

**DETECÇÃO E PREDIÇÃO DE FRONTEIRAS
AMBIENTAIS USANDO UM TIME DE ROBÔS**

DAVID JULIAN SALDAÑA SANTACRUZ

**DETECÇÃO E PREDIÇÃO DE FRONTEIRAS
AMBIENTAIS USANDO UM TIME DE ROBÔS**

Tese apresentada ao Programa de Pós-Graduação em Ciencia da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciencia da Computação.

ORIENTADOR: MARIO FERNANDO MONTENEGRO CAMPOS
COORIENTADOR: RENATO MARTINS ASSUNÇÃO

Belo Horizonte

Julho de 2017

DAVID JULIAN SALDAÑA SANTACRUZ

**DETECTING AND PREDICTING
ENVIRONMENTAL BOUNDARIES WITH A
TEAM OF ROBOTS**

Thesis presented to the Graduate Program
in Ciencia da Computação of the Univer-
sidade Federal de Minas Gerais in partial
fulfillment of the requirements for the de-
gree of Doctor in Ciencia da Computação.

ADVISOR: MARIO FERNANDO MONTENEGRO CAMPOS
CO-ADVISOR: RENATO MARTINS ASSUNÇÃO

Belo Horizonte

July 2017

© 2017, David Julián Saldaña Santacruz.
Todos os direitos reservados

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Saldaña Santacruz, David Julián.

S162d Detecting and predicting environmental boundaries with
a team of robots. / David Julián Saldaña Santacruz. – Belo
Horizonte, 2017.
xvii, 86 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de
Minas Gerais – Departamento de Ciência da Computação.

Orientador: Mário Fernando Montenegro Campos
Coorientador: Renato Martins Assunção

1. Computação - Teses. 2. Robótica. 3. Monitoramento
ambiental. 4. Sistemas multi-robôs. I. Orientador
II. Coorientador. III. Título.

CDU 519.6*82.9(043)



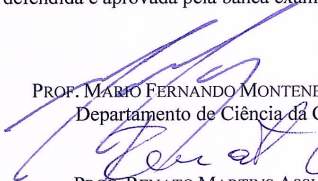
UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

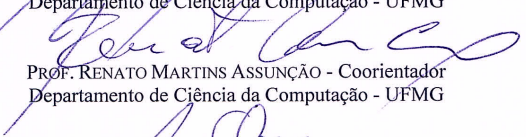
FOLHA DE APROVAÇÃO

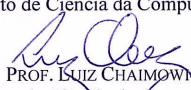
Detecting and predicting environmental boundaries with a team of robots

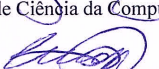
DAVID JULIÁN SALDAÑA SANTACRUZ

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. MARIO FERNANDO MONTENEGRO CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG


PROF. RENATO MARTINS ASSUNÇÃO - Coorientador
Departamento de Ciência da Computação - UFMG


PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG


PROF. LUCIANO CUNHA DE ARAÚJO PIMENTA
Departamento de Engenharia Eletrônica - UFMG


PROFA. M. ANI HSIEH
Departamento de Engenharia Mecânica - Drexel University


PROF. RAFAEL FIERRO
University of New Mexico

Belo Horizonte, 14 de julho de 2017

Abstract

Large area disasters are usually triggered by small-scale anomalies in small areas which could possibly be detected in their early stages. The past decade has seen effective proposals to approach this problem with the deployment of mobile sensors for monitoring disaster prone areas. In this work, we study three important tasks to achieve an autonomous system that can monitor the environment and prevent catastrophes. The first task concerns the disaster detection. The challenge is to coordinate multiple robots to explore the environment in order to find anomalies. The second task concerns the subsequent disaster tracking. Once the anomaly is detected, the robots must coordinate themselves to track the behavior of the environmental boundary. In the third task, the resulting tracking information from the second task is used to estimate the current and to predict its future shape. The combination of these three tasks integrates a monitoring system that can alert and mitigate the risk suffered by human and animal beings. In this dissertation, we present some contributions for each one of these tasks and for their integration. We validate our proposed methods by simulations and with actual robots. Our experiments showed good performance results.

List of Figures

1.1	Fire fronts in a forest with five monitoring robots	2
2.1	Two different manners of approaching a boundary.	9
2.2	<i>Hybrid wombling algorithm</i> [Cortés, 2012].	9
2.3	Spreading ten robots on two level curves [Cortés, 2012].	10
2.4	<i>Bang-bang</i> algorithm. Source [Kemp et al., 2004]	12
2.5	Estimation snapshots for a dynamic ellipse [Jin and Bertozzi, 2007a]. . . .	13
2.6	Multi-robot Plume Front Tracking Li et al. [2014].	14
2.7	Actual monitoring robotic boat in the British coasts. Source [Mellucci et al., 2017].	15
2.8	Hybrid controller for perimeter detection and tracking [Clark and Fierro, 2007].	16
2.9	State machine by Zhang et al. [2013].	17
2.10	Simulated results for splitting boundaries. Source [Zhang et al., 2013]. . . .	17
3.1	Processing robot path to identify a polygon.	26
3.2	Fusioning polylines	28
3.3	Simulation with six robots tracking an anomaly and two robots exploring the environment.	30
3.4	Allocation of robots to three static anomalies that appear at three different times.	31
3.5	Two dynamic anomalies expanding their area.	32
3.6	Two dynamic anomalies reducing their area.	32
3.7	Experimental testbed with three actual robots and two anomalies.	33
4.1	Dynamic boundary and the robot trajectory.	36
4.2	Robot trajectory in a short time interval.	38
4.3	Trajectory of a robot that is moving with constant velocity.	41
4.4	Trajectory of an arbitrary point in the boundary.	43

4.5	Dynamic boundary γ_1 , the robot trajectory and the resultant prediction. .	49
4.6	Dynamic boundary and the robot trajectory. The desired shape to be predicted is represented by the dashed line.	50
4.7	Estimation of the curve parameter s	50
4.8	Viscous substance in a liquid and the robot trajectory around it.	52
4.9	δ -error in time with different velocities.	53
4.10	Actual robot circulating around the dynamic boundary. The dots represent the sampled points along the robot trajectory.	53
4.11	Liquid estimation by an actual robot. Continuous lines represent the estimations and dashed lines are the real boundaries.	54
5.1	Enclosing the boundary with four robots.	58
5.2	Distributed parametrization of the closed-curve.	61
5.3	Consensus error for $n = 3$ and $n = 9$ number of robots.	70
5.4	Prediction error for different number of robots n	73
5.5	Prediction error for a fixed dataset.	73
5.6	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. .	74
5.7	A robotic boat. It is driven by two propellers in the back. The reflective markers on the top are used for localization.	74
5.8	Snapshots of experiment using three boats.	75
5.9	Prediction error using three robots.	76

List of Symbols

Notation	Description	Page List
$S(t)$	Trajectory of the robot along the boundary.	35–37
$\alpha(t, r)$	Parametrized curve by arc-length	60, 61
$\gamma(t, s)$	The function that describes the behavior of the boundary $\partial\Omega_t$	ix, 34–39, 41, 42, 45–47, 54, 55, 60–63, 69
δ	Metric based on the area of the symmetric difference.	x, 28, 46, 47, 49, 51, 68
$\dot{\mathbf{z}}$	Velocity of a point in the curve.	x, 55–59
ℓ	Path length	60
$\mathbf{T}$	Tangent vector.	54, 55, 57, 61
$\mathbf{x}$	Robot location.	55, 61
$\mathbf{z}$	Curve location.	55, 57–59, 64
$\mathcal{D}$	Set of observed data by robot moving along the boundary. Each entry contains the robot location; time and curvature parameter.	35, 38, 40, 44, 46, 64, 65, 67

Notation	Description	Page List
$\mathcal{S}$	Set of data sampled by robot moving along the boundary. It is defined by the location and time for each sample.	xv, xvi, 38–40
$\mathcal{B}$	Piecewise boundary.	58, 59
$\mathcal{O}$	Set of identified cycles in a sampled trajectory	40, 44,
$\mathcal{S}$		51
$\mathcal{O}$	The obstacle set	23
$\mathcal{P}$	Set of a team of robots	58
$\mathcal{R}$	Set of a team of robots.	19, 20
$\mathcal{W}$	Robot workspace.	19–21
μ	Lebesgue measure.	46, 47, 51, 68, 69
$\partial\Omega_t$	Dynamic boundary which changes its form along the time t	xv, 34, 35, 38–40, 54, 56
s	The curve parameter	x, xviii, 34–44, 46, 48, 51, 54, 63–65
θ	Robot orientation	20, 34
k	Boolean flag.	58, 59
r	Robot configuration.	20, 22, 24, 34, 35
v	Linear velocity of the robot.	35, 37

Contents

Abstract	ix
List of Figures	xi
List of Symbols	xiii
1 Introduction	1
1.1 Problem definition	2
1.2 Objectives	3
1.3 Contributions	3
1.4 Organization	6
2 State of the Art	7
2.1 Finding the boundary in the environment	7
2.2 Tracking static and dynamic boundaries	10
2.2.1 Hybrid approaches	14
2.3 Mapping the boundary shape	16
2.4 Predicting the boundary behavior	18
2.5 Discussion	18
3 Detecting and Tracking Multiple Dynamic Boundaries	21
3.1 Problem Statement	21
3.1.1 Available Information	22
3.1.2 Robot Model	22
3.2 Searching for anomalies	22
3.2.1 Motion model for particles	23
3.2.2 Exploration	25
3.2.3 Tracking an anomaly	26
3.2.4 Estimating the enclosing polygons	26

3.3	Multiple Anomaly Support	27
3.3.1	Polygon estimation	27
3.3.2	Multi-Robot coordination	29
3.4	Experiments and Results	29
3.4.1	Simulations	30
3.4.2	Experiments with real robots	32
3.5	Discussion	33
4	Predicting Boundary Behaviors with a Mobile Robot	35
4.1	Problem Statement	36
4.2	Robot trajectory around the dynamic boundary	37
4.3	Estimating the curve parameter	40
4.3.1	Identifying cycles	41
4.3.2	Estimating s	42
4.4	Predicting the boundary behavior	42
4.4.1	Estimating the trajectory of a boundary point	43
4.4.2	Estimating the boundary behavior	44
4.4.3	Modeling as a linear system	46
4.4.4	Estimating by least-squares	46
4.5	Convergence analyses	47
4.6	Experiments	47
4.6.1	Simulating a boundary function	48
4.6.2	Estimating a plume-front behavior	51
4.7	Discussion	54
5	Predicting Boundary Behaviors with Multiple Robots	55
5.1	Problem Statement	55
5.1.1	Dynamic boundary	55
5.1.2	Multi-Robot Model	56
5.1.3	Available Information	57
5.1.4	Objective	58
5.2	Cooperative Parametrization of the Boundary	58
5.2.1	Enclosing the boundary	59
5.2.2	Initial parametrization of the curve	60
5.2.3	Online parametrization of the curve	61
5.2.4	Distributing the robots along the curve	61
5.3	Cooperative prediction	65

5.3.1	Modeling the curve function	65
5.3.2	Estimating the curve function	66
5.3.3	Online prediction	67
5.3.4	Method 1: Forgetting old measurements	68
5.3.5	Method 2: Sliding window	69
5.4	Experiments	70
5.4.1	Simulations	71
5.4.2	Experiments with actual robots	72
5.5	Discussion	75
6	Conclusion Remarks	79
	Bibliography	81

Chapter 1

Introduction

Real-time monitoring is paramount in environments where disasters may occur at any moment and when human or animal beings are in danger. Disasters are usually initiated by anomalies or small phenomena that are not timely detected and possibly corrected or even reported. In most cases it would be highly desirable to not only detect, but also to identify and predict the behavior of the affected area, whose perimeter may change over time. A typical example is the monitoring of a forest, where the identification of the increase in temperature, possibly due to a fire, is of utmost importance. In this case, it is also highly desirable to be able to determine the affected area and predict its behavior in real time. This could be of great relevance to firefighters. Similarly, detecting and tracking anomalies are important tasks in several scenarios such as oil spills in water bodies, radiation and gas leaks from nuclear power plants, and algae bloom in lakes. Such environments demand continuous monitoring, since they involve large amount of sensor data to constantly measure the conditions of the region of interest.

Systems composed of cameras or multiple static sensors dispersed in the environment may be used for detection, but their capabilities are constrained when dynamic anomalies need to be tracked. Therefore, one feasible approach is to spread multiple mobile robots in the environment, which may be coordinated to navigate and to dynamically monitor changing anomalies in the environment. Complementary sensor fusion may be readily applied to estimate the sensed phenomenon based on the information acquired by the multiple robots, taking advantage of any additional information that may become available to provide a better partial observation. Such mobile robots, which are usually equipped with processors, wireless communication, and several sensors, constitute what has been known in the literature as Robotic Sensor Networks (RSN) [Sibley et al., 2002; Bruemmer et al., 2002]. Their locomotion system enables

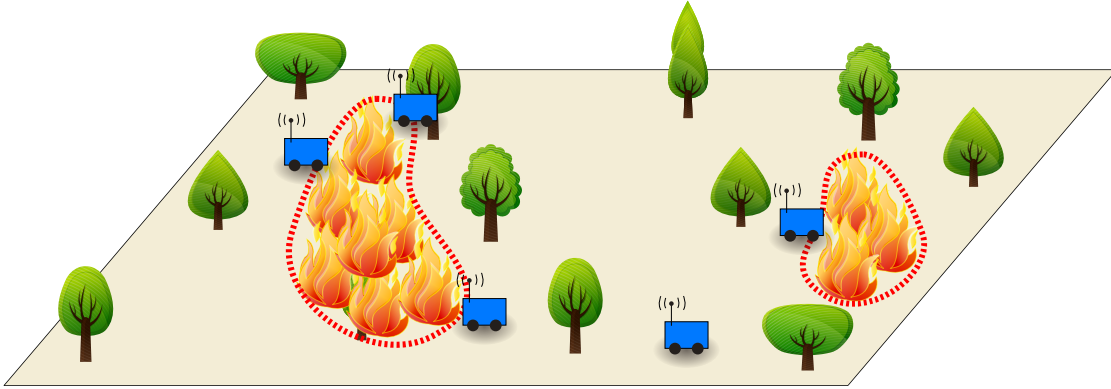


Figure 1.1: Fire fronts in a forest with five monitoring robots. The robots are represented by the blue squares and the fire front is represented by the red dotted line.

them to cover large areas and to adjust their location based on the environmental dynamics or other natural occurrences. In addition, RSNs have a great advantage over other monitoring techniques, such as wireless sensor networks or multiple static monitoring cameras, since they are able to dynamically modify their actions over time, which enables them to sense, detect, and also track anomalies in the monitored environment. Figure 1.1 illustrates five robots monitoring wildfire fronts in a forest. A single robot is allocated to explore the environment and the others are allocated to track two distinct fire anomalies at two different locations. In scenarios such as this, besides estimating the shape of the anomaly, it would be beneficial to predict how these areas would be affected in the near future. This prediction could guide pervasive actions to extinguish the problem.

In this dissertation, we use the term *anomaly* to designate an area in the environment where the value of a given physical variable is out of its typical range. This area is enclosed by a perimeter or boundary. An *anomaly* can be modeled as a *gradient surface* [Fiorelli and Leonard, 2007] or as a binary surface (*gradient-free surface*) [Kemp et al., 2004]. For instance, a gradient is generated by a physical phenomenon like temperature or light, which typically decays with distance. In this case, a robot would follow an iso-temperature or iso-illumination level-curve. In the gradient-free case, a robot would just identify whether it is within the affected area and then proceed to identify the boundary of the sensed anomaly.

1.1 Problem definition

The problem that we approach in this dissertation is how to detect and to predict the behavior of dynamic environmental anomalies delimited by a perimeter. Specifically,

given a set of robots in a finite planar environment, we want to detect and to track dynamic compact sets enclosed by boundaries. Next, we want to use the tracking measurements to estimate the current anomaly shape as well as to predict its future behavior.

1.2 Objectives

Our main objective in this dissertation is to propose a method to detect dynamic environmental boundaries and to predict their behavior. In order to achieve this objective, we need to accomplish the following specific objectives for a team of robots:

1. Explore the environment in order to detect anomalies.
2. Track the anomaly boundary to get point measurements of its perimeter.
3. Integrate the exploration and tracking processes in order to assign robots either to track current anomalies or to look for new anomalies.
4. Estimate the boundary shape and to predict its future behavior with a single mobile robot.
5. Generalize the previous objective by estimating and predicting boundary behaviors with a team of robots.

1.3 Contributions

The main contributions of this dissertation have been submitted and accepted for publications as follows:

1. In Chapter 3, we propose a probabilistic method to detect and to track environmental boundaries. This contribution is focused on Objectives 1 and 2. We propose an algorithm to coordinate a team of robots that are tasked to find and track multiple anomalies. The exploration under uncertainty maintains the history of the sensed areas to decrease redundancy and to allow for full-map coverage. This work was selected as one of the twelve best papers of the conference:

D. Saldaña, L. Chaimowicz, and M.F.M. Campos

“Searching and tracking anomalies with multiple robots: A probabilistic approach”

LARS'2014 - 11th Latin American Robotics Symposium.

An extended version, with additional experiments was published as a book chapter:

D. Saldaña, L. Chaimowicz, and M.F.M. Campos

“Searching for Regions Out of Normal Conditions Using a Team of Robots”

Communications in Computer and Information Science. Springer Berlin Heidelberg, 2015. p1-15.

2. Additionally in Chapter 3, our approach allows for autonomous on-line allocation and deallocation of robots during the tracking process, depending on the size of the anomalies, which we assume, may dynamically vary. In our approach, a growing anomaly demands closer attention and more robots to be allocated throughout mission execution. Meanwhile, if an anomaly is shrinking, robots formerly allocated to that anomaly may be released to explore other areas with higher priority. Therefore, we approach Objective 3 by developing an integration of the particle filter technique with polygonal approximation. This work was published in the conference:

D. Saldaña, R. Assunção, and M.F.M. Campos

“A distributed multi-robot approach for the detection and tracking of multiple dynamic anomalies”

ICRA 2015 - IEEE International Conference on Robotics and Automation. IEEE, 2015, pp. 1262–1267.

