Cooperative Prediction of Time-Varying Boundaries with a Team of Robots

David Saldaña[1], Renato Assunção[2], M. Ani Hsieh[1], Mario F. M. Campos[2], and Vijay Kumar[1], .

***Abstract*— Environmental boundaries, such as the borderline of a forest fire or an oil spill, pose a significant threat for living organisms. Anticipating the dynamics of these phenomena is a potentially life-saving indicator to support efficient and effective evacuations or to dispel the hazard. We propose a decentralized coordination method that allows multiple robots to efficiently sample and predict the behavior of environmental boundaries. Our method does not require a priori information about the boundary dynamics. We validate our proposal through experiments with actual robots. We demonstrate experimentally that our method can estimate and predict non-convex boundaries even with noisy measurements and inaccurate motion actuators.**

I. Introduction

Natural and urban environments with living organisms are susceptible to catastrophes due to external phenomena that generate hazardous contaminants. We can mention a couple of casualties such as oil spills [1], [2], forest fires [3], [4], harmful algae blooms [5], and radiation leaks [6]. These and other such elements possess a common characteristic, which is that the affected region in the environment is hemmed in by a perimeter. Anticipating the dynamics of these phenomena is a potentially life-saving indicator to support evacuations or to dispel the hazard.

When a team of robots robots starts exploring the environment looking for contaminated regions. These robots can execute different boundary searching algorithms such as random walk [6], spiral search [1], cooperative exploration [7], following an environmental function [8], or following a gradient [9]. A hybrid approach to search and track the boundary with multiple robots is presented in [1], [10]. A simplified hybrid hierarchical control technique is proposed in [11]. In a previous work [7], we proposed a coordination method to detect and to track multiple dynamic boundaries.

When the robots are already on the boundary, their main task is to accurately follow a static or time-varying boundary.

[1] D. Saldaña, M.A. Hsieh, and V. Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA: {dsaldana, m.hsieh, kumar}@seas.upenn.edu

[2] R. Assunção and M.F.M. Campos are with the VeRLab Laboratory, Universidade Federal de Minas Gerais, MG, Brazil: {assuncao, mario}@dcc.ufmg.br

*The authors appreciate the help and support of Dhanushka Kularatne at Drexel University during the experiments, and Professor Philip A. Yecko from The Cooper Union for providing the liquid dye experiment video.

*The authors gratefully acknowledge the support of the Colombian agency COLCIENCIAS, and the Brazilian agencies: CAPES, CNPq and FAPEMIG. We also acknowledge the support of DARPA grant HR00111520020, ONR grants N00014-15-1-2115 and N00014-14-1-0510, ARL grant W911NF-08-2-0004, NSF grant IIS-1426840, and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

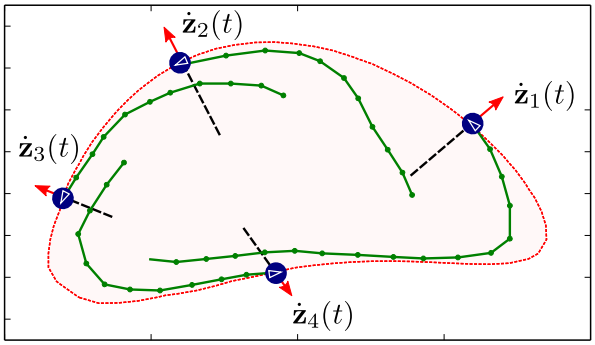


Fig. 1. Enclosing a boundary with four robots. The dashed red curve represents the recent shape of the boundary. The blue disks and their associated doted lines represent the robots on the curve and their paths respectively. The green points represent the point-wise samples. The arrows represent the velocity vector $\dot{\mathbf{z}}_i$.

In most of the tracking approaches, the robots are driven to circumnavigate the boundary in a counterclockwise manner [12], [13], [14]. We illustrate an environmental boundary tracked by four robots in Figure 1. Some works use gradient information to control the tracking action [12], [13], [15], 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 [16], [3], [17]. The most common and straightforward approach is implemented by the bang-bang algorithm [12], [18], [13], [17] in which a robot keeps persistently switching steering angles to change directions while circulating around the perimeter.

