) into two parts: a matrix of weights $\mathbf{C}$ with dimension $(n+1)(2m+1) \times 2$, and a vector $\mathbf{h}(t, s)$ as

$$\hat{\boldsymbol{\gamma}}(t, s) = \mathbf{C}^\top\, \mathbf{h}(t, s). \tag{11}$$

Based on this model, we can frame this problem as a linear regression system, where we have to analyze the number of terms that should be used in the function vector $\mathbf{h}$ and attempt to infer the matrix of weights $\mathbf{C}$.

5.2 Estimating the curve function

We use the time t and the curve parameter s as input variables, and the sampled locations $\mathbf{z}_i(t_j) = [x_i(t_j), y_i(t_j)]^\top$ as the output. In this way, we use the trajectory information $\mathcal{D} = \{(t_j, x_j, y_j, s_j) | j = 0, 1, ..., k\}$ to predict the anomaly behavior by attempting to estimate the parameter matrix $\mathbf{C}$ of (11). We model the problem as a linear system with the form

$$\mathbf{X}\,\mathbf{C} = \mathbf{Y}, \tag{12}$$

where $\mathbf{Y}$ is a matrix with dimension $k \times 2$ that contains each output location (x_i, y_i), $i = 1, ..., k$ in the robot's trajectory $\mathcal{D}$; $\mathbf{C}$ is the weighted matrix that we want to estimate; and $\mathbf{X}$ is the design matrix created using the input (t_i, s_i), $i = 1, ..., k$ and the functions of $\mathbf{h}(t_i, s_i)$, defined as

$$\mathbf{X} = \begin{bmatrix} \mathbf{h}(t_1, s_1)^\top \\ \vdots \\ \mathbf{h}(t_k, s_k)^\top \end{bmatrix}.$$

Our objective consists of adjusting the parameters matrix $\mathbf{C}$, from (12), of the model function $\hat{\boldsymbol{\gamma}}$ to best fit the data set $\mathcal{D}$. Assuming that $\mathbf{X}$ is a full-rank matrix, we can estimate $\widetilde{\mathbf{C}}$ by solving

$$\widetilde{\mathbf{C}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}. \tag{13}$$

Therefore, the curve function $\hat{\boldsymbol{\gamma}}(t, s)$ is approximated by

$$\hat{\boldsymbol{\gamma}}(t, s) = \widetilde{\mathbf{C}}^T\, \mathbf{h}(t, s).$$

The matrix $\mathbf{X}$ can be estimated if a robot has the full dataset $\mathcal{D}$ and requires computational power to compute $(\mathbf{X}^\top \mathbf{X})^{-1}$. Then, in the next section, we extend this method to allow the team of robots to estimate the curve in a distributed collaborative way, where the samples are distributed among all robots.

5.3 Online prediction

For each time step, the dataset $\mathcal{D}$ with k samples can be used to estimate the boundary behavior. We include a subscript to the design matrix $\mathbf{X}_k$ to denote the number of samples k. When a robot takes a new sample $(t_{k+1}, x_{k+1}, y_{k+1}, s_{k+1})$, the estimation requires the computation of $(\mathbf{X}_{k+1}^\top \mathbf{X}_{k+1})^{-1}$, which has an $\mathcal{O}(k^3)$ time complexity . Additionally, the robot needs to have all the data in $\mathbf{X}_{k+1}$. Our objective is to use the computed estimation from $\mathbf{X}_k$ and the new sample to compute $\mathbf{X}_{k+1}$. In this way, robots do not need the whole dataset, and they can update the estimation using the new samples and the old estimation.

We extend our estimation method using *Recursive Least Squares* to obtain the updated prediction by taking advantage of the already computed $(\mathbf{X}_k^\top \mathbf{X}_k)^{-1}$ and avoiding to invert the whole matrix at every time step.

In order to make (13) iterative, we rewrite it as the operation with the rows of $\widetilde{\mathbf{C}}$,

$$\widetilde{\mathbf{C}} = \Big(\sum_{i=1}^{k} \mathbf{h}_i \mathbf{h}_i^\top \Big)^{-1} \sum_{i=1}^{k} \mathbf{h}_i \mathbf{y}_i^\top, \tag{14}$$

where $\mathbf{h}_i = \mathbf{h}(t_i, s_i)$ and $\mathbf{y}_i = [x_i, y_i]^\top$. The left sum in the right side of (14) defines the correlation matrix $\mathbf{P}$, which can be recursively calculated,

$$\mathbf{P}(i) = \mathbf{P}(i-1) + \mathbf{h}_i\, \mathbf{h}_i^\top, \tag{15}$$

with $\mathbf{P}(1) = \mathbf{h}_1\, \mathbf{h}_1^\top$, and the *cross-correlation vector* of right sum in (14),