3. We propose a novel method to predict environmental boundary behaviors in Chapter 4. The innovative contribution is that we do not use previous knowledge evolution or dynamics of the phenomenon. We approach the problem in a different way, by modeling the evolving boundary with a position-specific parametric analytical continuous function, where the parameters can be estimated using the sampled points by the robot moving along the perimeter. Our proposal not

only estimates the current closed-curve state, but it can also predict the future boundary behavior without previous knowledge of the phenomenon. This contributions approaches the Objective 4. It is published in the the following journal:

D. Saldaña, R. Assunção, and M. F. M. Campos
 “Predicting Environmental Boundary Behaviors with a Mobile Robot”
IEEE Robotics and Automation Letters (Volume: 1, Issue: 2, July 2016).

4. In Chapter 4, we also propose a method to circulate around dynamic boundaries with a mobile robot. It is published in a conference proceedings:

D. Saldaña, R. Javanmard, L.C.A. Pimenta, R. Assunção, and M.F.M. Campos
 “Dynamic Perimeter Surveillance by a Team of Robots”
ICRA 2016 - IEEE International Conference on Robotics and Automation.

and an extended version:

A. Jahn, R. Javanmard Alitappeh, D. Saldaña, A.G. Santos, L. Pimenta, and M.F.M. Campos
 “Distributed Multi-Robot Coordination for Dynamic Perimeter Surveillance in Uncertain Environments”
ICRA’2017 - IEEE International Conference on Robotics and Automation.

5. In Chapter 5, we extend our prediction method for multiple robots:

D. Saldaña, R. Assunção, M.A. Hsieh, M. F. M. Campos, and V. Kumar
 “Cooperative Prediction of Time-Varying Boundaries with a Team of Robots”
MRS’2017 - International Symposium on Multi-Robot and Multi-Agent Systems.
 (Submitted).

There are additional contributions that were developed during the Doctorate-Interchange program in the GRASP Laboratory, University of Pennsylvania, under the supervision of Prof. Vijay Kumar. These contributions on multi-robot systems and swarm robotics were published in the following conferences:

1. D. Saldaña, B. Gabrich, M. Whitzer, A. Prorok, M.F.M. Campos, M. Yim, and V. Kumar

“A Decentralized Algorithm for Assembling Structures with Modular Robots”
IROS’2017 - IEEE/RSJ International Conference on Intelligent Robots and Systems. (To be published).

2. D. Saldaña, A. Prorok, M.F.M. Campos, and V. Kumar
 “Triangular Networks for Resilient Formations”
DARS’2016 - 13th International Symposium on Distributed Autonomous Robotic Systems.
3. D. Saldaña, A. Prorok, M.F.M. Campos, and V. Kumar
 “Resilient Consensus for Time-Varying Networks of Dynamic Agents”
ACC’2017 - IEEE American Control Conference.
4. K. Saulnier, D. Saldaña, A. Prorok, G. Pappas, and V. Kumar
 “Resilient Flocking for Mobile Robot Teams”
RAL’2017 - IEEE Robotics and Automation Letters (July 2017).

1.4 Organization

The following chapters of this dissertation are organized as follows. In Chapter 2, we characterize the related works in detection, tracking, and prediction for environmental boundaries. In order to obtain the data for prediction, we need to detect and track the boundaries in the environment. We present our multi-robot approach for allocating robots to the searching and tracking tasks in Chapter 3. We describe our method to predict boundary behaviors using a single robot in Chapter 4. An extension of this method for multiple robots is presented in Chapter 5. Finally, we present conclusions and the future directions.

Chapter 2

State of the Art

In the robotics literature, related works about monitoring environmental boundaries can be categorized in four main research areas:

1. Finding the boundary in the environment.
2. Tracking static and dynamic boundaries.
3. Mapping the boundary shape.
4. Predicting the boundary behavior.

We expand the description and study the main works in each category along the following sections in this chapter. The concept of estimating and tracking boundaries is directly related to environmental sets, level curves, and perimeters. The boundaries that we study can be either dynamic or static, and the environment may offer gradient information as an scalar field or just a binary response indicating if the sensor is inside or outside the anomalous area.

2.1 Finding the boundary in the environment

In this section, we describe different methods to find the boundaries, where mobile sensors are scattered in the environment and are tasked to search for anomalous regions.

The work of Bruemmer et al. [2002] is one of the pioneers in this area. They propose a bio-inspired algorithm using a *robot-swarm* to detect gradient-free chemical contamination area. It uses social potential fields to search for static anomalies in a finite area. Their objectives are focused on leading the robots: to minimize the redundancy; to increase the social interaction; and to maximize the covered areas. By

on-line adaptation, the swarm uses feedback to adjust the sensibility of the sensors, after which, it can be adapted to new environments. Initially the robots are deployed into the environment in a leader following mode. Then, the exploration mode is based on attractive and repulsive forces based on sounds. The attractive force maintains the group together and connected, and the repulsive force avoid collisions among the robots. The robots navigate in the environment and avoid the fixed anomalies. After moving around every possible spot, the remaining area corresponds to the desired regions of interest. Although this method offers a full map coverage, the boundaries must have fixed size and the exploration is uncoordinated. Only after exploring the entire map, it is possible to identify the desired regions.

Srinivasan et al. [2008] propose an algorithm to move the robots towards a contour or a level curve in a scalar field. The mobile sensors use the gradient information in the field to move towards the contour. Simultaneously, the robots spread apart while approaching the contour. There is a tread-off between these two actions. Spreading too early or too much increases the time to trace the boundary. Hence, an on-line decision making mechanism was developed. The authors propose three approaches to surround contours. The first one is *Direct Descent* (DD), a greedy heuristic that uses the gradient to move the sensors directly to the contour value. When the robots converge to the contour, DD uses a wall following approach to trace the contour until it connects the robots in a ring topology. Then, the robots move in a clockwise direction just to trace the uncovered curve segment. This approach has a high redundancy in the wall following stage when the robots start together. In Figure 2.1a it can be seen how the robots start together, move together and arrive in the same way to the boundary. The tracing stage takes a long time because only the robots in the extremes will be working efficiently.

An improved approach is called *Always Spreading* (AS), which uses the centroid of the boundary to spread the robots along the contour. An example of this improved version is depicted in Figure 2.1b. However, the robots do not converge to the contour for all cases. Therefore, an integration of the two first approaches defines a third method, named as *Adaptive Contour Estimation* (ACE). It dynamically makes decisions on how much to spread the robots, and when to approach the boundary. These decisions are based on the on-line estimation of two variables: the distance between the robots and the level; and the arc-length of the boundary. Therefore, the authors present an efficient method to move a set of robots towards a static level curve based on measuring the gradient locally. However, this method does not work with phenomena that can only be sensed when the robot is close or inside the anomalous area.

Cortés [2012] proposes a cooperative algorithm to detect boundaries that sur-

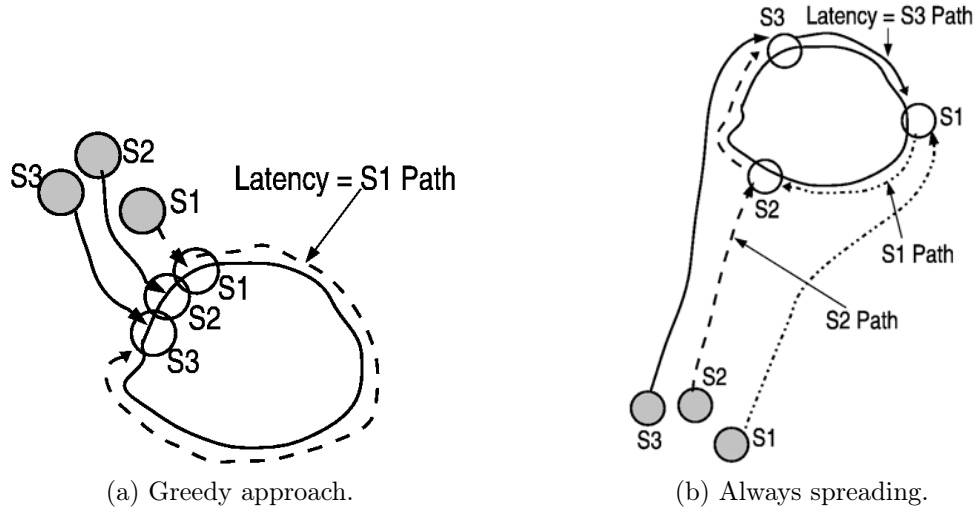


Figure 2.1: Two different manners of approaching a boundary. Panel (a) shows a greedy approach, where the robots are always following the gradient directly (Direct Descent). Panel (b) shows the opposite case, where the robots spread around the centroid of the contour (Always Spreading). Source [Srinivasan et al., 2008].

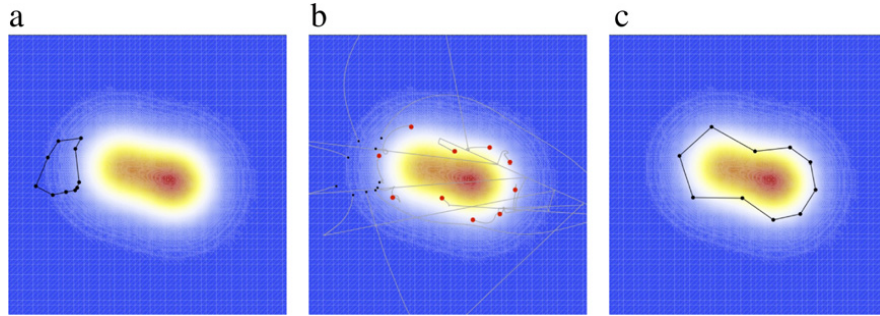


Figure 2.2: *Hybrid wombling algorithm*: Spreading ten robots on a level curve. Their initial configuration is shown in Panel (a), robot trajectories in Panel (b), and their final configuration is shown in Panel (c). Source [Cortés, 2012].

round areas with rapid changes based on statistical estimation. Figure 2.2 presents the execution of his *hybrid wombling algorithm*. This method also supports splitting and merging curves, see Figure 2.3. As the ACE method, these techniques are based on gradient information to follow the direction where the desired level curves are. This constraint limits the applications in gradient-free environments.

Searching for boundaries at the first stage, when the robots coordinate the way to explore the environment, can also be seen as a target searching problem [Bertuccelli and How, 2006; Mottaghi and Vaughan, 2006; Chung and Furukawa, 2009; Grocholsky et al., 2006]. However, their work do not make it clear how to extend a point target to a closed curve target.

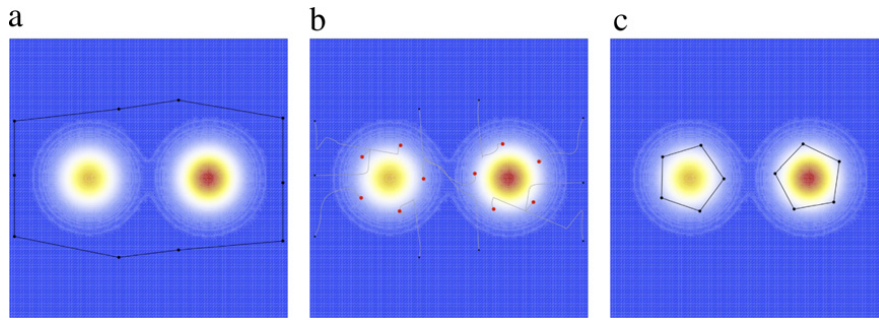


Figure 2.3: Spreading ten robots on two disjoint level curves. Their initial configuration is shown in Panel (a), robot trajectories in Panel (b), and their final configuration is shown in Panel (c). Source [Cortés, 2012].

2.2 Tracking static and dynamic boundaries

After detecting the existence of the anomaly in the environment and all the robots are already on the boundary, their main task is accurately following a static or time-varying boundary. In most of the tracking approaches, the robots are driven to circumnavigate around the boundary in 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]. Other works use gradient-free approaches where each robot can only sense if it is inside or outside the contaminant [Barat and Rendas, 2003; Casbeer et al., 2005; Joshi et al., 2009]. Additionally, to keep the robots circulating around the boundary, a desired objective in the tracking task is keeping the robots evenly distributed along 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.

Marthaler and Bertozzi [2003a] propose a method to cooperatively track a boundary using a robotic swarm. Their algorithm is inspired in a well-known technique in image processing called *snake* [Kass et al., 1988]. In this process, an elastic closed curve is simulated to dynamically adapt its shape to the contour of the object by iterative, stepwise deformations. In their initial work, they show that the adaptation can be accomplished by a set of mobile agents. Different behaviors are employed depending on the initial deployment. If the agents are initially deployed as a large closed curve that encompasses the desired region, then the shrinking behavior reduces the closed curve progressively until all robots are located along the boundary. When the group of robots start inside the anomaly, they move according to the inflating behavior. The robots mutually repel each other to progressively cover a bigger region until the boundary

is reached. An improved version was presented in [Marthaler and Bertozzi, 2003b], which includes robustness to noisy sensors. The method works for static boundaries and requires a large number of robots and there is not a straightforward way to account for obstacles in the environment.

Barat and Rendas [2003] presented an experimental work to make an Autonomous Underwater Vehicle (AUV) to track multiple benthic zones autonomously. It is a robust technique that works with a noisy sonar sensor under the sea. It is based on two main modules: (i) *Acoustic data processing module*, which filter the noisy data from the profile sonar; (ii) *Acoustic controller*, which uses the processed information to send the control signals to guide the robot along the boundary. Although, benthic zones are assumed to be static as well as their boundaries, the challenge is how to make the robot work in such difficult environment.

Zhang and Leonard [2005] proposed a cooperative method to track level curves by means of the gradient information 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. They prove that this method can achieve asymptotic convergence to: (i) maintain the formation center moving along a level curve with constant velocity, and (ii) maintain stable robot formation during the tracking process.

Specifically for underwater applications, Fiorelli and Leonard [2007] develop a control model for cooperative gliders with adaptive sampling strategies. It allows a team of four robots to navigate along gradients with different formations. As most of the related approaches that work on vector fields, they are based on strong assumptions that the gradient can be measured in the whole environment and that there are no singular points (where the gradient is vanished). As these works are focused on the accurate tracking, the estimation procedure is very poor as it is an straightforward interpolation of the sampled points.

One of the most widely used algorithms to track level curves or boundaries is proposed by Kemp et al. [2004]. The classic algorithm is called *bang-bang* type steering controller. The idea is based on the an unicycle robot model, $u = (v, \omega)$, where the linear speed v and angular velocity ω are the control inputs. The robot is located at the point $x \in \mathbb{R}^2$, and it is able to measure if it is inside the boundary $z(x) > 0$, on the boundary $z(x) = 0$, or outside $z(x) < 0$. Then, the robot moves with constant linear

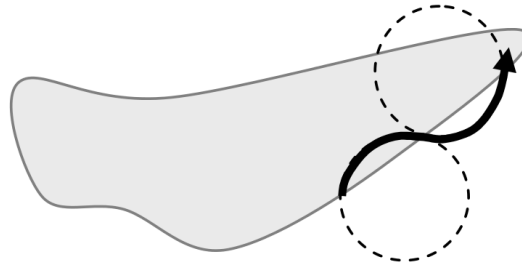


Figure 2.4: *Bang-bang* algorithm. Source [Kemp et al., 2004]

velocity $v = v_0$ and its angular velocity is given by:

$$\omega = \begin{cases} +w, & \text{if } z(x) > 0, \\ 0, & \text{if } z(x) = 0, \\ -w, & \text{otherwise.} \end{cases}$$

where $w > 0$ is a constant angular velocity of the robot. Figure 2.4 illustrates a robot trajectory applying the *bang-bang* algorithm. This straight-forward algorithm has been widely used because of its simplicity and because it can also be used to track dynamic boundaries. Experimental validations are performed by the *Caltech Multi-Vehicle Wireless Testbed* [Hsieh et al., 2005; Nguyen et al., 2005]. However, this technique presents an oscillatory behavior around the boundary line that besides being desirable, is not efficient. For this reason, Jin and Bertozzi [2007a] propose an improved version that uses an accumulative sum filter (CUSUM) [Srivastava and Wu, 1993] to reduce the error in the tracking process.

Jin and Bertozzi [2007a] present a centralized method to estimate dynamic boundaries. It assumes that there is a global observer which can perceive all robot locations at the same time. The prediction requires a model of the dynamics of the phenomenon. A Hidden Markov Model (HMM) is used to infer the real boundary state based on the robots measurements and a dynamics model. Figure 2.5 shows some snapshots to compare boundary estimation with the actual one. An experimental validation of the tracking algorithm is performed in [Joshi et al., 2009]. This work experimentally compares latency and precision to track static boundaries.

Menon et al. [2014] present 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] propose a multi-target tracking approach for a single robot model where a conglomerated set of targets is modeled as a dynamic environmental boundary.

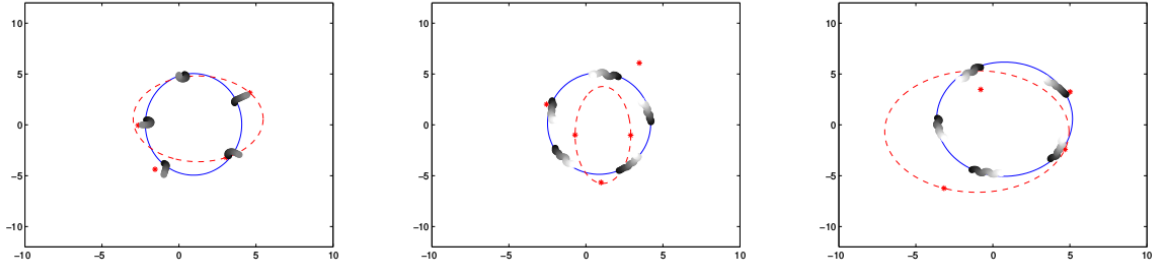


Figure 2.5: Estimation snapshots for a dynamic ellipse in [Jin and Bertozzi, 2007a]. The dashed line represents the estimation and the continuous line is the actual boundary.

Specifically in the context of forest fire surveillance, Casbeer et al. [2005, 2006] deal with the problem of monitoring and tracking the propagation of large forest fires with multiple *Unmanned Aerial Vehicles*, in which onboard cameras detect the fire and image based techniques are used to detect the perimeter of the affected zone. Instead of sampling point-wise data, this work uses the information from the 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. In [Euler et al., 2012], a control model is proposed to track multiple gradients with UAVs, where robots starts following the gradient and then track level curves.

The work of Duttagupta et al. [2011] introduces the concept of prediction for non-closed curves by using robots with range sensors. However, this kind of method cannot be directly applied to any of the aforementioned applications because it assumes that a single robot can track multiple points at the same time.