Once the robots are sampling the boundary using a tracking algorithm, an important task is to estimate the shape of the boundary by using the collected point-wise measurements. Usually, these point-wise measurements come from the location of the robot, for instance, using a GPS sensor. A preliminary work was presented by Kemp et al. [12]. They represent the boundary curve by 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 [19], [20], approximation theory of convex bodies is used to estimate slow-moving boundaries with a polygon with a fixed number of vertices. The work of [21] predicts the behavior of non-closed curves by using robots with range sensors. The method by Valli et al. [22], [23] takes into account the communication issues for reporting the boundary estimation as a sink. In a recent work [24], we proposed a method to predict the behavior of environmental boundaries without previous knowledge of the dynamics of the boundary

using a single robot. Extending this centralized approach to multiple robots brings challenging issues such as: design a coordination method to distribute the robots along the dynamic curve; integrate the sampled information from all robots without a central approach; parameterize the curve without a global view of the boundary and the samples. Figure 1 illustrates the sampled points during the robot motion, denoted by the green dotted line. We can see that each robot only has an updated point of the boundary (the point where it is located) the the rest of the sampled points are outdated and some of them unuseful to predict the boundary shape. In a centralized approach, all robots might send all their measurements to a central server. However, this would saturate the communication network and constrain the scalability of the multi-robot system.

Our main contribution is a method that allows multiple robots to estimate the dynamics of a environmental boundary using only point-wise measurements. We propose a mathematical framework to represent and estimate dynamic curves in the planar space. Our curve representation is based on the combination of polynomials and Fourier series and it naturally incorporates the variation in time. We highlight that our method does not require the dynamic model of the phenomenon.

II. 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. 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 all point $\boldsymbol{z} \in \partial\Omega_t$, and for any arbitrarily specified $\xi > 0$, the open disc centered at point $\boldsymbol{z}$ with radius ξ contains points of Ω_t and its complement $\Omega_t^{\complement}$.*

The boundary $\partial\Omega_t$ can be modeled by an unknown 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-intersection points.*

The unitarian tangent vector, at any point, is given by

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

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 $\boldsymbol{\gamma}$ 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 an ever increasing speed. Therefore, we assume the following.

Assumption 2 (Bounded motion). — *The magnitude of the velocity of any point $p \in \partial\Omega_t$ in the boundary is upper bounded by*

$$\left\|\frac{\partial\boldsymbol{\gamma}(t,s_0)}{\partial t}\right\| \leq \epsilon_v.$$

The robot team is composed by n robots that are initially distributed along the boundary. Their motion is constrained along the perimeter of the dynamic boundary $\partial\Omega_t$. Given a curve function $\boldsymbol{\gamma}(t,s)$, the location of robot i along the curve at time t can be described by an ever increasing curve parameter $s_i(t)$.

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 $i-1$) of the robot, 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 [12], [3], [17]. By using one of these tracking controllers, we can assume that the robots always move on the curve. In this way, the robot location in the Euclidean space $\mathbb{R}^2$ and its first order dynamic are given by

$$\begin{aligned} \mathbf{x}_i(t) &= \boldsymbol{\gamma}(t, s_i(t)) \\ \dot{\mathbf{x}}_i(t) &= \frac{d\boldsymbol{\gamma}(t, s_i(t))}{dt}. \end{aligned} \tag{2}$$

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

$$\|\dot{\mathbf{x}}_i(t)\| = u_i(t). \tag{3}$$

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

Assumption 3 (Relative 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

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

Each robot i is equipped with sensors to obtain local information in discrete time. At each sampling step t_k, $k \in \mathbb{N}$, the robot can observe a point in the curve as its own location, the velocity of a point in the boundary

$$\begin{aligned} \mathbf{z}_i(t_k) &= \mathbf{x}_i(t_k) \\ \dot{\mathbf{z}}_i(t_k) &= \frac{\partial\boldsymbol{\gamma}(t, s_i(t_k))}{\partial t}, \end{aligned}$$

and the unitarian tangent vector $\mathbf{T}_i := \mathbf{T}(t, s_i(t))$ of the curve at the robot location. We illustrate a scenario with four robots and their sampled 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 (Assumption 3), 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 [24], we presented this approximation with satisfactory results.

The robots use a ring communication topology to send and receive messages. 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 interchange messages with a successor $i+1$ and a predecessor $i-1$, where $n+1=1$.

Our objective is to use a team of robots to predict the behavior of a time-varying boundary. In Section III, we control the robot motion u_i to move along the boundary and to parametrize the unknown time-varying curve $\boldsymbol{\gamma}$(t,s) in a decentralized way. Using this method, we can associate a curve parameter s_j to each sample $(t_j, \dot{\mathbf{z}}_i(t_j))$. In this manner, each robot i has its own dataset $\mathcal{D}_i = \{(t_j, s_j, \mathbf{z}_i(t_j)) | j = 1, ..., k\}$. Then, the main problem of this paper can be stated as follows.