$$\mathbf{q}(i) = \mathbf{q}(i-1) + \mathbf{h}_i\, \mathbf{y}_i^\top, \tag{16}$$

with $\mathbf{q}(1) = \mathbf{h}_1\, \mathbf{y}_1$. Then, (13) is rewritten as the multiplication of two iterative terms,

$$\widetilde{\mathbf{C}} = \mathbf{P}(k)^{-1} \mathbf{q}(k). \tag{17}$$

For every iteration, it is possible to compute the inverse $\mathbf{P}(k)^{-1}$ using the last estimation $\mathbf{P}(k-1)^{-1}$. Assuming that the matrices $\mathbf{P} = \mathbf{P}^\top$ and $(\mathbf{P} + \mathbf{h}\mathbf{h}^\top)$ are invertible, we apply the Sherman-Morrison formula (Sherman and Morrison, 1950) as,

$$\begin{aligned} \mathbf{P}(k)^{-1} &= (\mathbf{P}(k-1) + \mathbf{h}_k \mathbf{h}_k^\top)^{-1} \\ &= \mathbf{P}(k-1)^{-1} - \tfrac{(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)^\top}{1 + \mathbf{h}_k^\top \mathbf{P}(k-1)^{-1}\mathbf{h}_k}. \end{aligned} \tag{18}$$

Based on this mathematical framework, we present two coordination methods to integrate all the distributed information.

5.4 Integrating distributed information

We integrate the distributed sampled point-wise measurements with a *sliding window* approach. Our on-line prediction is based on the latest k sampled points (window), and the robots remove (slide) the older points. Then, each robot keeps in memory its own dataset

$$\mathcal{D}_i = \{(t_j, s_j, x_j, y_j) | j = 1, ..., k\}.$$

We can ask any robot about the boundary prediction. Assuming that we ask robot n, it computes the boundary prediction, represented by the vector $\mathbf{q}_n(t_k)$ and the matrix $\mathbf{P}_n^{-1}(t_k)$, using (16) and (18) respectively. Robot n sends the matrix and the vector to robot $n-1$. When robot $i < n$ receives the matrix $\mathbf{P}_{i+1}^{-1}(t_k)$ and the vector $\mathbf{q}_{i+1}(t_k)$ from robot $i+1$, it proceeds to update

the received matrix and vector with its own k samples using the same equations. Next, robot i sends $\mathbf{P}_i^{-1}(t_k)$ and $\mathbf{q}_i(t_k)$ to robot $i-1$. This iterative process is finally completed when robot n receives the information from robot 1. When this happens, robot n uses the received information to compute the matrix of weights

$$\widetilde{\mathbf{C}} = \mathbf{P}_1(k)^{-1}\mathbf{q}_1(k).$$

Finally, we obtain the curve function

$$\hat{\boldsymbol{\gamma}}_k(t,s) = \widetilde{\mathbf{C}}^T\,\mathbf{h}(t,s),$$

which integrates the distributed information from n robots. We want to emphasize that the estimation is obtained by exchanging n messages, and the size of the message package is fixed. The message package only contains $\mathbf{P}_i^{-1}(t_k)$ and $\mathbf{q}_i(t_k)$, and they are independent of the size of the local dataset k and the number of robots n.

By using this method, every robot keeps its own updated dataset with the latest sampled points. When a robot receives a request for a boundary estimation, it computes a pair $(\mathbf{P}_i^{-1}(t_k), \mathbf{q}_i(t_k))$, representing the collected information for the estimation. Then it makes them circulate around the ring topology in order to iteratively collect the information of all robots.

We highlight that the integration of the distributed information can be started from any robot. This implies that a time step, every robot can request an estimation, and then we can have n estimation pairs circulating around the ring topology.

6 Experiments

We want to validate our method in experiments with actual robots. Using the Lebesgue measure to quantify the area of a set in $\mathbb{R}^2$, denoted by μ, we define a metric to quantify the difference between the actual bounded area Ω_t and our estimation $\hat{\Omega}_t$. Our *estimation error*,

$$\delta(\Omega_t, \hat{\Omega}_t) = \frac{\mu(\Omega_t \setminus \hat{\Omega}_t) + \mu(\hat{\Omega}_t \setminus \Omega_t)}{\mu(\Omega_t)}, \qquad (19)$$

takes into account the non-estimated area $\mu(\Omega_t \setminus \hat{\Omega}_t)$ plus the misestimated area $\mu(\hat{\Omega}_t \setminus \Omega_t)$ (see Figure 3). In order to have a proportion of the error with respect to the actual area, we include the actual boundary area $\mu(\Omega_t)$ in the denominator. Similar metrics to quantify the accuracy in boundary estimation were presented in (Susca et al, 2008; Saldaña et al, 2015, 2016).