A 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. The robot controller that they apply is analytically constructed with proved convergence for chemicals or liquid substances poured in a marine environment. Their method is specifically tailored for boundaries with dynamics satisfying the advection-diffusion behavior. A similar approach to track plumes with multiple quadcopters is presented in [Shen et al., 2017]. There is a recent interest to deploy monitoring robots on real environments [Mellucci et al., 2017]. Figure 2.7 shows an actual monitoring robotic boat in the British coasts.

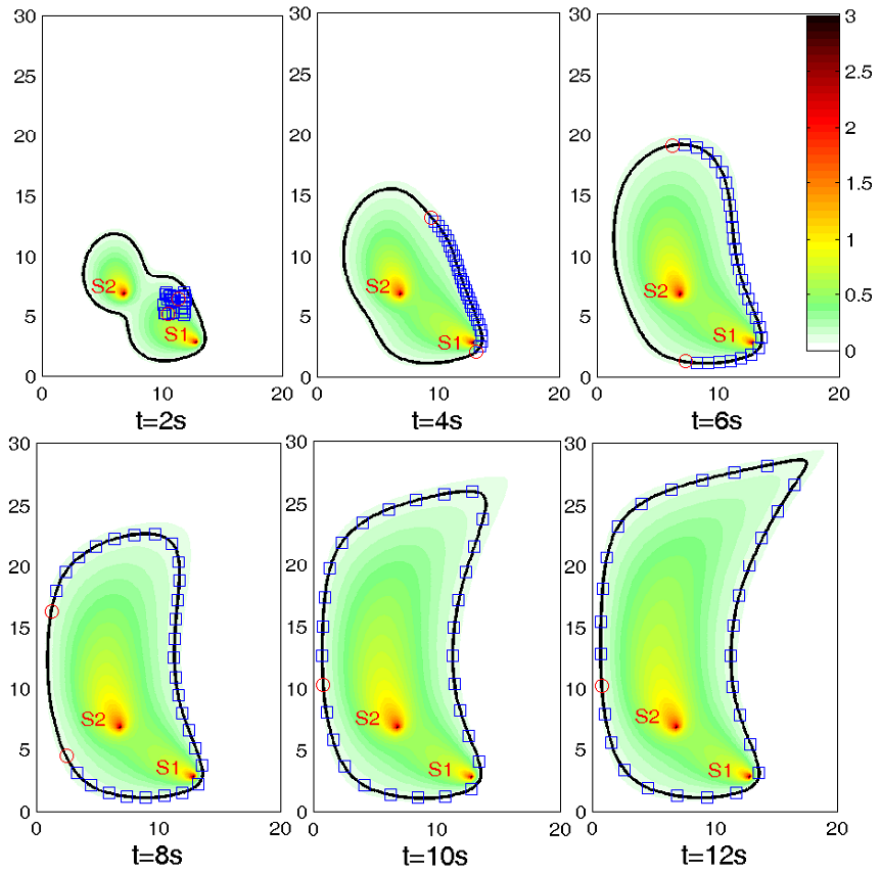


Figure 2.6: Multi-robot Plume Front Tracking. The snapshots show the evolution of the plume and the robots around it. The red dots, S_1 and S_2 , represent the leak sources. Source [Li et al., 2014].

2.2.1 Hybrid approaches

In this sub-category there are methods which integrate both searching and tracking tasks. The objective is to develop systems that improve the synergy between the two aforementioned tasks. One such a work is proposed by Clark and Fierro [2005, 2007], where the authors describe a hierarchical controller, modeled as a finite state machine that integrates different independent controllers such as: spiral exploration, obstacle avoidance, potential attraction, and tracking. Figure 2.8 shows the proposed state machine and the links that trigger the interaction among the states. A detailed description of the implementation in actual robots is extended in [Cruz et al., 2007].

In [Marthaler and Bertozzi, 2003b], the authors extend the robot formation for boundary detection based on the *snake* algorithm [Marthaler and Bertozzi, 2003a] to also track an anomaly.

Another hybrid hierarchical control technique is proposed by Zhang et al. [2013]. Each robotic agent has three possible states: searching, pursuing, and tracking, as



Figure 2.7: Actual monitoring robotic boat in the British coasts. Source [Mellucci et al., 2017].

presented in Figure 2.9. In contrast to [Clark and Fierro, 2007], this method support perimeters that can split in two or multiple parts. Figure 2.10 shows the experimental results for the special case when the boundary splits. Additionally, this Zhang’s controller does not require submodes for obstacle or vehicle avoidance. The potential functions are embedded in the main control states to support collision avoidance. The works of [Clark and Fierro, 2007] and [Zhang et al., 2013] focus on the interaction among the control states, but the states are very simple. For example, in the exploration process, Clark and Fierro [2007] apply spiral exploration and Zhang et al. [2013] apply random walk. In a recent work, we presented an algorithm that offers synergy with the tracking process [Saldaña et al., 2015].

The method by [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 sample data to the sink is predicted. The contour is modeled as a non-closed curve with drifted Brownian motion. They consider error in the measurements of the boundary, and the curve estimation is made by interpolating the filtered sampled points. In a recent work Matveev et al. [2015], a single non-holonomic Dubins-car like robot is used to detect and track a boundary.

Even though recent works such as [Clark and Fierro, 2007; Zhang et al., 2013; Nagarathna and S., 2015; Saldaña et al., 2015] have the advantage of integrating the detection, tracking and estimation processes, the boundary is projected by interpolating

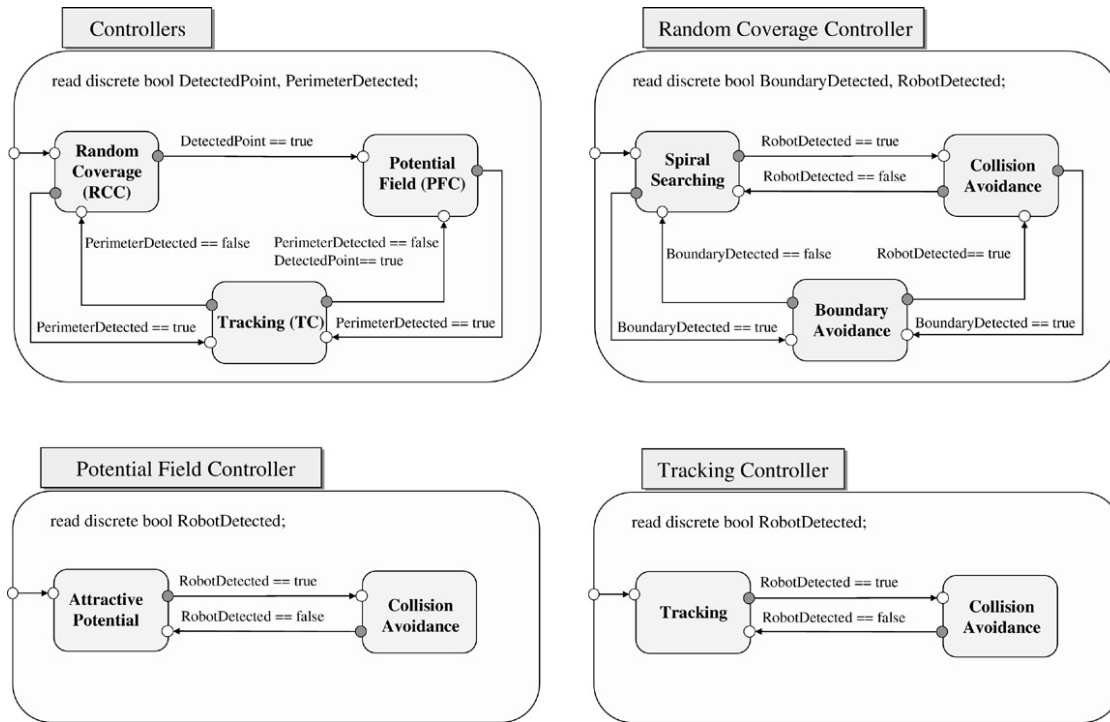


Figure 2.8: Hybrid controller for perimeter detection and tracking. The finite state machine integrates: an area coverage controller, a tracking controller, and a potential field controller. Source [Clark and Fierro, 2007].

the robot localization and the sampled points. This implies a necessity to use a large number of robots, because the boundary evolves making the older sampled points less relevant to the estimation of the current shape. A recent work [Turgeman and Werner, 2017] studies the integration of simultaneous boundary and source tracking.

2.3 Mapping the boundary shape

Once the robots are sampling the boundary using a tracking algorithm, the objective of the works is to estimate the boundary shape by using the collected point-wise measurements. Usually, this point-wise measurements come from the location of the robot, for instance, using 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 because the number of mobile sensors is finite. For this reason, mapping a dynamic boundary as a closed curve is still a challenge.

A preliminary work was presented by Kemp et al. [2004]. They represent the boundary curve by n points (where n is the number of robots) and use an *snake*-based algorithm (a well-known technique in computer vision) to drive the

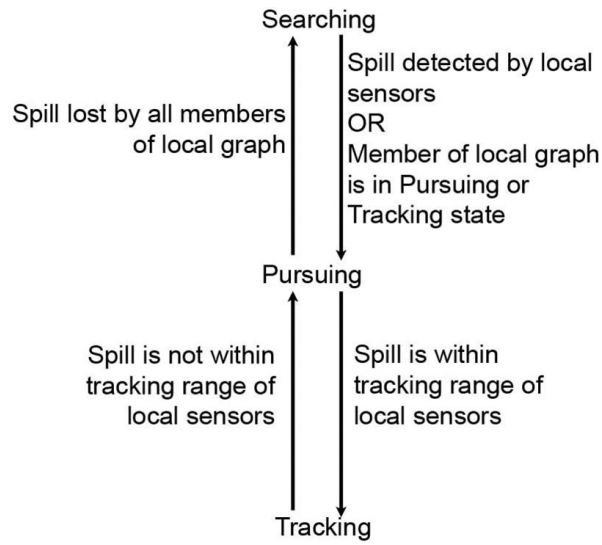


Figure 2.9: State machine by Zhang et al. [2013].

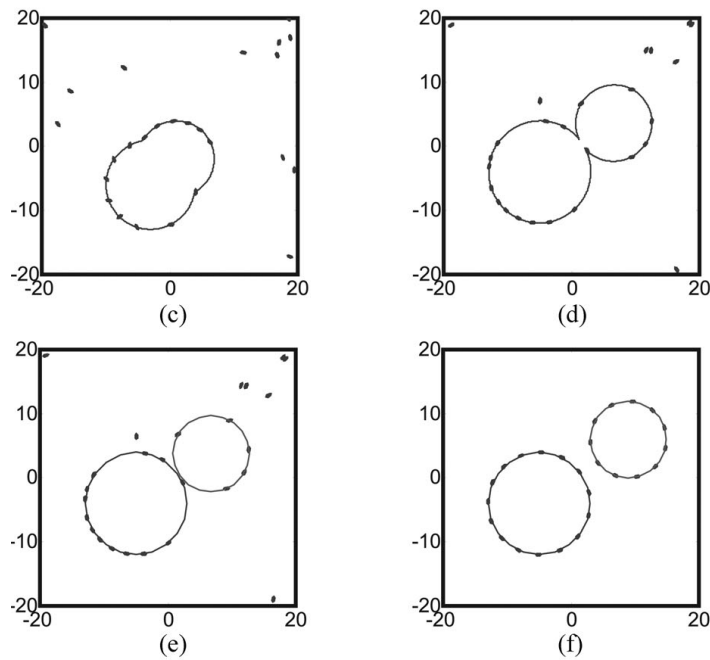


Figure 2.10: Simulated results for splitting boundaries. Source [Zhang et al., 2013].

robots towards the perimeter in a distributed manner. In [Susca et al., 2008, 2009], approximation theory of convex bodies is used to estimate the boundary by a polygon with a fixed number of vertices. Their estimation converges to the actual boundary by increasing the interpolation points, but it requires many robots to check and update the vertices. Their method use robotic sensor network to approximate a convex static contour with a polygon in order to track it. A gradient descent control law is applied to approximate a polygon and reduce a defined metric. That metric counts the remaining

area of the estimation that does not belong to the actual bounded region, plus the area that was not covered by the estimation. A formal definition of the metric is given in Section 5.4. The authors state that their method works for slow-moving boundaries, but their analytical development and simulations do not prove that claim.

2.4 Predicting the boundary behavior

In this category, the objective is not only mapping the boundary shape, but also predicting how the boundary changes in time. Jin and Bertozzi [2007b] assume that the boundary shape can be approximated by an ellipse. In their centralized method, the team of robots act as single observer that estimates the boundary dynamics using a Bayesian filter. In a posterior work, the robustness of the technique is shown by an implementation on actual robots [Joshi et al., 2009]. Another centralized method was proposed by White et al. [2007], where multiple aerial vehicles sample clouds in 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 only is a feasible solution 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 robots with range sensors. The method by 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 sample data to the sink is predicted. The contour is modeled as a non-closed curve with drifted Brownian motion. They consider error in the measurements of the boundary, and the curve estimation is made by interpolating the filtered sampled points.

2.5 Discussion

Searching, tracking and estimating the shape of environmental boundaries has been a highlighted topic in robotics since the last decade. There is a variety of applications where robotic sensors have a relevant impact, specially when monitoring phenomena such as oil spills [Bertozzi et al., 2005; Li et al., 2014; Fahad et al., 2015], forest fire [Casbeer et al., 2005], chemical leaks [Bruemmer et al., 2002], and harmful algae blooms [Jin and Bertozzi, 2007a]. In such environments where the anomalies are growing and

moving progressively, having the ability to forecast the affected areas is a very desirable feature to be included in an autonomous monitoring system.

As we have presented in this chapter, the tracking and estimation processes are directly related. On the one hand, some early works focus on detecting static boundary shapes and more recent ones on dynamic shapes. On the other hand, a set of works focus specifically on the tracking process. However, there are still some relevant open problems yet to be studied and assumptions to be lifted in this field.

Complex coordination problems are also scarcely studied like: how to coordinate the searching and tracking processes for multiple boundaries, or how to model and fuse information from heterogeneous robots. There is a lack of techniques that exploit the interaction among different correlated processes like searching and tracking. Current methods [Clark and Fierro, 2007; Zhang et al., 2013] model the whole problem as the combination of independent control modules. However, the synergy among the states could be improved as we shall propose in Chapter 3.

As it was described in the related works, predicting environmental boundaries has not been extensively studied in the literature. Few works like [Duttagupta et al., 2011] and [Jin and Bertozzi, 2007a] present incipient ideas in this field. Most of the related works approximate the boundary by interpolating robot locations or sample points. Nonetheless, when the boundary is approximated by a polygon, each vertex is updated as the robot moves close to it. Good approximations can be achieved when many robots are distributed along the boundary, since every vertex has to be quickly updated by the robots. However, when a single robot is used, each vertex is only updated after the completion of a lap and therefore, in the case of time-varying boundaries, the error in this method will not converge to zero. We deal with this issue in Chapter 4. The main problem of tracking a boundary with a single robot is the need of high speed motion to complete cycles around the boundary periodically. We propose a new approach to track and to predict the boundary with multiple slower robots in Chapter 5.

Chapter 3

Detecting and Tracking Multiple Dynamic Boundaries

In this chapter, we propose a combined approach for the effective exploration under uncertainty, the anomaly tracking, and the autonomous on-line allocation of agents. Robots explore the workspace maintaining the history of the sensed areas to reduce redundancy and to allow for full-map coverage. When an anomaly is detected, a robot autonomously determines to either track the anomaly or to continue the exploration of the environment, depending on the size of the anomaly, which is estimated by the length of the perimeter of the enclosing polygon.

As we have described in Chapter 2, our work falls into the category of hybrid approaches that integrate detection and tracking, since we propose a probabilistic approach that integrates full-map coverage and tracking of multiple dynamic anomalies with a team of robots. In the searching phase, robots are guided toward the spots where the presence of an anomaly has a high probability (Section 3.2). When an anomaly is detected, the robots autonomously determine to either track the anomaly or to proceed with the exploration of the environment (Section 3.3).

3.1 Problem Statement

Consider a team $\mathcal{R}$ of n robots, located in an environment defined in the Euclidean space $\mathbb{R}^2$. Let $\mathcal{W} \subset \mathbb{R}^2$ be a convex area that represents the workspace for all robots in the environment. Within this workspace, at time t a set of m anomalies may appear and dynamically evolve. We denote the set of anomalies by $\Omega_t^{[i]} \subseteq \mathcal{W}$, $i = 1, 2, \dots, m$, where $\Omega_t^{[i]}$ is the i th anomaly at time t . These anomalies are modeled as compact areas with nonempty interior, and it is assumed that there are no intersections among

them, that is, $\bigcap_l \Omega_t^{[l]} = \emptyset$. Let $P_t^{[l]}$ be a polygon defined by a sequence of points $\{p_1, p_2, \dots, p_l\}$, $l \geq 3$, which enclose the anomaly $\Omega_t^{[l]}$.

The anomalies can change their size, shape, and position over time, but we assume that they have a slow-moving behavior in comparison to the robots' velocity. The problems that we study are stated as follows.

Problem 1 (Full-map coverage for anomaly detection): How to coordinate a team $\mathcal{R}$ of robots to explore the full-map $\mathcal{W}$ so they detect and enclose each anomaly $\Omega_t^{[l]}$, with a polygon $P_t^{[l]}$.

Problem 2 (Robot allocation for tracking and exploring tasks): How to allocate robots (possibly all n) either to track multiple dynamic anomalies or to explore. This decision should depend on each anomaly size (e.g on the perimeter of the enclosing polygon), such that larger anomalies require more robots for tracking.

3.1.1 Available Information

Each robot $r_j \in \mathcal{R}$, $j = 1, 2, \dots, n$ has sensing capabilities to acquire local information, including its pose, and the concentration levels of the anomaly in the environment. The robot r_j can measure the anomaly concentration $c(t, r_j)$ and the unitary tangent vector T at its location. The points in the anomalies obey the constraint $c(t, r_j) > \tau$, with $c(t, r_j) \in \cup_i \Omega_{ti}$, where τ is a threshold.

In addition, we assume that each robot can communicate with all other robots of the team by broadcasting messages. Therefore, at time t , each robot can receive the samples of all other robots or polygons, if detected. Additionally, each robot can also broadcast through the communication network its approximated polygon.

3.1.2 Robot Model

Our model assumes that every robot r_i is non-holonomic, and its pose is represented by the vector $(x_i, y_i, \theta_i)^T$. We use the *unicycle control model*, which assumes that a robot may be controlled by issuing both the rotational and the translational velocity $u_i = [v_i, \omega_i]^T$.