Problem. — *Given a team of n robots, where each robot i has its own dataset $\mathcal{D}_i$, estimate the function $\boldsymbol{\gamma}(t, s)$, which describes the dynamics of the environmental boundary.*

III. Cooperative Parametrization of the Boundary

In order to track the motion of every point on the closed-curve, we propose a method to parametrize the curve by using the measured information and local communication. In the initial stage, the robot with identification number equal to n is a temporary leader in charge of identifying if the boundary is enclosed, and creating the initial parametrization of the curve. After the initial parametrization, the robots continue working without a leader and estimating the boundary by only using local information and communication with their neighbors in the ring topology.

A. Enclosing the boundary

In the beginning, the robots start at different arbitrary locations on the curve. Since, the robots do not know their locations on the curve and they do not have enough information to estimate it, they move with constant speed until the boundary is cooperatively enclosed. The robots move in the counterclockwise direction along the curve by

$$u_i = \Upsilon,$$

where $\Upsilon > 0$ is a desired constant speed. During this stage, the local task of each robot i is to look for the initial point of the robot $i+1$ by projecting the vector $\mathbf{z}_i(t)$. The local task is completed if the projection of the vector $\mathbf{z}_i(t)$ crosses the path of the robot $i+1$. We illustrate a case where all robots complete their task in Figure 1. We can see four robots projecting their vector $\mathbf{z}_i(t)$ to enclose a boundary.

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. The robot $i-1$ is able to identify the trajectory of the robot i only using 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 interchanging messages. Each robot shares to its neighbors a boolean flag representing whether its local task was 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 resultant flag. After exchanges 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*.

B. Initial parametrization of the curve

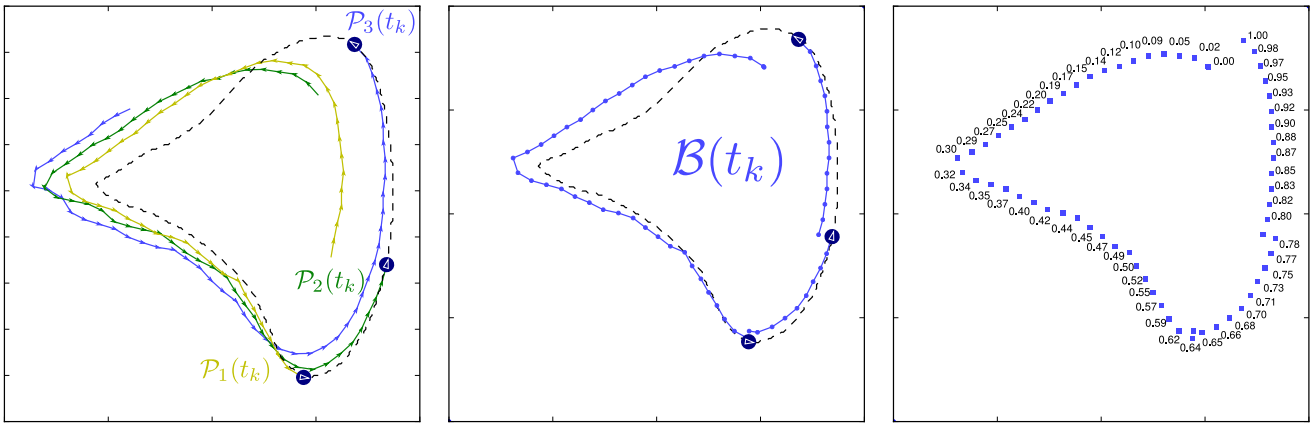
Once the boundary is enclosed, we proceed to assign a parameter value to each sampled point. If a previous estimation has not been computed yet, an initial way to enclose a boundary, is by using polylines [7]. A *polyline* defines a continuous line based on straight line segments, and it is composed 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.

In this task, robot n sends a set $\{\mathcal{P}_n(t_k)\}$, containing its own path, to the robot $n-1$. When robot $n-1$ receives this message, it computes the intersecting point $\mathbf{p}$ between the projection of its vector $\dot{\mathbf{z}}_{n-1}(t_k)$ and the polyline $\mathcal{P}_n(t_k)$. It removes the points after the intersecting point $\mathbf{p}$ to get a *shorter path* $\mathcal{P}'_n(t_k)$. Then, it sends the set $\{\mathcal{P}'_n(t_k), \mathcal{P}_{n-1}(t_k)\}$. The same process is repeated n times through the network until the temporal leader receives and computes the set $\{\mathcal{P}'_n(t_k), ..., \mathcal{P}'_1(t_k)\}$.

Definition 3. — *A* piecewise boundary $\mathcal{B}(t_k) := \{\mathcal{P}'_n(t_k), ..., \mathcal{P}'_1\}$ *is a set of n polylines that surrounds a boundary. Each of these polylines comes from a robot path, and they can be connected 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 that were presented in Figure 2(a). It is important to highlight that the piecewise boundary is the base of our method. In the next sections, we will describe our proposal to control the robots in order to improve its quality and increase the updating frequency of every point in the curve.