In a previous work (Saldaña et al, 2016), we presented a quantitative comparison between our method using a fast single robot and a polygonal approximation

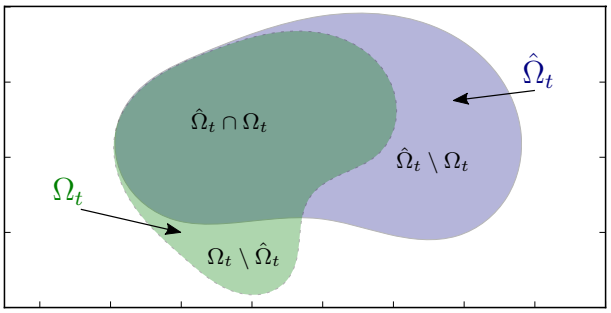


Fig. 3 Illustration of the metric $\delta(\Omega_t, \hat{\Omega}_t)$ for a bounded region Ω_t (green area) and its estimation $\hat{\Omega}_t$ (blue area). This metric quantifies the error in the estimated bounded region, by taking into account the non-estimated region $\Omega_t \setminus \hat{\Omega}_t$ and the incorrectly estimated region $\hat{\Omega}_t \setminus \Omega_t$.

(Susca et al, 2008). In shape estimation, the error of the polygon approximation never goes to zero as the polygon passes over old points. In contrast, our estimation improves by increasing the number of sampling points, and the number of polynomial and Fourier terms.

The experiments in the present paper validate our estimation method for a non-linear changing analytic boundary and a real liquid boundary on a flow.

6.1 Simulations

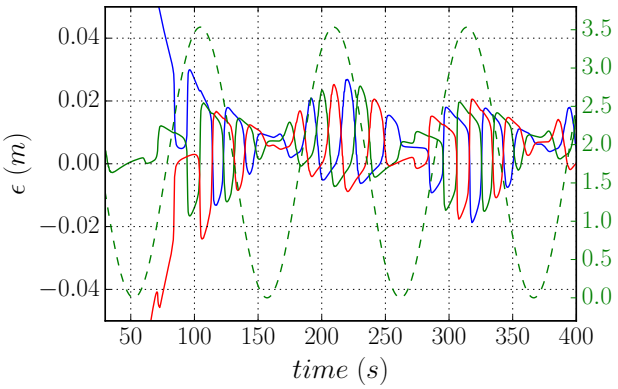
We evaluate the performance of our method with respect to the numbers of robots. Initially, we study how well the robots can maintain their equidistant locations in the curve and how this affects the prediction. The robots circulate an analytic function described by the function

$$\boldsymbol{\gamma}(t,s) = \begin{bmatrix} 50(2 + d\sin(0.03t))\cos(2\pi s) \\ 50\sin(2\pi s) \end{bmatrix},$$

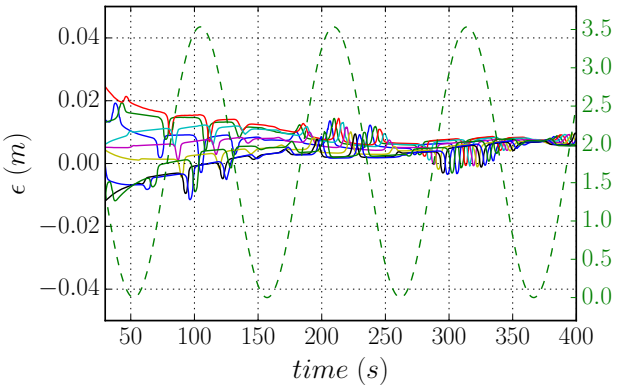
where $d = 0$ if $s \in [1/4, 3/4]$ and $d = 1$ otherwise. This boundary function describes a circle in which one half stays static, and the other half stretches along the x-axis in an oscillatory manner. This non-linear function is challenging since it cannot be modeled by a polynomial, and it combines static and dynamic sections. The perimeter and the internal area of this boundary also describes an oscillatory behavior.

6.1.1 Enclosing

We studied how to enclose the boundary while maintaining the robots at equidistant locations. In this simulation, we evaluated the performance of the enclosing method (using (6)) and the impact of the boundary



(a) $n = 3$ robots.



(b) $n = 9$ robots.

Fig. 4 Consensus error for $n = 3$ and $n = 9$ number of robots respectively. Each colored line represents a different robot. The dashed line represents the total variation of the boundary (from (21)).

variation. Since, we want to keep the robots equidistant on the curve, the *distribution error* is

$$\epsilon_i = \frac{\ell_{i+1} + \ell_{i-1}}{2} - \ell_i. \tag{20}$$

We can describe the *total variation* of the curve by

$$e(t) = \int_0^1 \left\| \frac{\partial \boldsymbol{\gamma}(t,s)}{\partial t} \right\| ds. \tag{21}$$

We can see in Figure 4(a) that the error for $n = 3$ robots is bounded even when the boundary achieves its maximum variation. The dashed line shows the oscillatory behavior of the total variation (from (21)). This graphic shows that the oscillatory behavior is coupled with the distribution error. However, the magnitude of the error is reduced as the number of robots increases (see Figure 4(b)). We can see that once the robots converge, they maintain their errors within a finite interval, and this interval is proportional to the total change of the boundary.