3.2 Searching for anomalies

In this section, we propose a method to solve Problem 1. We treat the anomalies in the environment as a multi-modal distribution that can change dynamically at each time step. Then, our method tries to incrementally improve the estimation of the

distribution by merging robot measurements and simulating the random behavior of the anomaly. A modification of the particle filter technique [Mottaghi and Vaughan, 2006] is developed since it offers a probabilistic method that converges to multi-modal probability distributions. Robot navigation and coordination are based on the uncertainty about anomalies, with every robot trying to visit the nearest spot with highest likelihood of being an anomaly.

There are three relevant reasons for using particle filter in this context. *(i)* If an anomaly exists in the environment, exploration with multiple robots leads the particles to converge to the shape of the anomaly, otherwise the robots will continue to explore the environment which tends to reduce the uncertainty of the occurrence of an anomaly along the traversed path. *(ii)* Each robot has an independent representation of the updated map, based on the robot's measurements and the communicated information from other robots. The set of particles offer a representation of the areas in the map which have been recently covered, incorporating the uncertainty in the non-visited zones and reducing the belief of the sensed path in time. This updating process could be modeled as a complex geometric process, but under this approach, it can be computed with simple arithmetic operations on the particle's motion. *(iii)* The natural, randomized movement of the anomaly is incorporated in the particle's motion. Any additional information on the anomaly can be added as *a priori* data in order to improve the estimation updating of the map.

A particle can be considered as a point on a map, and it is associated with the probability of having an anomaly in that location. Each robot has its own set of particles $\mathcal{X}_t = \{x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[N]}\}$, which represents its belief over the possible anomalies in the environment. An accumulation of particles in a region may indicate a high probability of having an anomaly there. As a consequence, the robot will try to pursue it in order to reduce the uncertainty at that specific location.

At the initial state, robots start searching the area without previous knowledge of the existence or position of any anomaly. Therefore, the N particles are spread based on a uniform distribution over $\mathcal{W}$. The particle filter is an iterative process that requires two fundamental models to be defined: the motion and the updating of the particles.

3.2.1 Motion model for particles

A random motion model displaces a particle along a random distance from time $t - 1$ to t based on a Gaussian distribution. The particle's motion is defined by a (2×2) covariance matrix Σ in order to capture the spreading velocity of the anomaly.

Equation (3.1) describes the motion model for particle k , $k = 1, 2, \dots, N$ based on the last estimation, $x_{t-1}^{[k]}$, and a random motion with normal distribution $\mathcal{N}(0, \Sigma)$:

$$p(x_t^{[k]} | u_t, x_{t-1}^{[k]}) \sim \mathcal{N}(x_{t-1}^{[k]}, \Sigma). \quad (3.1)$$

This model is used when no additional information about the behavior of the anomaly is available. In this case, a random motion is assumed. Although in this work no *a priori* information about the dynamic behavior of the anomaly is used, this information may help to enhance the exploration process as long as the particles simulate that behavior. For example, a fire in a forest evolves in an apparently random fashion, but if the wind is blowing, the fire will tend to spread in the direction of the wind, and if the wind's strength and orientation is known, a better estimation of the evolving anomaly may be obtained.

The estimation of the anomaly's distribution is updated using the robot's sensor readings. At each iteration, every particle is re-sampled based on weights. The following equation determines how the weight of a particle k is updated based on sensor readings,

$$w_t^{[k]} = \begin{cases} a, & \text{if } c(t, r_j) < \tau, \\ b, & \text{if } c(t, r_j) \geq \tau \vee x_t^{[k]} \in \cup_i P_i, \\ 0, & \text{if } x_t^{[k]} \notin \mathcal{W}, \\ 1/N, & \text{outside sensor range,} \end{cases} \quad (3.2)$$

where the constant $a > 0$ is a small value that represents a low probability of the existence of an anomaly at a given point when no anomaly is sampled ($c = 0$); $b \geq 1$ is a value used to increase the particle's weight given that an anomaly is sampled or particle is inside an *available-polygon* $x_t^{[k]} \in \cup_i P_i$; value 0 is set if the particle is outside the map $x_t^{[k]} \notin \mathcal{W}$; and for a point not sampled at time t , a normalized value $1/N$ is set. In our experiments, we used $a = 0.1$ and $b = 1.3$.

When each particle has its own weight, the group of particles is re-sampled with replacement to randomly clone particles proportionally to their updated weight [Hol et al., 2006]. These unsampled particles are then moved according to Equation (3.1). After several iterations, the result is an accumulation of particles in places with higher possibility of the existence of a boundary. When no anomaly is detected in the environment, the robots will try to navigate towards the areas with particle accumulations in order to visit the spots with large probability of finding anomalies.

We define two working modes: The exploration and tracking modes. When a robot does not detect an anomaly locally, it starts to explore the environment by

pursuing particles in order to visit places more likely to contain anomalies.

3.2.2 Exploration

The exploration process with multiple robots must attain the following objectives: (1) maximize the number of visited particles, giving priority to the nearest one, (2) maximize the distance to the other robots, (3) maximize the distances to the obstacles and to the map borders. Therefore, we use potential fields to navigate in the continuous space [Koren and Borenstein, 1991]. This well known technique is based on the physical model of electrical charges, assuming that a robot is a positive electrical charge which is attracted by all the negative charges. Other robots and obstacles are also modeled as positive charges that repel the robot. Therefore, the robot's velocity and orientation are computed as the vector sum of all forces involved.

The forces acting on robot r_j are generated by particles $x \in \mathcal{X}$, other robots $r_k \in \mathcal{R} \setminus \{r_j\}$, $\forall k \neq j$, and obstacles $o \in \mathcal{O}$ (map boundaries are also introduced and are also modeled as obstacles). Equation (3.3) computes the resulting force magnitude for any entity $z = \{x, r_k, o\}$ with constants of proportionality β_x , $\beta_{r'}$, and β_o respectively. The orientation of vector F_z is computed as the angle between the location of the entity z and the robot r_k , and its magnitude by

$$|F_z| = \frac{\beta_z}{d(r_j, x)^2}. \quad (3.3)$$

The total force F_T is computed as the result of all the attraction forces generated by all particles and the repulsion forces due to obstacles and other robots,

$$F_T = \sum_{x \in \mathcal{X}} F_x - \sum_{r_k \in \mathcal{R} \setminus \{r_j\}} F_{r_k} - \sum_{o \in \mathcal{O}} F_o. \quad (3.4)$$

Then, the controller $u_i = [v_i, \omega_i]^T$ follows the total force vector F_T , maintaining the linear velocity proportional to the vector magnitude:

$$v_i = \begin{cases} K_v |F_T| & \text{if } K_v |F_T| \leq v_{max}, \\ v_{max} & \text{otherwise,} \end{cases}$$

where $K_v > 0$ is a gain constant and v_{max} is the maximum velocity that the robot can achieve. The angular velocity is driven by the vector orientation ϕ as:

$$\omega_i = K_\phi (\phi - \theta_i),$$

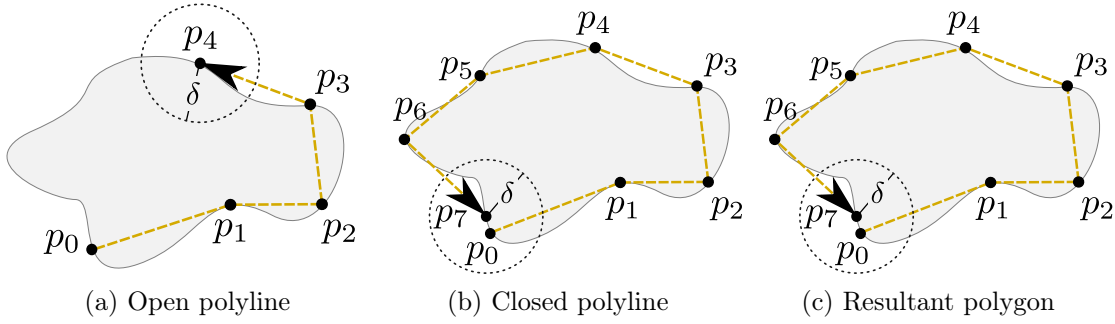


Figure 3.1: Processing robot path to identify a polygon. The arrowhead represents the robot location, and the points represent the samples of the boundary along the robot path.

with a gain constant $K_\phi > 0$.

3.2.3 Tracking an anomaly

When a robot r_i detects an anomaly $c_i > \tau$, it enters in its tracking mode to move around the anomaly. The control of its linear speed is based on the distance to the other robots in order to avoid collisions, and the angular speed is computed based on the tangential angle φ ,

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} v_{track} - \frac{K_{track}}{d(r_i, r')} \\ K_\varphi(\varphi - \theta_i) \end{bmatrix}, \quad (3.5)$$

where $K_{track} > 0$ and $K_\varphi > 0$ are gain constants, v_{track} is the desired linear speed to track the boundary, K_{track} is a proportionality constant, and r' is a robot in front of robot r_i . The function $d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ is just the Euclidean distance between both robots and its objective is avoiding collisions in the tracking process.

3.2.4 Estimating the enclosing polygons

Our method tries to approximate the dynamic anomaly $\Omega_t^{[i]}$ by a polygon $P_t^{[i]}$. When an anomaly is detected by a robot r_j , ($c_j > \tau$), it enters in the tracking mode and creates a polyline $Q_t^{[j]}$. For each new sample c_j , the polyline will add the location of the sensed value to the polyline $Q_{t-1}^{[j]}$,

$$Q_t = Q_{t-1} \cup \{r_j(t)\}. \quad (3.6)$$

At each time step, we check if the polyline closes, that is, if a polygon has been formed by that polyline. In the verification, if Q_t generates a polygon is accomplished

by comparing the distance of the last sampled point p_n to all the other lines. The resulting sequence to generate a polygon is restricted to be ordered in counter-clockwise manner. If the last point p_l is close ($< \delta$) to line (p_j, p_{j+1}) , Q_t is updated to $Q_t = \{p_{j+1}, p_{j+2} \dots p_l\}$. For instance, Figure 3.1a shows an open polygon. The point p_4 is very far from the other edges. The robot proceeds on tracking and sampling until the polyline closes. Figure 3.1b depicts the case where the first point p_0 is close to the edge (p_0, p_1) , then, point p_0 is deleted and the edge (p_7, p_1) is added in order to close the polygon.

The parameter δ is related to how fast the robot can complete a cycle. However, it must be defined carefully because a larger value enclose the polygon too early and consequently failing on the intended task. In contrast, a short value could not identify the polygon when the anomaly is shrinking or expanding.

3.3 Multiple Anomaly Support

Most of the state of the art techniques are focused on tracking a single anomaly. The existence of multiple regions of interest involve coordination issues, task allocation and complementary sensor fusion. For instance, allocating many robots to enclose a perimeter is a problem for most of the approaches, because this creates redundancy in the sensed area and reduces the perception for new anomalies.

3.3.1 Polygon estimation

Each robot continuously communicates the value of a sensed point or a polygon, if the last was obtained. If the robot has a polygon, then it broadcasts the simplified polygon data and the time of its detection. This shared information is fused with local information as follows.

3.3.1.1 Fusing polylines

Robot r can locally estimate the polylines of the other robots by receiving the punctual samples and applying Equation (3.6) for each one independently. Those polylines can be merged to estimate a polygon. To merge the polyline $Q_t^{[j]}$ and the polyline of other robot $Q_t^{[k]}$, $j \neq k$, we check if one of the edges in $Q_t^{[j]}$ intersects with one of edges in $Q_t^{[k]}$ by cross-validation. If an edge $(p_i, p_{i+1}) \in Q_t^{[j]}$ intersects with an edge $(p'_{j+1}, p'_j) \in Q_t^{[k]}$, at the intersecting point p_{int} , then the merge polyline, which is computed by

$$Q_t^{[j]} = \{p'_0, p'_1, \dots, p'_j\} \cup \{p_{int}\} \cup \{p_{i+1}, \dots, p_l\}. \quad (3.7)$$

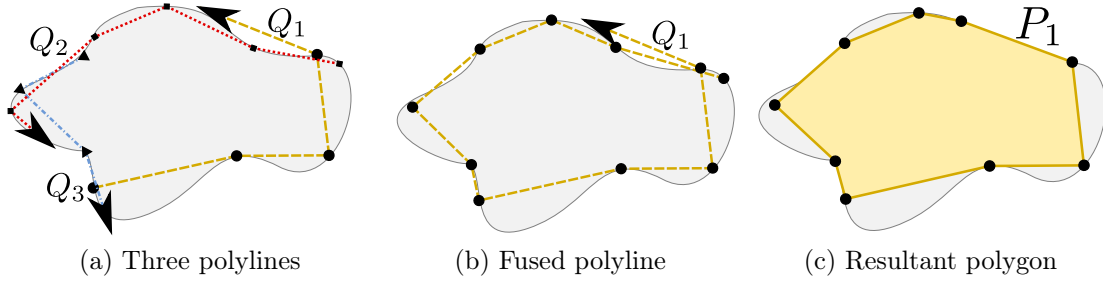


Figure 3.2: Fusioning three polylines, Q_1 , Q_2 , and Q_3 , from three different robots.

The new merged polyline is replaced by the current polyline only if the length of the merged polyline is larger than the current one. Then the new polyline is checked to identify if it closes. This process is repeated for each polyline. Figure 3.2a illustrates an example with three polylines from three different robots, which are merged by applying Equation (3.7) twice to get a larger polyline Q_1 (Figure 3.1b), which finally obtains polygon P_1 of Figure 3.2c.

3.3.1.2 Using shared polygons

Each robot compares its samples with the measurements sent by the other robots in order to take advantage of a previously detected polygon. Polygons shared in the network have two characteristics: (1) They are simplified to drastically reduce the number of points to be shared by applying the Ramer-Douglas-Peucker algorithm [Douglas and Peucker, 1973], which deletes redundant points based on a pre-defined distance threshold, and (2) each polygon includes a tag *first-detection-time* so it may have priority in robot allocation.

The metric to determine the required number of robots η_i to track an anomaly $\Omega_t^{[i]}$ is defined by

$$\eta_i = \lceil \kappa \sum_{i=1}^k d(p_i, p_{i-1}) \rceil, \quad (3.8)$$

where κ is a constant that determines the number of robots required per unit of length. Therefore, we tag the polygon as *available-polygon* if the number of robots tracking the anomaly is less than η_i , and *full-polygon* otherwise.

If the distance from an *available-polygon* (shared by other robot) is less than a threshold, then we merge the point to polygon, and use the previously described approach. Let P'_t be the polygon received by other robot r_k , and (p'_j, p'_{j+1}) be the

nearest edge to the robot r_j . The resulting polygon for r_j is computed as

$$P_t = \{p'_{j+1}, \dots, p'_l\} \cup \{p'_0, \dots, p'_j\} \cup \{p_0\}. \quad (3.9)$$

3.3.2 Multi-Robot coordination

We use the particles to represent and to update the world model. In order to determine if a polygon is open or closed for a robot, we use Equation (3.8) and attract η_i robots by increasing the particles according to Equation (3.2). However, if the anomaly increases in size, then η_i value will be augmented, and the tag will change to *available-polygon* to attract other robots. The case when the anomaly reduces its size is more complex. For a shrinking anomaly, Equation (3.8) gives a smaller number which will cause some of the robots to stop tracking and start exploration. Choosing which robots will start exploration and which will continue tracking the anomaly is determined by comparing the *first-detection-time* of the polygon. In polygon with unnecessary robots, the recent robots, that identified the polygon recently, tag the polygon as *full-polygon* and spread their particles randomly across the map. To avoid robots to be attracted by *full-polygons*, Equation (3.2) is extended with the clause $w_t^{[i]} = 0$ if the particle $x_t^{[i]}$ is within a *full-polygon*.

3.4 Experiments and Results

Three validation scenarios have been used with three types of anomalies: Static, expanding and shrinking. We analyze the accuracy of the estimated area using a metric that is the difference between the estimation and the actual one. A formal definition of this metric is described in Section 5.4.

We initially conducted experiments in the ROS-Gazebo 3D simulator, and then with real robots. Real and virtual robots are assumed to be the *iCreate* base, differential mobile platform equipped with: Onboard-computer, communication module IEEE 802.3.11 (WIFI), and an RGB camera. Figure 3.3 shows the Gazebo environment with eight robots, and those that are tracking the anomaly (represented by a rug with a texture of fire). Our software for real and virtual robots is open source and is available¹.

¹The source code is available at https://github.com/dsaldana/roomba_sensor_network

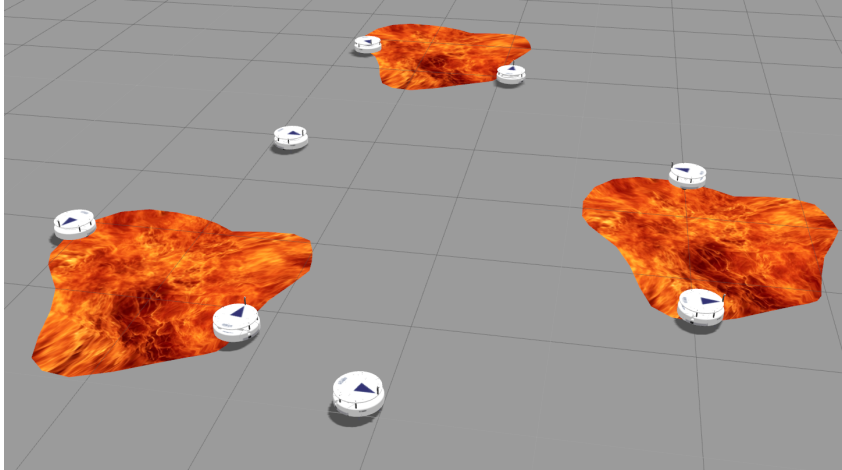


Figure 3.3: Simulation with six robots tracking an anomaly and two robots exploring the environment.

3.4.1 Simulations

An important factor that needs to be taken into account is the number of points that conforms to the detected polygon and that will be broadcasted. The error $\epsilon_t = \delta$ increases linearly with the applied threshold in the *Ramer-Douglas-Peucker* algorithm. However, the number of required points decreases exponentially. We used a non-convex static anomaly with $area = 3.49m^2$ and $perimeter = 7.15m$, which was sampled with 186 points by a single robot. Using a $threshold = 0.1m$, the number of points are drastically reduced to 10 points (e.g. 95% of the original number of points) with error $\bar{\epsilon}_t \approx 0.23m^2$.