Now, we proceed to parametrize the piecewise boundary $\mathcal{B}(t_k)$. The length of the polyline, associated to robot i, is denoted by $\ell_i := \text{length}(\mathcal{P}'_i(t_k))$. The total length of the piecewise boundary is $\sum_{i=1}^{n} \ell_i$. The number of points in



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

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

(c) Parametrization of the sampled points (blue dots) by using the piecewise boundary.

Fig. 2. Distributed parametrization 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 paramatrization of the piecewise boundary based on the arc-lenght is illustrated in Panel (c).

a polyline is denoted by $|\mathcal{P}_i'(t_k))|$. In this way, the curve parameter for a given point $\mathbf{z}_i(t_j) \in \mathcal{P}_i'(t_k)$ is computed by

$$s_{ij} = \frac{1}{\sum_{i=1}^{n} \ell_i}\Big(\sum_{j=i+1}^{n} \ell_j + \sum_{k=1}^{j-1} \|\mathbf{z}_i(t_{k+1}) - \mathbf{z}_i(t_k)\| \Big).$$

Figure 2(c) exemplifies the parametrization of the piecewise boundary for the aforementioned example.

C. Online parametrization of the curve

Once the robots parametrize the curve for the first time, we map the new sampled points to the old ones in order to maintain an updated parametrization. At each time step t_k, robot i updates the piecewise boundary $\mathcal{B}(t_{k-1})$ using the samples $\mathbf{z}_i(t_k)$ and $\dot{\mathbf{z}}_i(t_k)$. It projects the vector $\dot{\mathbf{z}}_i(t_k)$ until it intersects the path of the robot $i+1$. Assume that the point of intersection is $\mathbf{z}_{i+1}(t_j)$. Since the new point $\mathbf{z}_i(t_k)$ is associated to the old parametrized point $\mathbf{z}_{i+1}(t_j)$, we reassign the parameter s_{ij} from the old point to the new point. In this way, we use the initial parametrization and try to track every point on the curve.

D. Distributing the robots along the curve

In this stage, 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 counterclockwise manner and we control the speed of the robots (from (3)). If the robots circulate with speed Υ around a relative slow-moving boundary, our problem becomes a distribution problem. It is possible to show that the best way to distribute the robots in the curve is to update every point as frequently as possible by distributing them equidistantly by arc-length. In this way, we minimize the update time of the least visited point and try to visit all the points with a constant frequency.

Franchi et al. [25] designed a control law to move and distribute a robot team around a circular shape. Sabattini et al. [26], [27] designed a distributed algorithm to make a team of robots circulate around arbitrary closed curves. However, they do not take into account the arc-length to distribute the robots along the curve. Then, our approach is modeled according to [25].

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

$$u_i = \Upsilon + K(\ell_{i+1} - \ell_i), \tag{4}$$

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 $\bar{\ell}_i = (\ell_{i+1} + \ell_{i-1})/2$. We highlight that using (4), 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 sampled 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 relative slow-changing boundaries (see Assumption 3).

IV. Cooperative Prediction

In this section, we extend the single-robot model [24] for distributed on-line prediction with multiple robots.

A. Modeling the curve function

Since we are tracking every single point in the curve, we start computing the trajectory of an arbitrary point. Assume that we want to approximate the trajectory of a single point with a parameter $s_0 \in [0, 1]$. Then, the trajectory of this point can be approximated by an n-degree polynomial,

$$\hat{\boldsymbol{\gamma}}(t, s_0) = \begin{bmatrix} \beta_{00} & \beta_{01} & \dots & \beta_{0n} \\ \beta_{10} & \beta_{11} & \dots & \beta_{1n} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ \vdots \\ t^n \end{bmatrix}. \tag{5}$$

Letting $\boldsymbol{\beta}(s_0)$ be the $2 \times (n+1)$ constant matrix for s_0, and $\mathbf{f}(t) = [t^0, t^1, ..., t^n]^T$ the exponents of the variable t, we

represent the trajectory of the boundary point as $\hat{\boldsymbol{\gamma}}(t, s_0) = \boldsymbol{\beta}(s_0)\,\mathbf{f}(t)$. Since the curvature changes smoothly, we can generalize this for every point in the curve $s \in [0, 1]$, by generalizing the matrix $\boldsymbol{\beta}$ as,

$$\hat{\boldsymbol{\gamma}}(t, s) = \boldsymbol{\beta}(s)\,\mathbf{f}(t). \tag{6}$$

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 $\mathbf{g}(s) = [1, \sin(2\pi s), \ldots, \sin(2\pi m s), \cos(2\pi s), \ldots, \cos(2\pi m s)]^T$. We join the Fourier vector $\mathbf{g}(s)$ and the polynomial vector $\mathbf{f}(t)$ by using the Kronecker product as

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

With some algebraic manipulation (details in [24]), we can separate the curve function (6) 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

$$\hat{\boldsymbol{\gamma}}(t, s) = \mathbf{C}^T\ \mathbf{h}(t, s). \tag{7}$$

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 $\mathbf{C}$.