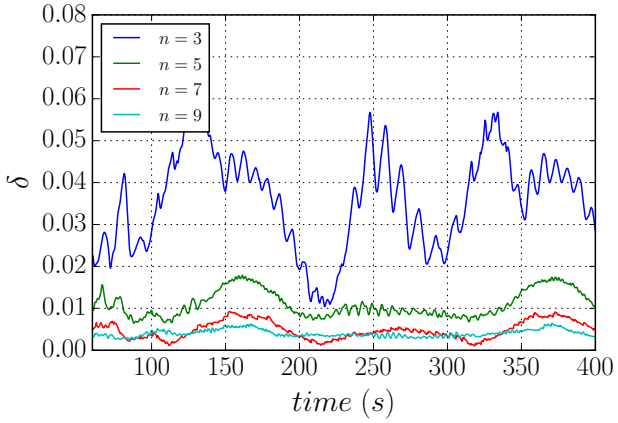


Fig. 5 Prediction error for different number of robots n. Each colored line denotes the prediction error δ in time.

6.1.2 Predicting

As aforementioned, our prediction method is improved if the robots are equidistantly distributed along the boundary. We show that the estimation error (19) depends on the number of robots (see Figure 5.). We can see that the prediction method is also affected by the oscillatory behavior of the boundary as the error function also exhibits a periodic pattern. The predictions are very accurate, even in the worst case for $n = 3$ robots where the error is below 6.0%. The error reduces rapidly as the number of robots increases. For this specific boundary dynamics, seven robots are enough to keep the prediction error smaller than 1.0%.

We also want to analyze the future estimations of the boundary shape. Given a set of parametrized sampled points $\mathcal{D}$, we want to estimate the shape of the boundary during a time interval (including future time). Using the same previous simulated testbed, the prediction error can be seen in Figure 6. In this figure, we present predictions of the curve during a time interval using the same dataset. We used a fixed number of Fourier terms, $M = 10$, and polynomials with degrees $N = 0, 1, 2$. A polynomial degree $N = 0$ represents a constant value and it means that the shape does not change in time. The estimator tries to approximate a fixed boundary that fits all sampled points. We can see that this estimator has the largest error δ most of the time, although occasionally it has a small error. The linear estimator (with $N = 1$) offers a better approximation, and its associated error changes slowly. This is a good estimator since it requires less sampled points and does not ignore the time in the evolving process. The last one uses $N = 2$ and this approximation offers the best estimation, maintaining the error within a small percentage during the observation period. How-

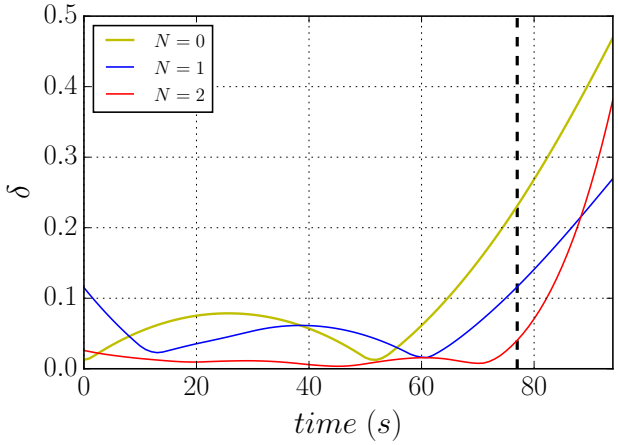


Fig. 6 Prediction error for a fixed dataset. The vertical dashed line represents the last time that a point was sampled. The continuous lines represent the prediction error for three polynomial degrees, $N = 0, 1, 2$.

ever, the error grows faster than the linear estimator for future predictions. This is one of the main problems of estimators based on high-degree polynomials. We also simulated higher-degree polynomials, but they require a larger number of points in the dataset to satisfy the minimum required data for least-squares.

6.2 Experiments with actual robots

In this section, we show that our method can be used with actual robots and real boundaries. We use a team of three robotic boats in a tank in order to track and predict the shape of a liquid. In our testbed, each robot is driven by two propellers in the back as pictured in Figure 7. We can control these robots by using the differential drive model. The location of the robots is obtained using an external motion capture system around the tank. Although we assumed holonomic motion in the problem statement, we can relax this assumption by using these differential robots, which can track the boundary and control their velocity. We implemented the bang-bang algorithm (Marthaler and Bertozzi, 2003b) to keep the robots moving on the boundary.