The simulated environment is defined by a map with dimensions of 5×4 meters. In the initial setup, eight robots are distributed over the map. We analyze the following three scenarios.

3.4.1.1 Static anomalies

In this scenario, the robots begin in a empty environment, and then three static anomalies turn up in different places at different times $t = 40s, 160s, 280s$ respectively. Figure 3.4a illustrates how the robots are allocated to the tracking task along the time. The continuous line represents the exact number of robots that must be allocated (the robots do not know this information). When the first anomaly appears, the number of robots increases progressively until the polygon is closed at time $t = 100s$, as described in Section 3.3. The required number of robots to cover the first anomaly is computed by Equation (3.8), and the remaining robots proceed to continue the exploration. We can see this kind of behavior when a new anomaly appears in the environment (times

$t = 220s$ and $t = 360s$). It can be seen in Figure 3.4a three initial peaks in the tracking robots, which are reduced to one after the anomaly is detected. Since the anomalies are static, the error ϵ_t is proportional to the total area comprised by the anomalies. Figure 3.4b shows the detected polygons at different times.

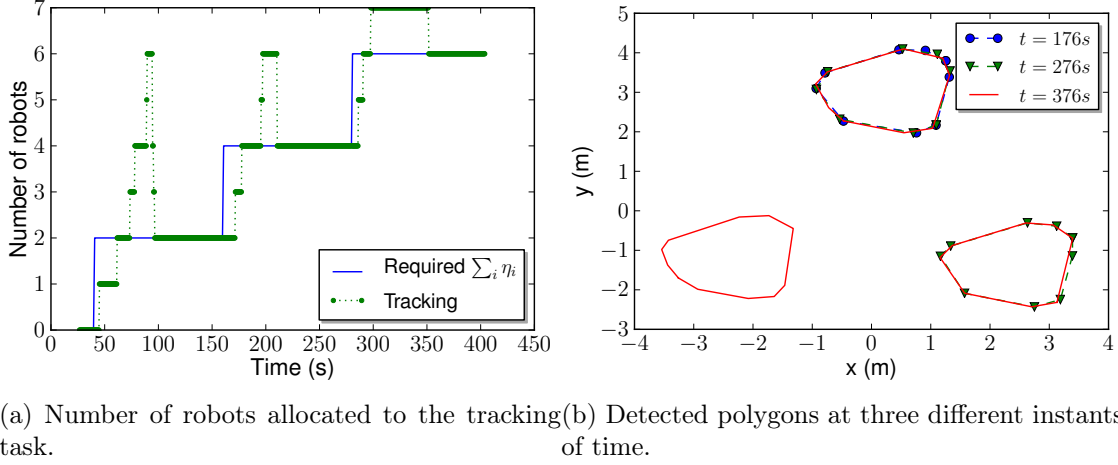


Figure 3.4: Allocation of robots to three static anomalies that appear at three different times.

3.4.1.2 Expanding

In the second scenario, two anomalies show up at different locations of the map. They start to expand in different directions and with different velocities. Figure 3.5a shows that while an anomaly is growing, the area is sub-estimated and the error ϵ_t is proportional to the velocity of expansion. Figure 3.5b illustrates the heterogeneous expansion along the time.

3.4.1.3 Shrinking

In the third scenario, the same two anomalies shrink until they stabilize to get a constant area. As the anomaly is tracked in counter-clockwise manner, the controller generates movements to the right-side but the anomaly is moving to the opposite direction. This continuous rectification in orientation generates an oscillatory movement that is reflected in the error ϵ_t , as it can be seen in Figure 3.6a. Similarly in the expanding scenario, the ϵ_t is affected by the velocity of the anomaly behavior and by the rectification of the tracking controller.

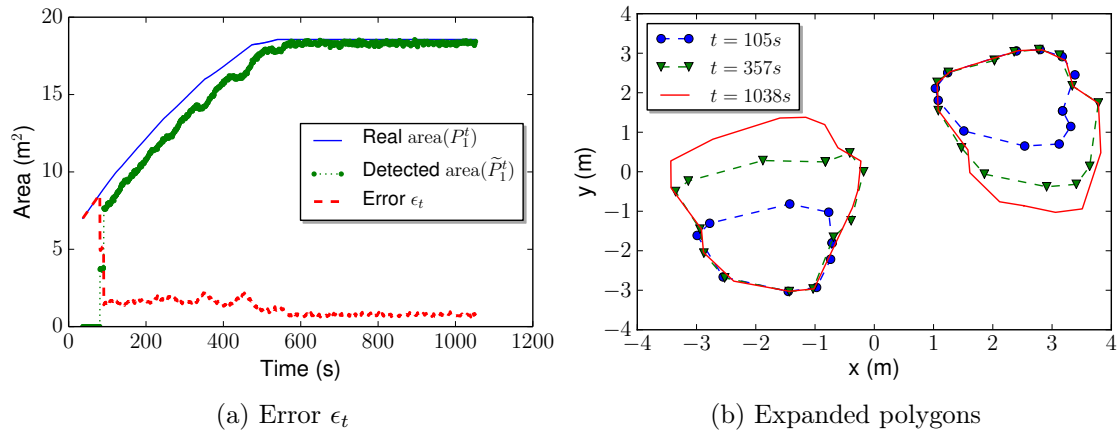


Figure 3.5: Two dynamic anomalies expanding their area.

3.4.2 Experiments with real robots

Our testbed is composed of a map with 3×1.8 meters and three robots (Figure 3.7). Localization is acquired by processing images of an overhead camera with a wide-angle lens. Dynamic anomalies are computer generated and projected on the floor.

On the one hand, the reduced space limited the deployment and movement of a larger number of robots in the environment. On the other hand, the time to find all anomalies and to generate enclosing polygons is reduced. In these experiments, we found that the detecting process speeds up in comparison with simulations because of the small size of the workspace and the anomalies, but the precision of our sensor is adequate to the limited quality of the estimated polygons.

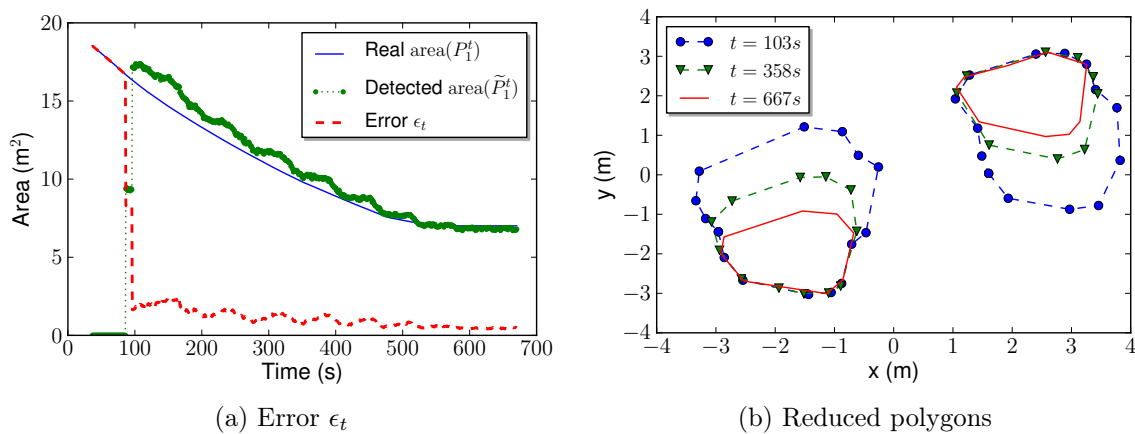


Figure 3.6: Two dynamic anomalies reducing their area.



Figure 3.7: Experimental testbed with three actual robots and two anomalies.

3.5 Discussion

In this chapter, we proposed a probabilistic distributed coordination method to detect and track multiple dynamic anomalies using a robotic sensor network. Two main problems were tackled, *(i)* full map coverage to find anomalies, and *(ii)* multi-robot coordination to track multiple dynamic anomalies. We have experimentally shown, both in simulation and with real robots, that our approach can simultaneously detect regions out of normal conditions. Experimental results demonstrated that the proposed method works in different scenarios, where autonomous robot allocation is effectively executed based on the regions which require more attention.

An important issue in scalability is the message broadcasting. It works for multiple robots, but communication may become burdensome for larger number of robots.

Chapter 4

Predicting Boundary Behaviors with a Mobile Robot

Predicting the behavior of the boundary of anomalies, specially those related to dangerous environmental phenomena, like a spreading fire or an oil spill, provides relevant information to mitigate the environmental problem or even to support evacuation actions in order to save human or animal lives. In this chapter, we present a model that uses a single robot moving around an environmental boundary in order to predict its shape by an analytical continuous function, based on the combination of polynomial approximation and Fourier series. Our method converges to the exact boundary when we increase both the sample frequency and the robot velocity. In order to evaluate the estimation quality, we performed experiments with simulated and actual robots. We applied our model in some dynamic boundaries presented in the literature, as in the application of plume-front estimation, showing that it produces accurate results with significant improvement in most cases when compared to current (state of the art) techniques.

As an illustration of the problem, Figure 4.1 shows the evolution of a simulated dynamic boundary at times $t = 0, 100, 200$. This boundary undergoes several simultaneous transformations such as translation, shrinking and expansion. Starting at time $t = 0$ as a circle, it is successively deformed to assume a flower like shape at $t = 200$. The robot moves along the boundary from time $t = 0$ to $t = 100$ sampling its location. At each time t , the robot can only track a single point on the curve. The sampled points are represented by small dots connected by the smooth spiral-shaped curve representing the robot's trajectory up to $t = 100$. The objective is to use the sampled collected points to estimate the entire boundary, including its forecasting in future moments. For instance, at time $t = 100$, we want to estimate its present entire

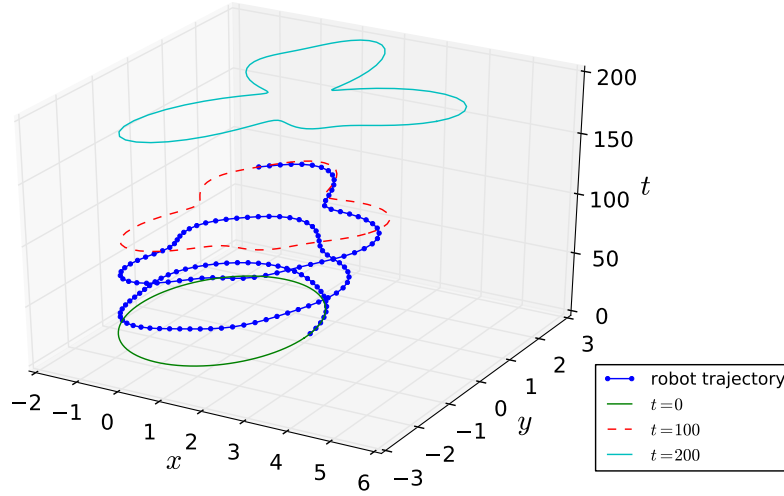


Figure 4.1: Dynamic boundary and the robot trajectory. The desired shape to be predicted is represented by the dashed line.

red dashed curve as well as to predict its future star-shaped curve at time $t = 200$.

4.1 Problem Statement

We are interested into a region in the environment where there is a phenomenon delimited by a perimeter. This region of interest $\Omega_t \subset \mathbb{R}^2$ is a connected set with finite area, indexed by time $t \in \mathbb{R}$, and enclosed by a dynamic boundary defined as follows.

Definition 1. A *dynamic boundary* is a set of planar points $\partial\Omega_t$ such that for each point $p \in \partial\Omega_t$, and for any arbitrarily specified $\xi > 0$, the open disc centered at p with radius ξ contains points of Ω_t and its complement Ω_t' .

We assume that the dynamic boundary $\partial\Omega_t$ is a simple closed curve (also called Jordan curve), which can be modeled by a parametric function as

$$\partial\Omega_t = \{\gamma(t, s) \mid s \in [0, 1]\},$$

where the image of the function $\gamma : \mathbb{R}_{\geq t_0} \times [0, 1] \rightarrow \mathbb{R}^2$ is a curve in $\mathbb{R}^2$, mapped by the parameter $s \in [0, 1]$, such that $\gamma(t, 0) = \gamma(t, 1)$ and the restriction that $\gamma(t, s)$ is an injective function of $s \in [0, 1)$ for a fixed time t . Hence, $\partial\Omega_t$ is a continuous loop with no self-intersecting points. Additionally, we assume that the function γ changes smoothly in time t .

To sample the dynamic boundary $\partial\Omega_t$, we use a robot whose configuration is represented by the vector $r = [x, y, \theta]^T$, where x, y define the coordinates in the Euclidean space and θ is the robot's orientation.

At each time step t , the robot can observe its location (x, y) , and estimate the curve parameter s (more details about the s -estimation in Section 5.2). Therefore, the robot data, in a finite time interval, is a set of k samples

$$\mathcal{D} = \{(t_i, x_i, y_i, s_i) | i = 1, \dots, k\}.$$

We want to estimate the past and the current boundary shape, as well as to predict its future shape based on the data $\mathcal{D}$:

Problem 1. *Given a set of samples $\mathcal{D} = \{(t_i, x_i, y_i, s_i) | i = 0, 1, \dots, k\}$ by a robot moving along a dynamic boundary $\partial\Omega_t$ in counterclockwise manner, how to estimate the past boundary for $t_0 \leq t \leq t_k$ and to predict the future $\partial\Omega_t$ for $t > t_k$.*

To solve this problem, we first propose in Section 5.2 an algorithm to estimate the arc-length parameter s . Next, in Section 4.4, we propose a model to solve Problem 1.

4.2 Robot trajectory around the dynamic boundary

This section describes how to move the robot around a given boundary function γ , which is very useful for the simulations and *dynamic perimeter surveillance* [Saldana et al., 2016]. An application of this algorithm and some experiments can be seen in the accompanying video¹

In order to track the perimeter $\partial\Omega_t$, the robot r circulates along the dynamic boundary in counter-clockwise manner with a desired velocity v . At $t = t_0$, the robot r starts at location $\gamma(t_0, s_0)$, where the constant $s_0 \in s$ maps to an arbitrary point of the boundary. Now, we define a function describing the trajectory of each robot along the boundary.

Definition 2. *The **trajectory of a robot** along the boundary is represented by the function $S(t)$, where for each time t , there is a function $s(t) : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$ such that*

$$S(t) = \gamma(t, s(t)), \quad \forall t > t_0$$

in which, $s(t)$ maps the curve parameter s to the robot position. It satisfies the initial condition $s_0 = s(t_0)$.

¹Video link: <https://www.youtube.com/watch?v=Ug5Ne6v34iQ>

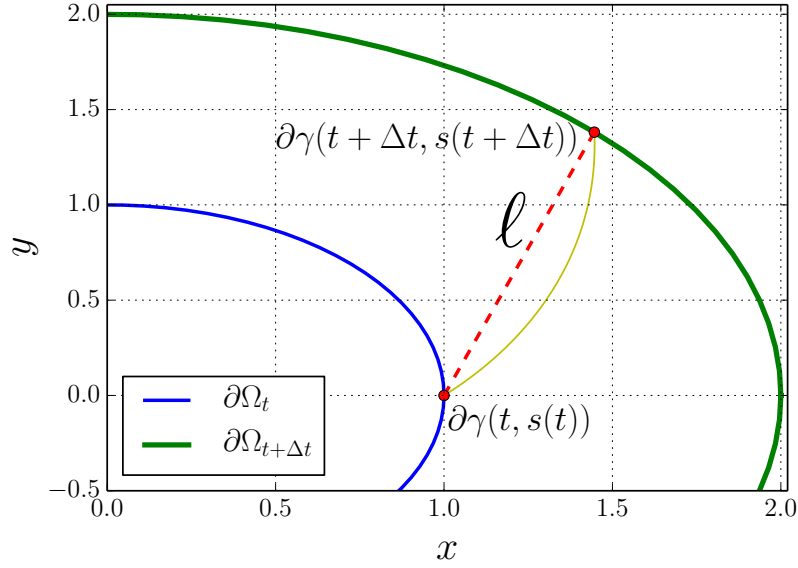


Figure 4.2: Robot trajectory (yellow line) in a short time interval $[t, t + \Delta t]$, it is approximated by a linear movement (red dashed line) for a little Δt . The blue curve is the boundary at time t , and the green circle is the boundary at time $t + \Delta t$. The robot starts at point $\gamma(t, s(t))$ and moves to the point $\gamma(t + \Delta t, s(t + \Delta t))$.

Proposition 1. *If a robot is moving with a desired velocity v in counter-clockwise direction along a dynamic boundary, described by the function $\gamma(t, s)$, then the robot trajectory $S(t) = \gamma(t, s(t))$ that starts at point in $\gamma(t_0, s_0)$, is such that $s(t)$ satisfies the ordinary differential equation*

$$\dot{s} = \frac{1}{x_s^2 + y_s^2} \left(-x_s x_t - y_s y_t + \sqrt{v^2(x_s^2 + y_s^2) - x_s^2 y_t^2 + 2x_s x_t y_s y_t - x_t^2 y_s^2} \right), \quad (4.1)$$

where $x_s = \partial x / \partial s$, $y_s = \partial y / \partial s$, $x_t = \partial x / \partial t$, and $y_t = \partial y / \partial t$.

Proof. We analyze the movement between time t and time $t + \Delta t$. It may be approximated by a straight linear movement for a very small Δt . Then between t and $t + \Delta t$, the robot moves a distance ℓ , as illustrated in Figure 4.2.