B. Estimating the curve function

We use the time t and the curve parameter s as input variables, and the sampled locations $\mathbf{z}_i(t_j) = [x_j, y_j]^T$ 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 (7). We model the problem as a linear system with the form

$$\mathbf{X}\,\mathbf{C} = \mathbf{Y}, \tag{8}$$

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

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

Our objective consists of adjusting the parameters matrix $\mathbf{C}$, from (8), 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

$$\widetilde{\mathbf{C}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{9}$$

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

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

The matrix $\mathbf{X}$ can be estimated if a robot has the full data set $\mathcal{D}$ and requires computational power to compute $(\mathbf{X}^T\mathbf{X})^{-1}$. 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.

C. Online prediction

For each time step, the dataset $\mathcal{D}$ with k samples can be used to estimate the boundary behavior. However, when the robot takes a new sample $(t_{k+1}, x_{k+1}, y_{k+1}, s_{k+1})$, we must carry out the the expensive computation $(\mathbf{X}_{k+1}^T\mathbf{X}_{k+1})^{-1}$, which has a time complexity exponential on the number of points. In order to make it faster, we extend our method using *Recursive Least Squares* to obtain the updated prediction by taking advantage of the already computed $(\mathbf{X}_k^T\mathbf{X}_k)^{-1}$ and avoiding to invert the whole matrix in every time step.

In order to make (9) iterative, we rewrite it as the operation with the rows of $\widetilde{\mathbf{C}}$,

$$\widetilde{\mathbf{C}} = \Big(\sum_{i=1}^{k} \mathbf{h}_i\mathbf{h}_i^T\Big)^{-1} \sum_{i=1}^{k} \mathbf{h}_i\mathbf{y}_i^T, \tag{10}$$

where $\mathbf{h}_i = \mathbf{h}(t_i, s_i)$ and $\mathbf{y}_i = [x_i, y_i]^T$. The left sum in the right side of (10) defines the correlation matrix $\mathbf{P}$, which can be calculated in a recursive manner,

$$\mathbf{P}(i) = \mathbf{P}(i-1) + \mathbf{h}_i\,\mathbf{h}_i^T, \tag{11}$$

with $\mathbf{P}(1) = \mathbf{h}_1\,\mathbf{h}_1^T$, and the *cross-correlation vector* of right sum in (10),

$$\mathbf{q}(i) = \mathbf{q}(i-1) + \mathbf{h}_i\,\mathbf{y}_i^T, \tag{12}$$

with $\mathbf{q}(1) = \mathbf{h}_1\,\mathbf{y}_1$. Then, (9) is rewritten as the multiplication of two iterative terms,

$$\widetilde{\mathbf{C}} = \mathbf{P}(k)^{-1}\mathbf{q}(k). \tag{13}$$

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 $\mathbf{P} = \mathbf{P}^T$ and $(\mathbf{P}+\mathbf{h}\mathbf{h}^T)$ are invertible, we apply the Sherman-Morrison formula as,

$$\begin{aligned} \mathbf{P}(k)^{-1} &= (\mathbf{P}(k-1) + \mathbf{h}_k\mathbf{h}_k^T)^{-1} \\ &= \mathbf{P}(k-1)^{-1} - \frac{(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)^T}{1+\mathbf{h}_k^T\mathbf{P}(k-1)^{-1}\mathbf{h}_k} \end{aligned} \tag{14}$$

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

D. Using distributed information

We integrate distributed sampled point-wise measurements with an *sliding window* approach. Our online 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 (12) and (14) 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 get the curve function

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

which integrates the information that is distributed among the n robots. We want to 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.

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

V. Experiments

We want to validate our method through 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 error function

$$\delta(\Omega_t, \hat{\Omega}_t) = \frac{\mu(\Omega_t \setminus \hat{\Omega}_t) + \mu(\hat{\Omega}_t \setminus \Omega_t)}{\mu(\Omega_t)}, \qquad (15)$$

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 [19], [7], [24].

In a previous work [24], we show that our method can use a fast single robot to get accurate estimations for analytic boundaries, such as the one presented in [19]. 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.