A small tank ($0.1 \times 0.1 \times 0.02 m^3$) was used to generate and record the boundary. The boundary was created by suspending two liquid dye droplets in a 1:1 glycerol-water mix. The submerged rotating disks are driven such that two adjacent 4×2 sets of disks are controlled by independent stepper motors and controllers. The experimental setup for creating the boundary is shown in Figure 8. We track the boundary of the viscous liquid using an RGB camera and image processing. Then, we take the recorded boundary, remap it to a larger tank, and command the robots to track the recorded time-varying boundary. This allows us to reproduce the experiment with different conditions for the same boundary[1]. The presented boundary is highly non-convex and changes in all directions; it translates, expands, shrinks all at the same time.

The experiments were conducted in the multi-robot Coherent Structure Testbed (mCoSTe) which consists of a $3.0 \times 3.0 \times 0.5\, m^3$ tank and a fleet of autonomous surface vehicles (ASVs). The tank was filled with still water, and we used a team of three ASVs as pictured in Figure 9. In the accompanying video[2], we show an experiment where the three actual robots track and predict a dynamic non-convex boundary. We can see that all boundary points change their locations at different rates and directions. In the tracking process, the ASVs experience difficulties following the boundary because their inertia causes the robots to slide on the water, making it difficult to compensate with its small actu-

[1] The dataset: `https://github.com/dsaldana/boundary-dataset`
[2] The video is available at `https://youtu.be/Zwc6vNuUFDw`

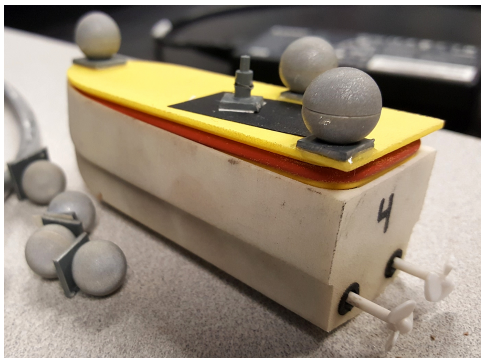


Fig. 7 A robotic boat driven by two propellers in the stern. The reflective markers on the top are used for localization. A sealing system prevents water from entering the boat

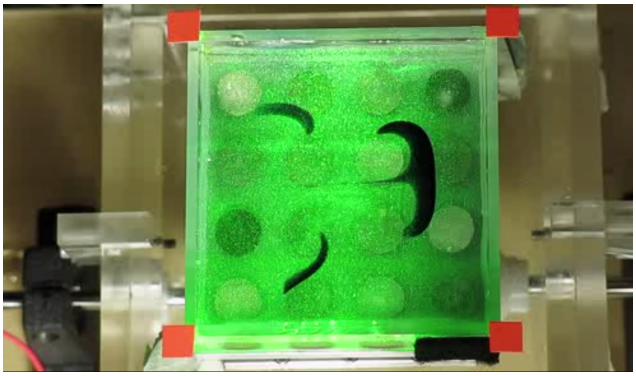


Fig. 8 Experimental setup creating a time-varying boundary. The tank contains two liquid dye droplets in a 1:1 glycerol-water mix. The nonlinear flow field is generated using a 4×4 lattice of submerged counter rotating disks. This image and corresponding video is courtesy of Prof. Peter Yecko.

ators. Figure 10 depicts three snapshots of the experiment. We can see that the robots move outside the boundary and experience some oscillations before resuming proper tracking of the boundary. Even though the boats ASVs were not able to move precisely along the boundary during the tracking process, we obtained estimates that match closely to the real dynamic boundary. We can see on the left side that the robots had difficulties moving through the sections with high curvature (see around the point $(1.2, 1.7)$ in Figure 10(c)). Situations with lower curvature did not result in large estimation errors (see around the point $(2.0, 2.3)$ in Figure 10(b)). The outcome is an estimated boundary that is closer to the robot path than to the actual boundary. Despite this, the estimation is approximately equal to the real boundary, even in the presence of these uncertainties. The estimation error (from (19)) during the experiment is presented in Figure 11. We can see that while the error fluctuates as a result of the oscillatory movements, it is generally under 13%.

Our method offers better estimates as we increase the number of sampled points. In Figure 12, we illustrate how the metric δ is rapidly reduced as we increase the number of sampled points. We can also observe that taking into account a large number of points (> 400) does not considerably reduce the accuracy. Depending on the dynamics of the curve, *e.g.*, boundaries that exhibit oscillatory behaviors, using a large number of sampled points can be a problem if the degree of the polynomial is not increased.