In this proof, we abuse the notation of x and y in order to represent separate coordinates of γ . Then we can rewrite the γ function as a combination of functions x , and y ,

$$\gamma(t, s) = \begin{bmatrix} x(t, s) \\ y(t, s) \end{bmatrix}.$$

The square distance ℓ^2 is calculated by the Pythagorean theorem,

$$\ell^2 = \Delta x^2 + \Delta y^2,$$

where $\Delta x = x(t + \Delta t, s(t + \Delta t)) - x(t, s(t))$ and $\Delta y = y(t + \Delta t, s(t + \Delta t)) - y(t, s(t))$. As the robot moves linearly with constant velocity v , then $\ell = v\Delta t$,

$$v^2 \Delta t^2 = \Delta x^2 + \Delta y^2.$$

Dividing in both sides by Δt^2 , we have

$$v^2 = \left(\frac{\Delta x}{\Delta t}\right)^2 + \left(\frac{\Delta y}{\Delta t}\right)^2,$$

which is the same in Newton's notation,

$$v^2 = \dot{x}^2 + \dot{y}^2. \quad (4.2)$$

Using the chain rule for $\dot{x}$ and $\dot{y}$, we obtain $\dot{x} = x_t + x_s \dot{s}$ and $\dot{y} = y_t + y_s \dot{s}$, respectively. Replacing $\dot{x}$ and $\dot{y}$ in (4.2),

$$v^2 = \left(x_t + x_s \dot{s}\right)^2 + \left(y_t + y_s \dot{s}\right)^2.$$

Expanding terms and simplifying we obtain,

$$v^2 = x_t^2 + 2\dot{s}x_s x_t + \dot{s}^2 x_s^2 + y_t^2 + 2\dot{s}y_s y_t + \dot{s}^2 y_s^2.$$

Solving for $\dot{s}$, we obtain two solutions. As the robot is moving in a counter-clockwise manner, the variable s is always increasing, then the solution is given by

$$\dot{s} = \frac{1}{x_s^2 + y_s^2} \left(-x_s x_t - y_s y_t + \sqrt{v^2(x_s^2 + y_s^2) - x_s^2 y_t^2 + 2x_s x_t y_s y_t - x_t^2 y_s^2} \right).$$

Hence, the robot's trajectory is computed by the function $S(t) = \gamma(t, s(t))$, that satisfies the initial condition $s_0 = s(t_0)$. $\square$

The first-degree differential equation above can be solved by numerical methods like *Runge-Kutta 4* [Kiusalaas, 2013]. Then, the starting point $\gamma(t_0, s_0)$ can be used as the initial condition in order to obtain an approximation of the function $s(t)$. Hence, the robot receives a boundary function γ with an initial location $s_0 \in [0, 1]$ in order to obtain its trajectory $S(t)$, by solving the differential equation from (4.1).

Minimum linear velocity to track the boundary

The minimum linear velocity, v_{min} , required to make the robot advance along the curve must satisfy $\dot{s} > 0 \forall t \in [t_0, t_f]$. Using (4.1) with $\dot{s} > 0$, we obtain

$$\frac{1}{x_s^2 + y_s^2} \left(-x_s x_t - y_s y_t + \sqrt{v^2(x_s^2 + y_s^2) - x_s^2 y_t^2 + 2x_s x_t y_s y_t - x_t^2 y_s^2} \right) > 0,$$

and isolating v , we obtain

$$v > \sqrt{\frac{x_t^2}{2} + \frac{y_t^2}{2}}.$$

Since the right side of the equation depends on time t , and the velocity is constant, the minimum constant velocity that allows a robot to move around the boundary is:

$$v_{min} > \max \left(\sqrt{\frac{x_t^2}{2} + \frac{y_t^2}{2}} \right). \quad (4.3)$$

For example, for the growing circle with boundary equation $\partial\Omega_t(t, s) = [t \cos(s), t \sin(s)]$, by applying (4.3), we obtain

$$v_{min} > \max \left(\sqrt{\frac{\cos^2(s)}{2} + \frac{\sin^2(s)}{2}} \right),$$

then, we obtain the minimum velocity $v_{min} > 1/\sqrt{2}$.

4.3 Estimating the curve parameter

During the robot trajectory, it takes m observations of its location (x, y) and sampling time t , producing the set $\mathcal{S} = \{(x_i, y_i, t_i) | i = 1, 2, \dots, m\}$. Figure 4.3 illustrates a trajectory for a robot moving with constant velocity along the boundary function $\gamma(t, s) = [0.1 t \cos(2\pi s), \sin(2\pi s)]^T$. The arc-length parameter s must be estimated for each sampled time in order to complete the robot data $\mathcal{D}$, which is used to analyze the behavior of the anomaly boundary $\partial\Omega_t$ (as it will be described in Section 4.4).

The work described in [Susca et al., 2008] presents a technique to estimate the s parameter for slow-varying boundaries with multiple robots and assuming that, at the initial time $t = 0$, the robots have an estimate of the boundary. In our approach, we avoid this assumption and use only the sampled points in the robot's trajectory.

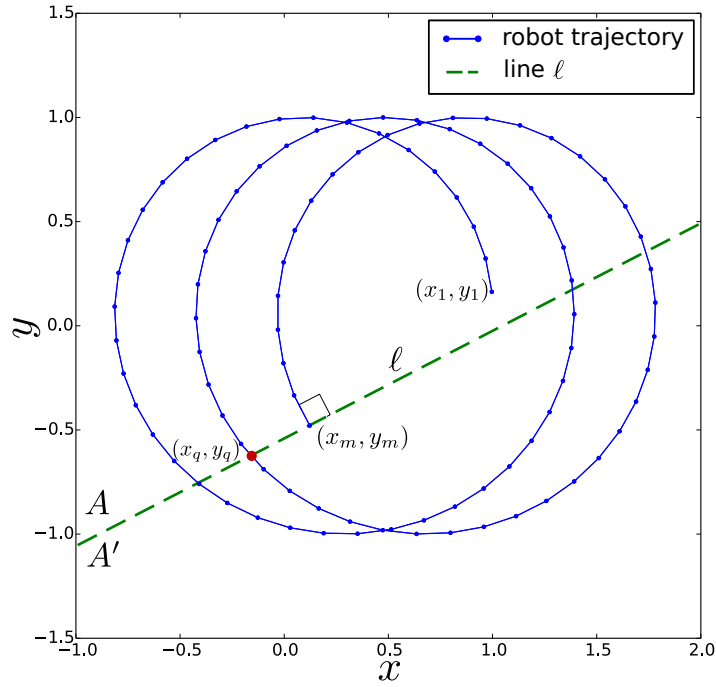


Figure 4.3: Trajectory of a robot that is moving with constant velocity $v = 2.6$ along the boundary $\gamma(t, s) = [0.1 t \cos(s), \sin(s)]^T$. The dots represent the sampled points.

4.3.1 Identifying cycles

As a requirement for estimating the s parameter, we need to identify the completed cycles in the robot trajectory. We assume that the boundary $\partial\Omega_t$ changes smoothly. If the robot leaves a mark on the boundary at its departure point, it can move along the boundary fast enough to reach the marked point in the evolving boundary at least once. Each of these laps will be called a *cycle*. The following algorithm identifies the completed cycles in the robot sampling data

1. Initially, we trace a line ℓ that passes over the last point (x_m, y_m) , and is perpendicular to the last line segment $\{(x_{m-1}, y_{m-1}), (x_m, y_m)\}$ of $\mathcal{S}$ (as exemplified in Figure 4.3 by the dashed line).
2. The line ℓ divides the space $\mathbb{R}^2$ into two sets, the set $A = \{p \in \mathbb{R}^2 \mid p \text{ is above } \ell\}$ and its complement A' .
3. We iterate along the samples $\mathcal{S}$ backwards to find the first line segment $\{(x_{k-1}, y_{k-1}), (x_k, y_k)\}$ that satisfies one of following two conditions:

$$(x_{m-1}, y_{m-1}) \in A' \Rightarrow (x_k, y_k) \in A \wedge (x_{k-1}, y_{k-1}) \in A',$$

or

$$(x_{m-1}, y_{m-1}) \in A \Rightarrow (x_k, y_k) \in A' \wedge ((x_{k-1}, y_{k-1}) \in A \vee (x_{k-1}, y_{k-1}) \in \ell).$$

4. As the segment $\{(x_{k-1}, y_{k-1}), (x_k, y_k)\}$ crosses from A to A' , from A' to A or from A' to ℓ , then there must be an intersecting point $(x_q, y_q) \in \ell$, which we use to represent the final point in the cycle.
5. We create a cycle $o_1 = \{(x_q, y_q, \hat{t}_q)\} \cup \{(x_i, y_i, t_i) | i = k, k+1, \dots, m\}$, where $\hat{t}_q$ is interpolated.
6. We iteratively repeat the first five steps with the subset $\mathcal{S}' = \{(x_i, y_i, t_i) | i = 1, 2, \dots, k\}$ until no more cycles are identified. In this way, we obtain the set of cycles $\mathcal{O}$.

It is important to note that this algorithm works for boundaries changing slowly with respect to the robot's velocity. If the shape deforms too fast, it is possible that no cycle will be identified.

4.3.2 Estimating s

The key of our estimation method is attempting to track the behavior of each point in the curve. In order to achieve this aim, we parametrize each sample point on the curve. We use the set of cycles $\mathcal{O}$ to assign a curve parameter s for each sampled point. Then, for each cycle $o \in \mathcal{O}$, composed by n_o points, with an arc-length L_o , we estimate the curve parameter of the i th point as

$$\hat{s}_i = \frac{1}{L_o} \sum_{j=1}^{i-1} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2},$$

which is the length of the line string from point 1 to point i . Then, we can extend the cycle information by including the estimated $\hat{s}$, as $\hat{o} = \{(x_j, y_j, t_j, \hat{s}_i) | j = 1, 2, \dots, n_o\}$. The trajectory data $\mathcal{D}$ is formed by the union of all cycles information (removing the repeated elements when $s = 0$ and $s = 1$). Hence, the complete dataset $\mathcal{D}$ with k samples is given by $\mathcal{D} = \{(t_i, x_i, y_i, \hat{s}_i) | i = 1, \dots, k\}$.

4.4 Predicting the boundary behavior

In this section, we propose an approach to solve Problem 1. We need to estimate the boundary $\partial\Omega_t$ for any $t > t_0$, using the robot's data $\mathcal{D}$.

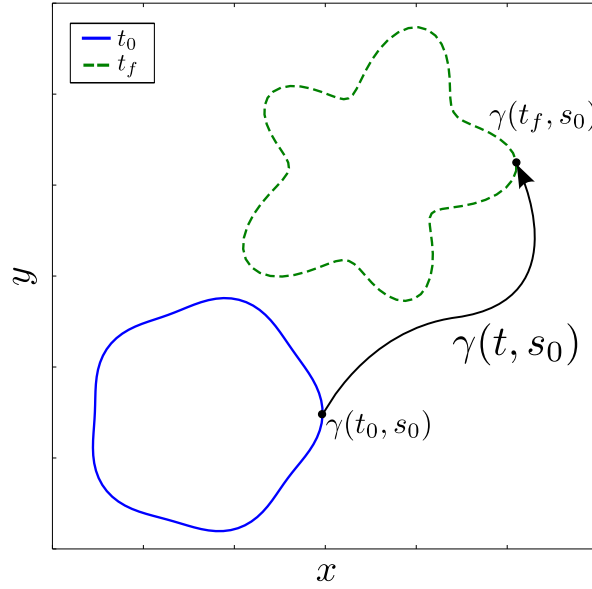


Figure 4.4: Illustration of a trajectory $\gamma(t, s_0)$, $t \in [t_0, t_f]$ of an arbitrary boundary point with parameter s_0 , that starts at point $\gamma(t_0, s_0)$, and finishes at point $\gamma(t_f, s_0)$. The continuous blue line represents the boundary curve at time t_0 . The dashed green line represents the final boundary curve at time t_f . The black arrow represents the trajectory of the point with parameter s_0 .

4.4.1 Estimating the trajectory of a boundary point

We start analyzing the trajectory of a single point in a dynamic boundary. Given a time interval $t = [t_0, t_f]$, we fix the curve parameter to an arbitrary value $s_0 \in [0, 1]$, the trajectory of this point can be represented by an open curve $\gamma(t, s_0)$, for $t \in [t_0, t_f]$ (Figure 4.4 illustrates a trajectory). The form of the trajectory depends on the dynamics of the boundary function. As we assume that it changes smoothly, we can approximate the open curve $\gamma(t, s_0)$ by a n -degree polynomial $\hat{\gamma}$:

$$\hat{\gamma}(t, s_0) = \begin{bmatrix} \beta_{0,0} & \beta_{0,1} & \dots & \beta_{0,n} \\ \beta_{1,0} & \beta_{1,1} & \dots & \beta_{1,n} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ \vdots \\ t^n \end{bmatrix}.$$

Letting $\beta(s_0)$ be the $2 \times (n+1)$ constant matrix for s_0 , and $\mathbf{f}(t) = [t^0, t^1, \dots, t^n]^T$ the exponents of the variable t , we may represent the trajectory of the boundary point as

$$\hat{\gamma}(t, s_0) = \beta(s_0) \mathbf{f}(t). \quad (4.4)$$

The general case for every s , not only s_0 , is described in the following subsection.

4.4.2 Estimating the boundary behavior

Now, we extend the trajectory of the point s_0 to all the points in the curve. Since the matrix β changes continuously with respect to the parameter s (Definition 1), we define a general function $\beta(s)$ that returns a matrix of $2 \times (n + 1)$, with values depending on the parameter s . Therefore, we can generalize (5.3.1) for every $s \in [0, 1]$ as

$$\hat{\gamma}(t, s) = \beta(s) \mathbf{f}(t). \quad (4.5)$$

Each value of the function $\beta(s)$ must satisfy the following properties.

Properties 1. *The function $\beta(s)$ satisfies the criteria below:*

1. *It forms a closed loop with $\beta(0) = \beta(1)$.*
2. *It is continuous and differentiable.*
3. *It is injective for every $s \in [0, 1)$.*

We approximate the function $\beta(s)$ with a periodic function $\hat{\beta}$ represented by a linear combination of sines and cosines. This approach satisfies the three mandatory properties for $\beta(s)$. We take the first $2m + 1$ terms of the Fourier series for each $\beta_{i,j}$,

$$\hat{\beta}_{ij}(s) = a_0^{(ij)} + \sum_{k=1}^m a_k^{(ij)} \sin(2\pi ks) + \sum_{k=1}^m b_k^{(ij)} \cos(2\pi ks). \quad (4.6)$$

Separating the coordinates of γ into two independent functions, we can rewrite the function γ as

$$\gamma(t, s) = \begin{bmatrix} x(t, s) \\ y(t, s) \end{bmatrix}.$$

We introduce the approximated function $\hat{\beta}$ in (5.11) to obtain

$$\hat{\gamma}(t, s) = \begin{bmatrix} \hat{x}(t, s) \\ \hat{y}(t, s) \end{bmatrix} = \begin{bmatrix} \hat{\beta}_x(s) \mathbf{f} \\ \hat{\beta}_y(s) \mathbf{f} \end{bmatrix}, \quad (4.7)$$

where $\beta_x(s)$ and $\beta_y(s)$ are the first and second rows of $\beta(s)$, respectively. Taking $\mathbf{g}(s) = [1, \sin(s), \dots, \sin(ms), \cos(s), \dots, \cos(ms)]^T$, and C_x and C_y as constant matrices with dimensions $(2m + 1) \times (n + 1)$, we have

$$\beta_x = \mathbf{g}^T C_x, \quad (4.8)$$

$$\beta_y = \mathbf{g}^T C_y. \quad (4.9)$$

Substituting (4.8) and (4.9) in (4.7),

$$\hat{\gamma}(t, s) = \begin{bmatrix} \mathbf{g}^T C_x \mathbf{f} \\ \mathbf{g}^T C_y \mathbf{f} \end{bmatrix}, \quad (4.10)$$

and multiplying the matrices, we have

$$\hat{\gamma}(t, s) = \begin{bmatrix} \sum_i^{n+1} \sum_j^{2m+1} c_{ij} f_i g_j \\ \sum_i^{n+1} \sum_j^{2m+1} d_{ij} f_i g_j \end{bmatrix},$$

which can be rewritten as

$$\hat{\gamma}(t, s) = \begin{bmatrix} c_1 & c_2 & \dots & c_{(2m+1)(n+1)} \\ d_1 & d_2 & \dots & d_{(2m+1)(n+1)} \end{bmatrix} \begin{bmatrix} f_1 g_1 \\ f_1 g_2 \\ \vdots \\ f_{n+1} g_{2m+1} \end{bmatrix},$$

where c_{ij} , and d_{ij} are the elements of the matrices C_x and C_y respectively. Finally, making $\mathbf{C}$ as the matrix of constants with dimension $(n+1)(2m+1) \times 2$ and $\mathbf{h}$ as the Kronecker product $\mathbf{h} = \mathbf{f} \otimes \mathbf{g} = [f_1 g_1, f_1 g_2, \dots, f_{n+1} g_{2m+1}]^T$, we finally obtain

$$\hat{\gamma}(t, s) = \mathbf{C}^T \mathbf{h}. \quad (4.11)$$

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 $\mathbf{C}$.

It is important to highlight the influence of the n and m parameters. The n parameter determines the polynomial degree to approximate the trajectory of a single point in the closed curve. It can be chosen based on the number of identified cycles or depending on the order of the phenomenon, *e.g.* in the plume experiment (Sec. 4.6.2), we used $n = 2$ as (4.16) is a second order differential equation. As with the m parameter, normally $m > 6$. It determines the maximum frequency along the closed curve. For example, the simplest case $m = 1$ approximates the boundary by a circle. A large number for m can generate over-fitting in the presence of noisy measurements.

4.4.3 Modeling as a linear system

We use the time t and the curve parameter s as input variables, and the spatially located variables (x, y) as the output. For these reasons, we use the trajectory information $\mathcal{D} = \{(t_i, x_i, y_i, s_i) | i = 0, 1, \dots, k\}$ to predict the anomaly behavior by attempting to estimate the parameter matrix $\mathbf{C}$ of (5.12).

We model the problem as a linear system with the form

$$\mathbf{X} \mathbf{C} = \mathbf{Y}, \quad (4.12)$$

where $\mathbf{Y}$ is a matrix with dimension $k \times 2$ that contains each output location (x_i, y_i) , $i = 1, \dots, k$ in the robot's trajectory $\mathcal{D}$; $\mathbf{C}$ is the matrix of weights (from (5.12)) that we want to estimate; and $\mathbf{X}$ is the design matrix created using the input (t_i, s_i) , $i = 1, \dots, k$ and the functions of H , defined as

$$\mathbf{X} = \begin{bmatrix} f_1(t_1, s_1) g_1(t_1, s_1) & \dots & f_{n+1}(t_1, s_1) g_{2m+1}(t_1, s_1) \\ f_1(t_2, s_2) g_1(t_2, s_2) & \dots & f_{n+1}(t_2, s_2) g_{2m+1}(t_2, s_2) \\ \vdots & \vdots & \vdots \\ f_1(t_k, s_k) g_1(t_k, s_k) & \dots & f_{n+1}(t_k, s_k) g_{2m+1}(t_k, s_k) \end{bmatrix}$$

Therefore, the independent regressors in matrix $\mathbf{A}$ are functions of t and s and the coordinates (x, y) are the dependent variables for the regression problem. The polynomial degree is associated with the number of cycles as $|\mathcal{O}| \geq n$ and the m terms in the finite Fourier series depends on the number of sample points k , because it must satisfy $k \geq (2m + 1)(n + 1)$.