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 4. 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

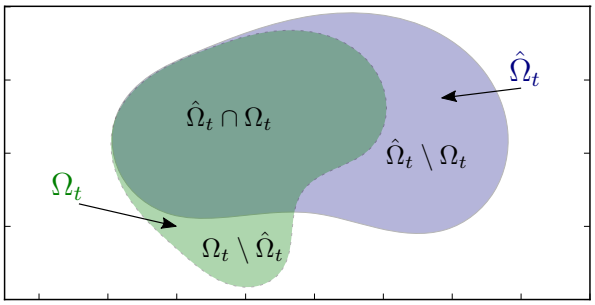


Fig. 3. Illustration of the metric $\delta(\Omega_t, \hat{\Omega}_t)$ for a bounded region Ω_t (green area) and its estimation $\hat{\Omega}_t$ (blue area). This metric quantifies the error in the estimated bounded region, by taking into account the non-estimated region $\Omega_t \setminus \hat{\Omega}_t$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

these differential robots which are able to track the boundary and to control their speed. We implemented the bang-bang algorithm [13] to keep the robots moving on boundary.

A small tank ($0.1 \times 0.1 \times 0.02m^3$) was used to generate and record the a boundary. The boundary was created by suspending two liquid dye droplets in a 1:1 glycerol-water mix. The submerged rotating disks is driven such that two adjacent 4×2 sets of disks are controlled by independent stepper motors and controllers. The experimental setting is shown in Figure 5. We track the boundary of the viscous liquid using an RGB camera and image processing. Then, we take the recorded boundary, remap it to the bigger tank and command the robots to track the recording. 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, and all at the same time. The environment is composed of a tank with a dimension of $3 \times 3 \times 0.5\, m^3$. The tank is filled with still water, and we use a team of three robotic boats as pictured in Figure 6. 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 boats show

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

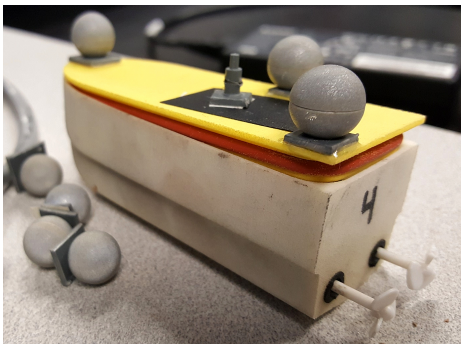


Fig. 4. A robotic boat driven by two propellers in the back. The reflective markers on the top are used for localization. A sealing system prevents water from entering the boat

difficulties to follow the boundary because the inertia makes the robots slide on the water making it difficult to compensate with its small actuators. Figure 7 depicts two snapshots of the experiment. We can see that the robots get outside the boundary and present some oscillations before returning to the proper tracking. Even though the boats were not able to move exactly along the boundary during the tracking process, we obtained estimations 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 7(b)). Other cases with less curvature did not present a high error in the estimation (see around the point $(2.0, 2.3)$ in Figure 7(a)). As a result, the estimation is also affected and it fits closer to the robot path than to the actual boundary. In contrast, we can also see that the estimation is approximately equal to the real boundary even after inaccurate samples. The error in the estimation (from (15)) during the experiment is presented in Figure 8. We can see that it presents some fluctuations due to the oscillatory movements, but in general the error is under 13%.

Our method offers better estimations as we increase the number of sampled points. In Figure 9, we illustrate how the metric δ is rapidly reduced as we increased 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, like curves that present oscillatory behaviors, using a large number of sampled points can be a problem if the degree of the polynomial is not increased.

VI. Conclusions and Future Work

In this paper, we presented a cooperative prediction method for dynamic boundaries using multiple robots. Our method is based on maintaining the robots equidistantly 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.

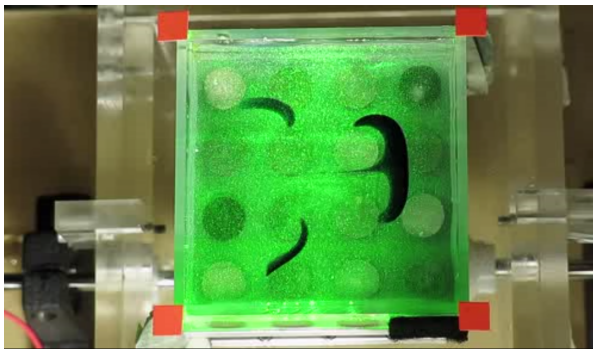


Fig. 5. Experimental setting to record the time-varying boundary. The tank contains two liquid dye droplets in a 1:1 glycerol-water mix. The nonlinear flow field is generated using a 4x4 lattice of submerged counter rotating disks. This image and a video of the time-varying boundary were courtesy of Prof. Peter Yecko.