Different from other approaches in the literature where the sample points are used to approximate the boundary (Susca et al, 2008), our method takes into account the motion of the points in time. In Figure 13, we can see the samples of three robots, depicted by the yellow points, our estimation of the boundary, depicted by the dashed line, and the real boundary, depicted by the solid line. We can see that sampled points are off in comparison with the real boundary. At the time, when the last points were sampled, the whole boundary was already shifted and shrunk. If we approximate the

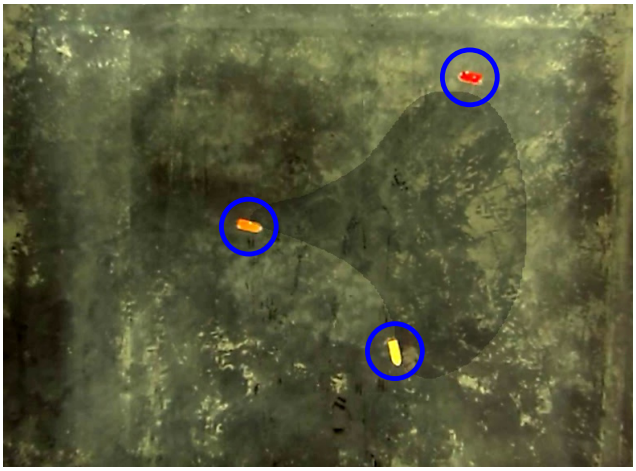
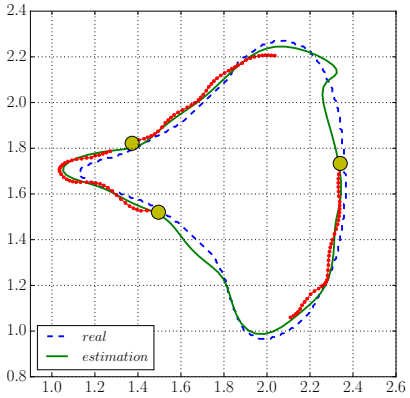
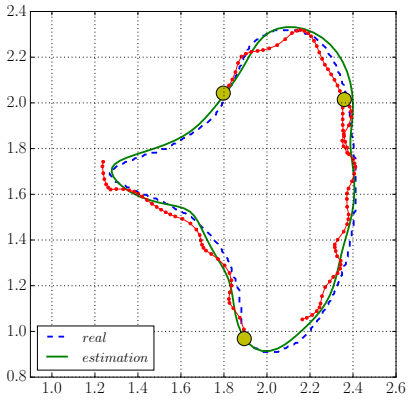


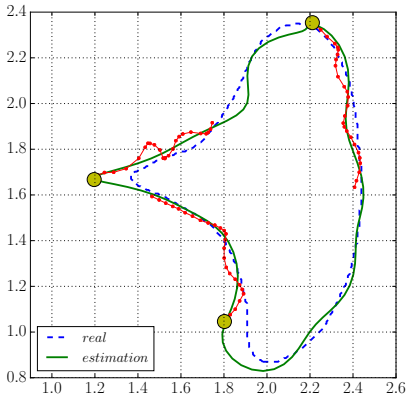
Fig. 9 Top view of the robotic boats in the tank. The robots are highlighted by the blue circles, and the tracked region is represented by the darker area.



(a) $t = 200$



(b) $t = 320$



(c) $t = 400$

Fig. 10 Snapshot of the experiment at two different time steps. The robot locations and their trajectories are represented by the yellow disks and the dotted line, respectively. The real boundary is represented by the dashed line and the estimation by the continuous green line.

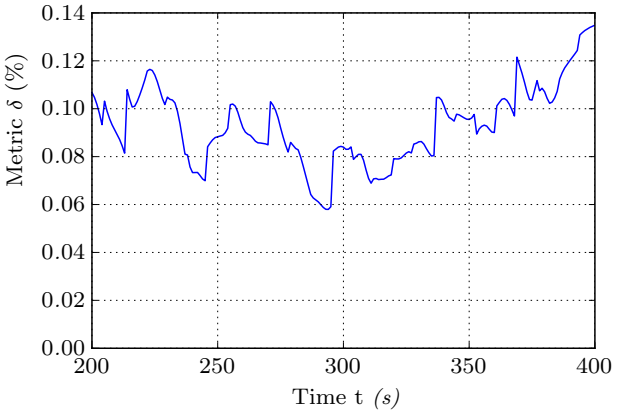


Fig. 11 Estimation error of the boundary for each time step using the last $k = 399$ samples (133 per robot).

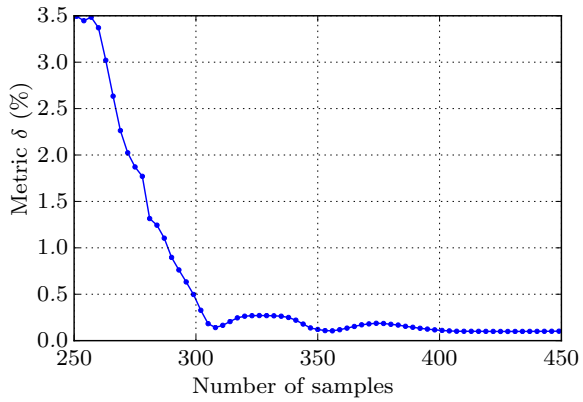


Fig. 12 Convergence of the estimator by increasing the number of samples k.

boundary by a polygon and apply the metric in (19) to compare it with the ground truth, we get a metric of 0.257, which is %222.84 larger than the metric for the estimation. The key to this method is combing old points and current points to project the location of the new points. We perform the projection by approximating the function in (11).