4.4.4 Estimating by least-squares

The objective consists in adjusting the parameter matrix $\mathbf{C}$ (from (5.13)) of the model function $\hat{\gamma}$ to best fit the data set $\mathcal{D} = \{(t_i, x_i, y_i, s_i) | i = 0, 1, \dots, k\}$. Thus, we try to find the matrix $\mathbf{C}$ that minimizes the Euclidean norm

$$\|\mathbf{Y} - \mathbf{X} \mathbf{C}\|^2.$$

The *least squares method* [Borowiak, 2001] finds its optimum by assuming that the errors $\epsilon_i = \mathbf{Y}_i - \mathbf{X}_i \mathbf{C}_i$ are independent random variables for all $i = 1, 2, \dots, k$. Assuming that $\mathbf{X}$ is a full-rank matrix, we can estimate $\tilde{\mathbf{C}}$ by solving

$$\tilde{\mathbf{C}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}. \quad (4.13)$$

Therefore, $\gamma(s, t)$ is approximated by $\tilde{\gamma}(s, t) = \tilde{C}^T H$. We present some experiments of the implementation of this method in Section 5.4.

4.5 Convergence analyses

In this section, we show that our method produces accurate estimate $\tilde{\gamma}(s, t)$ of $\gamma(s, t)$ when the number k of sampling points is large enough and the parameters n and m are also large enough to describe the phenomenon. By the repeated use of the triangle inequality, we have

$$|\tilde{\gamma}(s, t) - \gamma(s, t)| \leq |\tilde{\gamma}(s, t) - \hat{\gamma}(s, t)| + |\hat{\gamma}(s, t) - \hat{\gamma}(s, t)| + |\hat{\gamma}(s, t) - \gamma(s, t)|. \quad (4.14)$$

The Weierstrass approximation theorem [Simmons, 1963] guarantees that the last term on the right hand side can be made arbitrarily small by taking the polynomial degree n to be large enough. The second term can also be made arbitrarily small. The reason is that, by the Fourier approximation theorem, there is an integer m such that, for $m > m_{ij}$, we have

$$|\hat{\beta}_{ij}(s) - \beta_{ij}(s)| < \frac{\epsilon}{2(n+1)t_f^n}.$$

Taking $m > \max m_{ij}$ we have the inequality valid for all i, j and for all s . Therefore,

$$|\hat{x}(t, s) - \hat{x}(t, s)| \leq \sum_{j=0}^n |\hat{\beta}_{0j} - \beta_{0j}| |t^j| \leq \epsilon/2.$$

A similar calculation for $y(t, s)$ provides an arbitrarily small upper bound for the second term in (4.14). That is, $|\hat{\gamma}(s, t) - \hat{\gamma}(s, t)| \leq \epsilon$. Finally, the first term in (4.14) can also be made arbitrarily small assuming the classic linear regression model. Indeed, usual least squares regression asymptotic theory gives us that $\tilde{\gamma}(s, t)$ is a consistent estimator of $\hat{\gamma}(s, t)$. That is, $|\tilde{\gamma}(s, t) - \hat{\gamma}(s, t)| < \epsilon$ if k is large enough. Combining these three bounding errors together, we find that $|\tilde{\gamma}(s, t) - \gamma(s, t)| \leq 3\epsilon$.

4.6 Experiments

In our experiments, we initially work with a boundary based on an analytic function and subsequently with a simulated boundary produced by a substance that follows the advection-diffusion model. In order to quantify the difference between the real

anomaly Ω_t and the predicted anomaly $\hat{\Omega}_t$, we use the Lebesgue measure of the symmetric difference between the two sets as a metric

$$\delta(\Omega_t, \hat{\Omega}_t) := \mu(\Omega_t \setminus \hat{\Omega}_t) + \mu(\hat{\Omega}_t \setminus \Omega_t), \quad (4.15)$$

where μ is the Lebesgue measure, which computes the area of a set in $\mathbb{R}^2$. The symmetric difference computes the error between the prediction and the real anomaly. It takes into account the union of the wrong covered area ($\Omega_t \setminus \hat{\Omega}_t$) and the uncovered area ($\hat{\Omega}_t \setminus \Omega_t$). This metric was previously used in the related works [Susca et al., 2008] and [Saldaña et al., 2015].

4.6.1 Simulating a boundary function

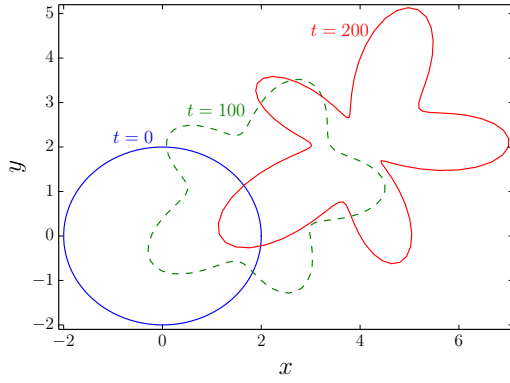
We initially simulate an anomaly which simultaneously translates, expands and shrinks its shape. The boundary satisfies the following equation,

$$\gamma_1 = \frac{1}{100} \begin{bmatrix} 4t + (t \sin(4\pi s) + 2t \cos(10\pi s) + 800) \cos(2\pi s) \\ 2t + (t \sin(4\pi s) + 2t \cos(10\pi s) + 800) \sin(2\pi s) \end{bmatrix},$$

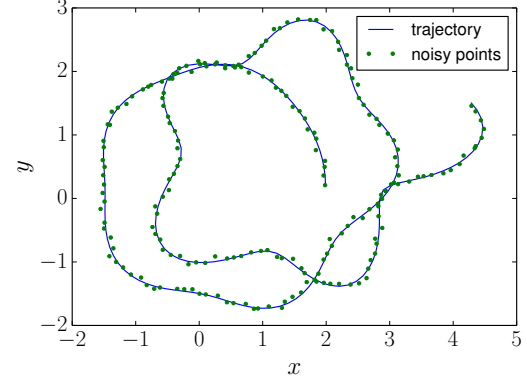
which is a variation of the boundary function in [Susca et al., 2008], but including a translation factor. In Figure 4.5a can be seen that the boundary γ_1 transforms its initial circular shape (at $t = 0$) to a star-shape (at $t = 200$). Additionally, we can see that the boundary is also translating with constant velocity. The robot moves with constant linear velocity $v = 0.79$ along the boundary (its trajectory is shown in Figure 4.5b, the trajectory is represented by the continuous line and the noisy measurements along the trajectory are represented by the dots.). This linear velocity is sufficient to allow the robot to circulate around the boundary and to complete two cycles.

Another way to present the estimated trajectory is shown in Figure 4.6, where the z-axis represents the time variable. We can see the boundary at different times and how the robot trajectory has only one point in the boundary for each time.

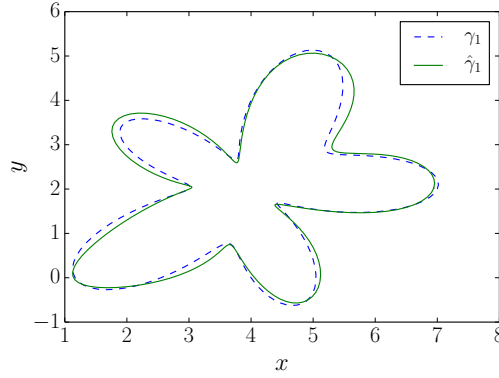
Our objective is to predict the boundary state at time $t = 200$ by using the sampled data $\mathcal{D}$ of the robot trajectory from time $t = 0$ to $t = 100$. We estimate the curve parameter $\hat{s}$ as described in Section 5.2. The estimation of the s parameter introduces an additional error that is minimized after applying the *least-squares*. The identified cycles are shown in Figure 4.7a and the resultant linear approximation for $\hat{s}$ in contrast to the ground truth s is shown in Figure 4.7b. The model that we approximate is based on (5.13). We use $n = 1$ for the polynomial and $m = 10$ for the Fourier series. Therefore, we estimated the 42 terms of the matrix $\mathbf{X}$ by using (5.14). Using the



(a) Evolution of the boundary γ_1 at three different times $t = 0, 100, 200$. It transforms its initial circular shape to a star-shape.



(b) Robot trajectory from time $t_0 = 0$ to $t_f = 100$, it is represented by the continuous blue line. The noisy measurements along the trajectory are represented by the green dots.



(c) Prediction for $t = 200$ by using noisy measurements. The real boundary is represented by the dashed line, and the prediction by the continuous line.

Figure 4.5: Dynamic boundary γ_1 , the robot trajectory and the resultant prediction.

exact points in the trajectory (represented by the continuous line in Figure 4.5b), the prediction error is $\delta = 4.27e - 12$ for all $t \in \mathbb{R}_{\geq 0}$, i.e. the approximation fits almost exactly to the real boundary.

In order to show that our method also works with noisy measurements, we introduced a Gaussian error for each measurement along the trajectory (small dots around the continuous line in Figure 4.5b). The resultant prediction is illustrated in Figure 4.5c, where the error is $\delta = 1.53$. In comparison to the area of the real boundary $\mu(\Omega_{200}) = 14.50$, it is a good approximation, since the last measurement was at time $t = 100$. We can observe that in this kind of analytical boundaries, the extrapolation for prediction fits very closely to the real one, even for long-term predictions with the presence of noisy measurements.

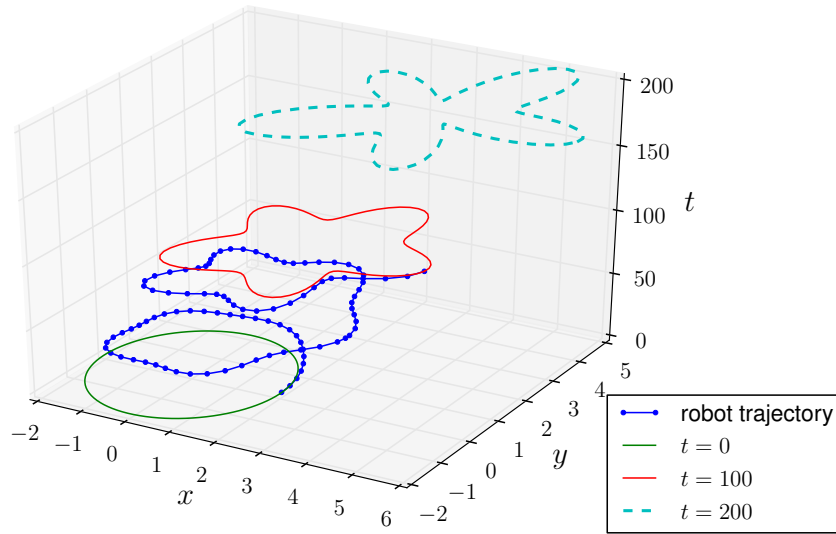
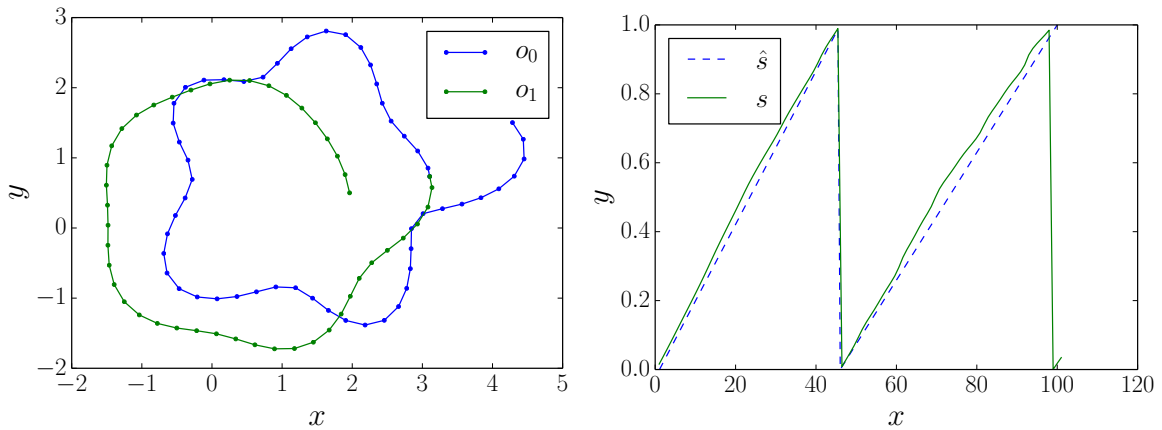


Figure 4.6: Dynamic boundary and the robot trajectory. The desired shape to be predicted is represented by the dashed line.



(a) Identified cycles in the robot trajectory. (b) Comparison between the estimated $\hat{s}$ and the real s .

Figure 4.7: Estimation of the curve parameter s .

In Susca et al. [2008], the boundary is approximated by a polygon, where each vertex is updated as the robot moves close to it. Good approximations can be achieved when many robots are distributed along the boundary. In this case, every vertex is updated quickly by different robots. However, when a single robot is used, each vertex is only updated after a lap and therefore, in the case of time-varying boundaries, the error in this method will not converge to zero.

Our technique is innovative in that, when the boundary shape is changing, our method is able to predict this motion for all the boundary points along the approx-

imated closed-curve. Our prediction method works with phenomena that changes smoothly in time. However, other behaviors that contain oscillations or random movements are difficult to estimate by the polynomial approximation.

4.6.2 Estimating a plume-front behavior

The propagation of a polluting substance in a non-static liquid is widely studied in physics and engineering. This physical phenomena is governed by two processes: diffusion and advection. The smooth behavior presented by the expansion of the concentration of a substance in the liquid is known as diffusion; and the motion of the substance “patch” due to the dynamics of the liquid is known as advection. The Navier-Stokes equations [Temam, 2001] describe the motion of viscous fluid substances and the equation for the advection-diffusion behavior is given by

$$\frac{\partial u}{\partial t} + v_x \frac{\partial u}{\partial x} + v_y \frac{\partial u}{\partial y} = k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad (4.16)$$

where the variable $u \in \mathbb{R}_{\geq 0}$ is the substance concentration; v_x and v_y are the advection velocity components in x and y respectively; and $k \in \mathbb{R}_{\geq 0}$ is a diffusion coefficient.

The estimation and tracking of plumes with multiple robots is studied in [Li et al., 2014; Fahad et al., 2015]. We apply our proposed model to show that a single robot is able to attain an accurate estimation and prediction of the behavior of the boundary of a plume.

In our experiment, we simulate a point-source pollution by implementing (4.16). The evolution of a substance is shown in Figure 4.8a. The trajectory of a robot moving with a linear velocity $v = 0.09$ around this substance patch is shown in Figure 4.8b. As a result, we can see in Figure 4.8c that our approach can estimate the whole boundary for an arbitrary time $t \in [0, 110]$ or predict for $t > 110$, even when the robot takes either only a sample or no samples for each specific time. The metric δ for all the estimations along the time t (based on the data of Figure 4.8b) is plotted in Figure 4.9. If the robot’s linear velocity increases from $v = 0.09$ to $v = 0.12$ or $v = 0.15$, the number of identified cycles increases from 3 to 4 and 5 respectively. Figure 4.9 also shows how the error δ is reduced because the same 110 samples are more evenly distributed in a rectangular (s, t) space. As our function interpolates the sample values with old and recent data, we can see that the estimated boundary in the time interval $t \in [20, 90]$ presents better accuracy. For the time $t = 110$, the error increases because there is data before but not after this moment. The method may predict the future boundaries for $t > 110$, but the error increases proportionally with time.

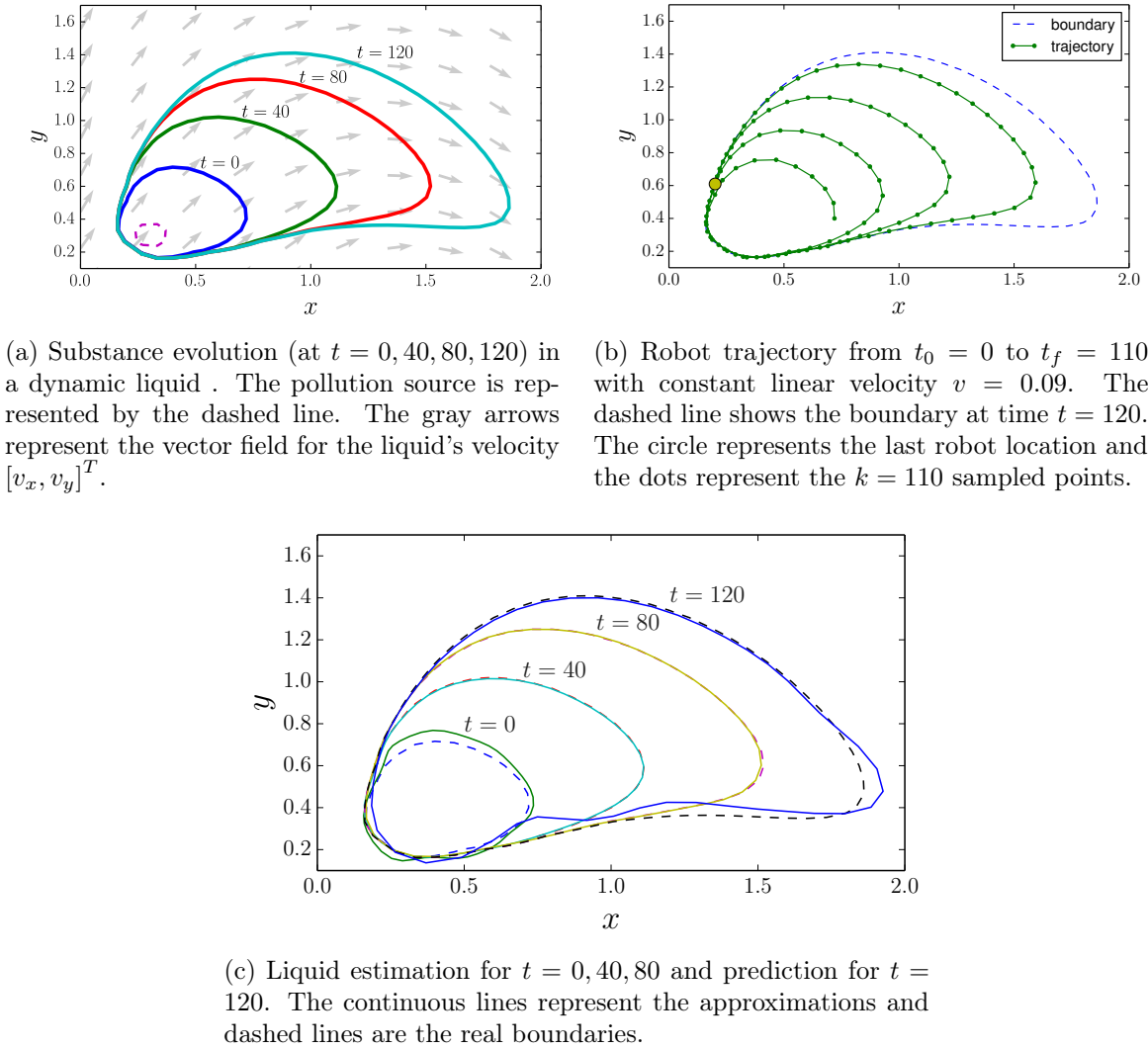


Figure 4.8: Viscous substance in a liquid and the robot trajectory around it.