In a previous work [24], we showed that a single fast robot is able to predict the behavior of dynamic boundaries. In this paper, we propose a method to coordinate a team of n robots to predict cooperatively. 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 desired, and also feasible, velocity Υ. We can use a large number of slow-moving robots or a single high-speed robot.

As a 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

[1] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," *Proceedings of the 2005, American Control Conference, 2005.*, pp. 3500–3505, 2005.

[2] M. Fahad, N. Saul, Y. Guo, and B. Bingham, "Robotic simulation of dynamic plume tracking by unmanned surface vessels," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 2654–2659.

[3] D. Casbeer, R. Beard, T. McLain, S. Li, and R. Mehra, "Forest fire monitoring with multiple small uavs," in *American Control Conference, 2005. Proceedings of the 2005*. Ieee, 2005, pp. 3530–3535.

[4] D. Casbeer, D. Kingston, R. Beard, and T. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[5] L. H. Pettersson and D. Pozdnyakov, *Monitoring of harmful algal blooms*. Springer Science & Business Media, 2012.

[6] D. J. Bruemmer, D. D. Dudenhoeffer, M. D. McKay, and M. O. Anderson, "A robotic swarm for spill finding and perimeter formation," DTIC Document, Tech. Rep., 2002.

[7] D. Saldaña, R. Assunção, and M. Campos, "A distributed multi-robot approach for the detection and tracking of multiple dynamic anomalies," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 1262–1267.

[8] D. Marthaler and A. Bertozzi, "Collective motion algorithms for determining environmental boundaries," in *SIAM Conference on Applications of Dynamical Systems*, 2003.

[9] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 05, pp. 691–701, 2008.

[10] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control - a multivehicle platform for research in networked embedded systems," *Control Systems, IEEE*, vol. 27, no. 3, pp. 58–78, June 2007.

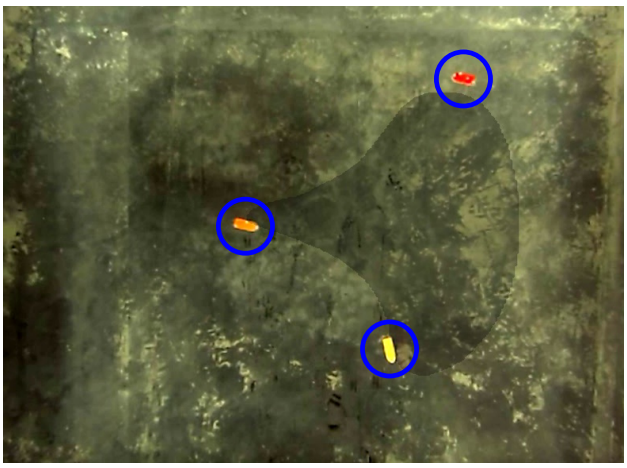
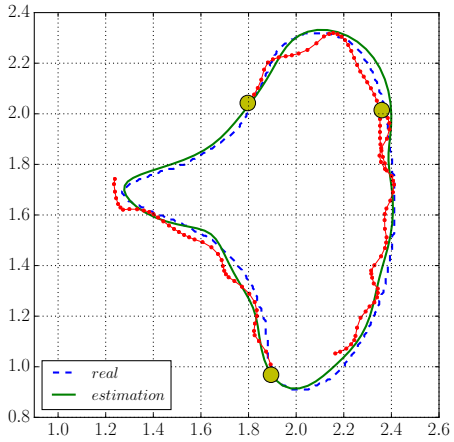
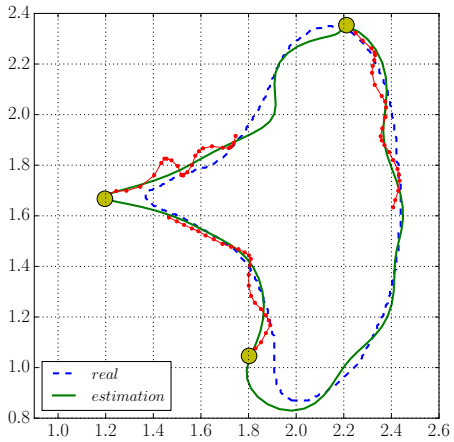


Fig. 6. Top view of the robotic boats in the tank. The robots are highlighted by the blue circles, and the tracked region is represented by the darker area.



(a) $t = 320$



(b) $t = 400$

Fig. 7. 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.