7 Conclusions and Future Work

In this paper, we presented a cooperative prediction method for dynamic boundaries using multiple robots. Our method maintains the robots uniformly distributed along the curve in order to track the boundary. We use point-wise measurements to fit a general boundary function, which is the combination of a polynomial and a truncated Fourier series. We validated our method through multiple simulations and experiments with actual robots.

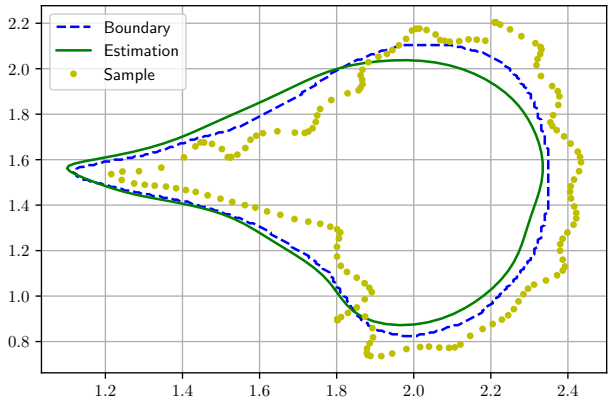


Fig. 13 Recent samples, estimation, and original boundary.

In a previous work (Saldaña et al, 2016), we showed that a single fast robot is able to predict the behavior of dynamic boundaries. In this paper, we study cooperative prediction using a team of n robots. We can say that given a phenomenon surrounded by a boundary with maximum variation ϵ_v, one or multiple robots can be used for the prediction, but there is a trade-off between the number robots and the feasible speed Υ. We can use a large number of slow-moving robots or a small number of high-speed robots.

As future work, we want to study how to predict the behavior of three-dimensional boundaries such as lava after a volcanic eruption, hurricanes, and whirlpools.

References

Barat C, Rendas MJ (2003) Benthic boundary tracking using a profiler sonar. In: Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, IEEE, vol 1, pp 830–835

Bertozzi AL, Kemp M, Marthaler D (2005) Determining Environmental Boundaries: Asynchronous Communication and Physical Scales, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 25–42. DOI 10.1007/978-3-540-31595-7˙2

Bruemmer DJ, Dudenhoeffer DD, McKay MD, Anderson MO (2002) A robotic swarm for spill finding and perimeter formation. Tech. rep., DTIC Document

Casbeer D, Beard R, McLain T, Li S, Mehra R (2005) Forest fire monitoring with multiple small uavs. In: American Control Conference, 2005. Proceedings of the 2005, Ieee, pp 3530–3535

Casbeer D, Kingston D, Beard R, McLain T (2006) Cooperative forest fire surveillance using a team of small unmanned air vehicles. International Journal of Systems Science 37(6):351–360

Clark J, Fierro R (2005) Cooperative hybrid control of robotic sensors for perimeter detection and tracking. Proceedings of the 2005, American Control Conference, 2005 pp 3500–3505

Cortés J (2012) Cooperative detection of areas of rapid change in spatial fields. Automatica 48(4):673–681

Cruz D, McClintock J, Perteet B, Orqueda O, Cao Y, Fierro R (2007) Decentralized cooperative control - a multivehicle platform for research in networked embedded systems. Control Systems, IEEE 27(3):58–78, DOI 10.1109/MCS.2007.365004

Duttagupta S, Ramamritham K, Kulkarni P (2011) Tracking dynamic boundaries using sensor network. Parallel and Distributed Systems, IEEE Transactions on 22(10):1766–1774

Fahad M, Saul N, Guo Y, Bingham B (2015) Robotic simulation of dynamic plume tracking by unmanned surface vessels. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp 2654–2659, DOI 10.1109/ICRA.2015.7139557

Franchi A, Stegagno P, Di Rocco M, Oriolo G (2010) Distributed target localization and encirclement with a multi-robot system. In: 7th IFAC Symposium on Intelligent Autonomous Vehicles, pp 151–156

Hsieh CH, Jin Z, Marthaler D, Nguyen BQ, Tung DJ, Bertozzi AL, Murray RM (2005) Experimental validation of an algorithm for cooperative boundary tracking. In: Proceedings of the 2005, American Control Conference, 2005., pp 1078–1083 vol. 2

Hsieh MA, Kumar V, Chaimowicz L (2008) Decentralized controllers for shape generation with robotic swarms. Robotica 26(05):691–701

Jin Z, Bertozzi AL (2007) Environmental boundary tracking and estimation using multiple autonomous vehicles. In: 2007 46th IEEE Conference on Decision and Control, pp 4918–4923, DOI 10.1109/CDC.2007.4434857