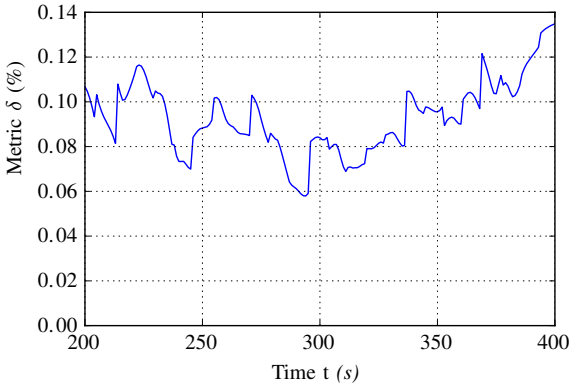


Fig. 8. Estimation error of the boundary for each time step using the last $k = 399$ samples (133 from each robot).

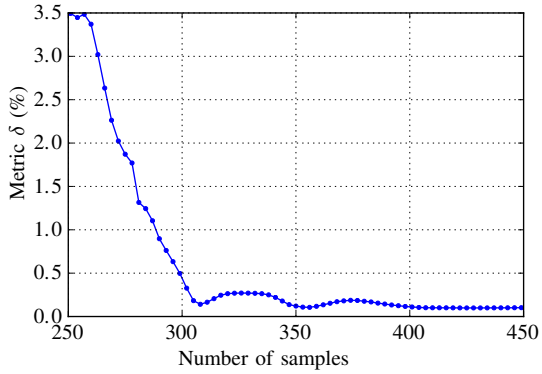


Fig. 9. Convergence of the estimator by increasing the number of samples k.

[11] G. Zhang, G. K. Fricke, and D. P. Garg, "Spill detection and perimeter surveillance via distributed swarming agents," *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 1, pp. 121–129, 2013.

[12] M. Kemp, A. L. Bertozzi, and D. Marthaler, "Multi-uuv perimeter surveillance," in *2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No.04CH37578)*, 2004, pp. 102–107.

[13] D. Marthaler and A. Bertozzi, "Tracking environmental level sets with autonomous vehicles," *Journal of the Electrochemical Society*, vol. 129, p. 2865, 2003.

[14] A. L. Bertozzi, M. Kemp, and D. Marthaler, *Determining Environmental Boundaries: Asynchronous Communication and Physical Scales*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 25–42.

[15] F. Zhang and N. E. Leonard, "Cooperative filters and control for cooperative exploration," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 650–663, March 2010.

[16] C. Barat and M.-J. Rendas, "Benthic boundary tracking using a profiler sonar," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2003, pp. 830–835.

[17] A. Joshi, T. Ashley, Y. R. Huang, and A. L. Bertozzi, "Experimental validation of cooperative environmental boundary tracking with onboard sensors," in *Proceedings of the 2009 Conference on American Control Conference*, ser. ACC'09, 2009, pp. 2630–2635.

[18] C. H. Hsieh, Z. Jin, D. Marthaler, B. Q. Nguyen, D. J. Tung, A. L. Bertozzi, and R. M. Murray, "Experimental validation of an algorithm for cooperative boundary tracking," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 1078–1083 vol. 2.

[19] S. Susca, F. Bullo, and S. Martinez, "Monitoring Environmental Boundaries With a Robotic Sensor Network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, Mar. 2008.

[20] ——, "Gradient algorithms for polygonal approximation of convex contours," *Automatica*, vol. 45, no. 2, pp. 510–516, Feb. 2009.

[21] S. Duttagupta, K. Ramamritham, and P. Kulkarni, "Tracking dynamic boundaries using sensor network," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 10, pp. 1766–1774, 2011.

[22] Nagarathna, S. Valli, and D. Manjunath, "A spatio-temporal model for estimation and efficient tracking of dynamic boundaries," in *Communications (NCC), 2014 Twentieth National Conference on*, Feb 2014, pp. 1–6.

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

[24] D. Saldaña, R. Assunção, and M. Campos, "Predicting environmental boundary behaviors with a mobile robot," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1133–1139, July 2016.

[25] A. Franchi, P. Stegagno, M. Di Rocco, and G. Oriolo, "Distributed target localization and encirclement with a multi-robot system," in *7th IFAC Symposium on Intelligent Autonomous Vehicles*, 2010, pp. 151–156.

[26] L. Sabattini, C. Secchi, C. Fantuzzi, and D. D. M. Possamai, "Tracking of Closed-Curve Trajectories for Multi-Robot Systems," pp. 6089–6094, 2010.

[27] L. Sabattini, C. Secchi, and C. Fantuzzi, "Closed-Curve Path Tracking for Decentralized Systems of Multiple Mobile Robots," *Journal of Intelligent & Robotic Systems*, pp. 109–123, 2012.