Joshi A, Ashley T, Huang YR, Bertozzi AL (2009) Experimental validation of cooperative environmental boundary tracking with on-board sensors. In: Proceedings of the 2009 Conference on American Control Conference, ACC'09, pp 2630–2635

Kemp M, Bertozzi AL, Marthaler D (2004) Multi-uuv perimeter surveillance. In: 2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No.04CH37578), pp 102–107

Li S, Guo Y, Bingham B (2014) Multi-robot cooperative control for monitoring and tracking dynamic plumes. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp 67–73

Marthaler D, Bertozzi A (2003a) Collective motion algorithms for determining environmental boundaries. In: SIAM Conference on Applications of Dynamical Systems

Marthaler D, Bertozzi A (2003b) Tracking environmental level sets with autonomous vehicles. Journal of the Electrochemical Society 129:2865

Matveev AS, Hoy MC, Ovchinnikov K, Anisimov A, Savkin AV (2015) Robot navigation for monitoring unsteady environmental boundaries without field gradient estimation. Automatica 62:227 – 235

Matveev AS, Semakova AA, Savkin AV (2017) Tight circumnavigation of multiple moving targets based on a new method of tracking environmental boundaries. Automatica 79:52 – 60

Mellucci C, Menon P, Edwards C, Challenor P (2017) Experimental validation of boundary tracking using the suboptimal sliding mode algorithm. In: 2017 American Control Conference, pp 4878–4883

Menon PP, Ghose D (2013) Boundary mapping of 3-dimensional regions. In: 2013 American Control Conference, pp 2984–2989, DOI 10.1109/ACC.2013.6580288

Menon PP, Edwards C, Shtessel YB, Ghose D, Haywood J (2014) Boundary tracking using a suboptimal sliding mode algorithm. In: 53rd IEEE Conference on Decision and Control, pp 5518–5523, DOI 10.1109/CDC.2014.7040252

Nagarathna, S V (2015) An adaptive refresh distributed model for estimation and efficient tracking of dynamic boundaries. In: 2015 Twenty First National Conference on Communications (NCC), pp 1–6

Nagarathna, Valli S, Manjunath D (2014) A spatio-temporal model for estimation and efficient tracking of dynamic boundaries. In: Communications (NCC), 2014 Twentieth National Conference on, pp 1–6, DOI 10.1109/NCC.2014.6811335

Pettersson LH, Pozdnyakov D (2012) Monitoring of harmful algal blooms. Springer Science & Business Media

Saldaña D, Assunção R, Campos M (2015) A distributed multi-robot approach for the detection and tracking of multiple dynamic anomalies. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp 1262–1267, DOI 10.1109/ICRA.2015.7139353

Saldaña D, Assunção R, Campos M (2016) Predicting environmental boundary behaviors with a mobile robot. IEEE Robotics and Automation Letters 1(2):1133–1139, DOI 10.1109/LRA.2016.2522500

Saldaña D, Assunção R, Hsieh MA, Campos MFM, Kumar V (2017) Cooperative prediction of time-varying boundaries with a team of robots. In: 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp 9–16, DOI https://doi.org/10.1109/MRS.2017.8250925

Sherman J, Morrison WJ (1950) Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics 21(1):124–127

Smith LC (1997) Satellite remote sensing of river inundation area, stage, and discharge: A review. Hydrological processes 11(10):1427–1439

Srinivasan S, Ramamritham K, Kulkarni P (2008) Ace in the hole: Adaptive contour estimation using collaborating mobile sensors. In: 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), pp 147–158, DOI 10.1109/IPSN.2008.38

Susca S, Bullo F, Martinez S (2008) Monitoring Environmental Boundaries With a Robotic Sensor Network. IEEE Transactions on Control Systems Technology 16(2):288–296

Susca S, Bullo F, Martinez S (2009) Gradient algorithms for polygonal approximation of convex contours. Automatica 45(2):510–516, DOI 10.1016/j.automatica.2008.08.020

Turgeman A, Werner H (2017) Mission control - combined solutions for source seeking and level curve tracking in a time-varying field. In: 2017 American Control Conference, pp 4268–4273

White B, Tsourdos A, Ashokaraj I, Subchan S, Zbikowski R (2007) Contaminant cloud boundary monitoring using uav sensor swarms. In: AIAA Guidance, Navigation and Control Conference and Exhibit, p 6761

Zhang F, Leonard NE (2005) Generating contour plots using multiple sensor platforms. In: IEEE Swarm Intelligence Symposium, pp 309–316

Zhang F, Leonard NE (2010) Cooperative filters and control for cooperative exploration. IEEE Transactions on Automatic Control 55(3):650–663, DOI 10.1109/TAC.2009.2039240

Zhang G, Fricke GK, Garg DP (2013) Spill detection and perimeter surveillance via distributed swarming agents. Mechatronics, IEEE/ASME Transactions on 18(1):121–129