In real experiments, we used a differential robot with a motion capture system for localization in order to track the plume-front as in simulations. In the accompanying video², it can be seen how the robot moves around the boundary in order to sample it. As the actual robot is nonholonomic, the maximum curvature to be tracked is restricted by its steering capability and the floor friction. We can see that the robot cannot always track the boundary exactly because of the steering limitation during the tracking process. The resultant sampled points along the trajectory are presented in Figure 4.10. The error when moving the robot along the boundary slightly affects the method, but the obtained prediction is very close to the real boundary as it can be seen in Figure 4.11.

²Video link: <https://www.youtube.com/watch?v=1bF9U3nbXJ8>

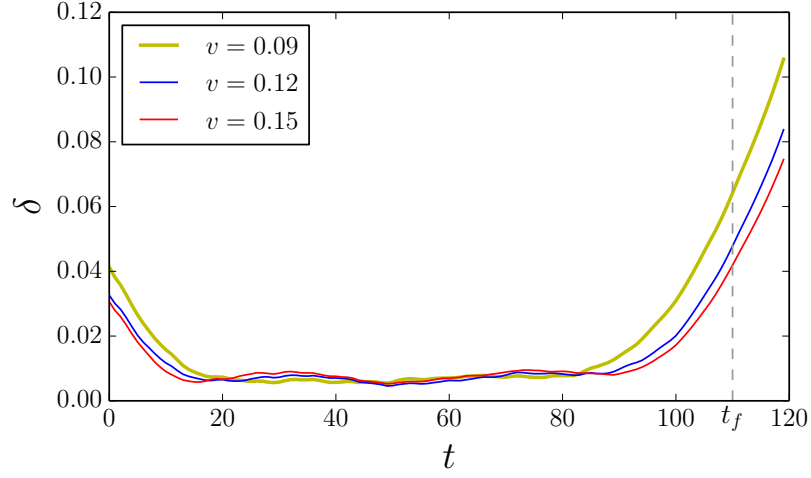
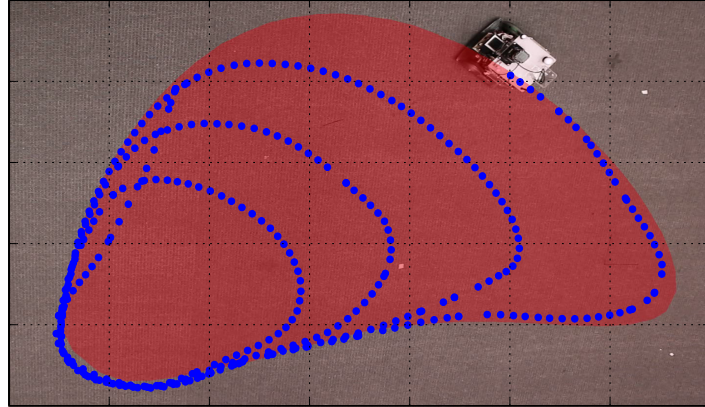
Figure 4.9: δ -error in time with different velocities.

Figure 4.10: Actual robot circulating around the dynamic boundary. The dots represent the sampled points along the robot trajectory.

As a result, we can use a single robot to estimate and predict boundaries with an error $\delta < 0.15m^2$, which is considerably small in comparison to the area of the bounded region $\mu(\Omega_{t_f}) = 3.28m^2$. The requirements to obtain a good estimation are: to complete the minimum number of cycles $|\mathcal{O}| \geq n$; to sample a number of points $k \geq (2m + 1)(n + 1)$; we also require that the robot move with high enough linear velocity v to obtain a good approximation $\hat{s}$ of the curve parameter s . As we argued in Section 4.5, the method works well even in the presence of noise in the location measurements, since the error $\delta \rightarrow 0$ when the number of sampled points increases.

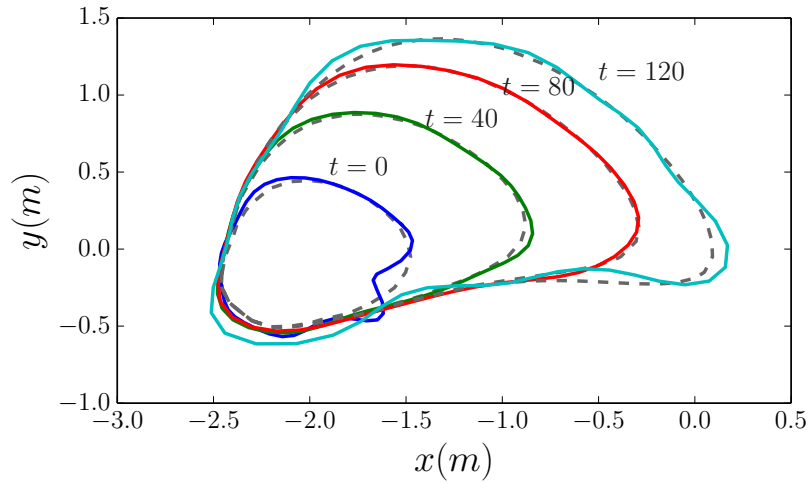


Figure 4.11: Liquid estimation by an actual robot. Continuous lines represent the estimations and dashed lines are the real boundaries.

4.7 Discussion

In this chapter, we presented a model that predicts the behavior of environmental boundaries by the parameters of an analytical function. We analyzed the convergence and demonstrated by simulations the efficiency of the proposed method. We showed how to estimate a boundary with high accuracy using a single robot without previous knowledge of the boundary dynamics.

The robot can predict the dynamics of the boundary by completing cycles around it. The estimation can be improved by increasing the sampling frequency and the velocity of the robot. When the robot moves fast in comparison to the motion of the boundary, it can identify similar cycles with more accuracy and, as a consequence, the estimation is also improved, as it has been shown in the simulations.

Chapter 5

Predicting Boundary Behaviors with Multiple Robots

In Chapter 3, we focused on the problem of predicting the boundary behavior using a single robot. In this chapter, we extend our method allowing multiple robots to predict the boundary behavior in a decentralized manner. Assuming that we have a good tracking method, we use the point-wise measurements to distribute the robots evenly along the curve, to map the boundary shape and to predict the boundary behavior. We highlight that our proposal do not require the dynamic model of the phenomenon; our method only needs the point-wise measurements to estimate the dynamics of the boundary.

5.1 Problem Statement

In this section, we describe the characteristics of the dynamic environmental boundary, and the multi-robot capabilities. Then, we state the problems that we consider in this chapter.

5.1.1 Dynamic boundary

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 3. A dynamic boundary is a set of planar points $\partial\Omega_t$ such that for all point $z \in \partial\Omega_t$, and for any arbitrarily specified $\xi > 0$, the open disc centered at point z with radius ξ contains points of Ω_t and its complement Ω_t^c .

The boundary $\partial\Omega_t$ can be modeled by an unknown boundary function such that

$$\partial\Omega_t = \{\gamma(t, s) \mid s \in [0, 1]\}.$$

Definition 4. A boundary function $\gamma : \mathbb{R}_{\geq t_0} \times [0, 1] \rightarrow \mathbb{R}^2$ describes a simple closed curve, mapped by the parameter $s \in [0, 1]$, such that $\gamma(t, 0) = \gamma(t, 1)$ and the restriction that $\gamma(t, s)$ is an injective function of $s \in [0, 1)$ for a fixed time t . Hence, γ describes a continuous curve with no self-intersection points.

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

$$\mathbf{T}(t, s) = \frac{\partial\gamma(t, s)/\partial s}{\|\partial\gamma(t, s)/\partial s\|}. \quad (5.1)$$

where $\|\cdot\|$ 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 γ 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

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

5.1.2 Multi-Robot Model

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 $\gamma(t, s)$, 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., $s_{i-1}(t) < s_i(t) < s_{i+1}(t)$. We assume that the robots are always on the curve, then the robot location in the Euclidean space $\mathbb{R}^2$ is given by $\mathbf{x}_i(t) = \boldsymbol{\gamma}(t, s_i(t))$, and its first order dynamics is

$$\dot{\mathbf{x}}_i(t) = \frac{d\boldsymbol{\gamma}(t, s_i(t))}{dt}. \quad (5.2)$$

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

$$\|\dot{\mathbf{x}}_i(t)\| = u_i(t). \quad (5.3)$$

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

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

5.1.3 Available Information

Each robot i is equipped with sensors to obtain local information in discrete time. At each sampling step t_k , $k \in \mathbb{N}$, it can observe a point in the curve as its own location $\mathbf{z}_i(t_k) = \mathbf{x}_i(t_k)$, the velocity of a point in the boundary $\dot{\mathbf{z}}_i(t_k) = \partial \boldsymbol{\gamma}(t, s_i(t_k)) / \partial t$, and the unitarian tangent vector $\mathbf{T}_i := \mathbf{T}(t, s_i(t))$ of the curve at the robot location. We illustrate the robots and the sampled vectors in Figure 5.1.

The robots use a ring communication topology to interchange messages among them. This communication network offers a natural way to maintain network connectivity and to reduce the communication links 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 $n + 1 = 1$.

5.1.4 Objective

Our objective is to use a team of robots to predict the behavior of a dynamic boundary. To achieve this, we have identified two main problems to be solved. Initially, we want to track the behavior of every point in the curve. To achieve this task, we have to solve the following problem.

Problem 2. *How to coordinate and control a team of n robots in order to frequently visit every point in a dynamic boundary $\partial\Omega_t$.*

Once tracking the time-varying boundary, we want to predict its behavior. Then, we have the second problem.

Problem 3. *How to use the distributed acquired information by the team of robots in order to predict the boundary behavior.*

We present a method to solve Problem 2 in Section 5.2. Then, we tackle Problem 3 by proposing a prediction method in Section 5.3.

5.2 Cooperative Parametrization of the Boundary

In order to track the motion of every point in 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 equals to n is a temporal

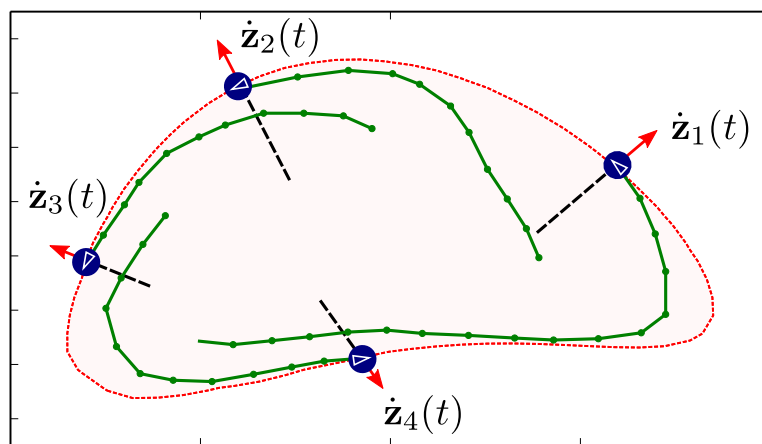


Figure 5.1: Enclosing a boundary with four robots. The continuous red curve and the dashed red curve represent the boundary at the initial time t_0 and a time t_k respectively. The blue disks represent the robots on the curve a time t_k . The arrows represent the velocity vector $\dot{\mathbf{z}}_i$. The gray regions represent the feasible region for the movement of initial points in the curve. The robot paths are represented by the dotted lines.

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.

5.2.1 Enclosing the boundary

In the beginning, the robots start at different arbitrary locations on the curve. They move in a counterclockwise manner along the curve with the control input

$$u_i = \Upsilon, \quad (5.4)$$

where $\Upsilon > 0$ is a desired constant speed. During this stage, the local task of each robot i is looking 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 $i + 1$. Figure 5.1 illustrates four robots moving along a boundary and projecting their vector $\mathbf{z}_i(t)$ to enclose a boundary. In relative slow-moving boundaries (Assumption 3), it is also possible to use the perpendicular vector to $\mathbf{T}_i$, if the vector $\dot{\mathbf{z}}_i(t_k)$ is unknown. In a previous work Saldaña et al. [2016], we presented this approximation with satisfactory results.

Following the ring communication topology, each robot i shares its sample $(t, \mathbf{z}_i(t), \dot{\mathbf{z}}_i(t))$ 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 about enclosing the boundary is completed when all the local tasks are completed. Checking the global task is straightforward; each robot shares a flag of its local variable

$$b_i = \begin{cases} \tau_i & \text{if } i = n \\ b_{j+1} \wedge \tau_i & \text{if } i < n, \end{cases}$$

where the boolean variable τ_i determines if the local task is completed or not by robot i . In this way, the temporal leader, robot n , identifies when the global task is completed by checking the received flag b_1 . In other words, this method is a decentralized logic comparator for the *and* operator where robot n checks if all the local tasks are completed.

5.2.2 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 is not computed yet, an initial way to enclose a boundary, is 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 $\mathcal{P}_i(t_k) = \{\mathbf{z}_i(t) | t = t_1, \dots, t_k\}$ with sorted points by descending time. Figure 5.2a illustrate three different robot paths after enclosing the boundary.

In this task, Robot n sends a set with its path $\{\mathcal{P}_n(t_k)\}$ to the robot $n - 1$. When robot $n - 1$ receives that 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 compute the set $\{\mathcal{P}'_n(t_k), \dots, \mathcal{P}'_1(t_k)\}$.

Definition 5. A piecewise boundary $\mathcal{B}(t_k) := \{\mathcal{P}'_n(t_k), \dots, \mathcal{P}'_1\}$ is a set of n polylines that surrounds a boundary.

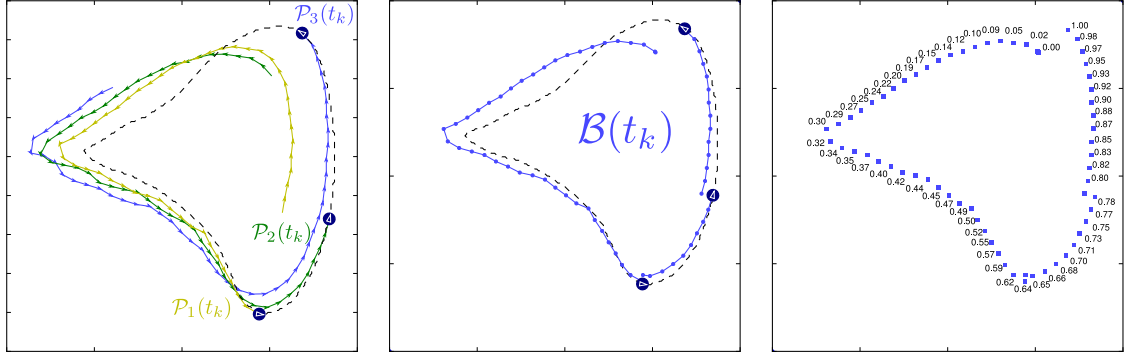
Each of these polylines comes from a robot path, and they can be connected by projecting the velocity vector $\dot{\mathbf{z}}$ of the first point of each polyline.

Figure 5.2b illustrates the resultant piecewise boundary after processing the paths that were presented in Figure 5.2a. 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 $\ell_i := \text{lenght}(\mathcal{P}'_i(t_k))$. The total length of the piecewise boundary is $\sum_{i=1}^n \ell_i$. The number of points in a 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

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

Figure 5.2c exemplifies the parametrization of the piecewise boundary for the aforementioned example.



(a) Initial paths after completing a cycle. The polylines are represented by the dotted line. The robot locations and actual boundary, at time t_k , are represented by the blue disks and the dashed line respectively.

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

(c) Parametrization of each sampled point by using the piecewise boundary.

Figure 5.2: 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). (iii) parametrizing the piecewise boundary based on the arc-length of each continuous line as illustrated in Panel (c).

5.2.3 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 the parametrization updated. At each time step t_k , given an piecewise boundary $\mathcal{B}(t_{k-1})$, robot i samples a new point $\mathbf{z}_i(t_k)$ and its velocity $\dot{\mathbf{z}}_i(t_k)$. It projects the vector $\dot{\mathbf{z}}_i(t_k)$ until it intersects the path of the robot $i + 1$. Assume that the intersected point 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 assign 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 in the curve.

5.2.4 Distributing the robots along the curve

In this stage, our objective is controlling the robots 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 (5.3)). 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 to update every point as frequently as possible is 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. [2010] designed a control law to move and distribute a robot team around a circular shape. Sabattini et al. [2010, 2012] 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 [Franchi et al., 2010].

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

$$u_i = \Upsilon + K(\ell_{i+1} - \ell_i), \quad (5.5)$$

where $K > 0$ is a gain constant. It means that we want all the length of the updated paths to be the same, $\ell_i \rightarrow \bar{\ell}_i$, where $\bar{\ell}_i = (\ell_{i+1} + \ell_{i-1})/2$. This behavior will homogeneously distribute the robots as long as they move on relative slow-changing boundaries (see Assumption 3).

Proposition 2. *If the boundary is relative slow-moving, satisfying Assumption 3, and the team of robots follows the control rule of (5.5), the robots asymptotically converge to a equidistant distribution along the curve (by arc-length). Additionally, the robots will keep moving with constant speed Υ .*

Proof. Since any simple closed curve is topologically equal to a circle, we can apply a similar approach as moving robots around a circle Franchi et al. [2010]. Initially, we re-parametrize the curve by arc-length. The arc-length parameter is given by

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

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

$$\alpha(t, r) = \gamma(t, s(r)),$$

where the parameter r is bounded by $0 \leq r \leq L(t)$, and $L(t)$ is the arc-length of the curve at time t . The location of the i th robot in the curve $\alpha(t, r)$ is denoted by $r_i(t)$.

We can decompose the robot dynamics from (5.2) by using the chain rule in the

new parametrization,

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

Then, its magnitude satisfies,

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

We know that the tangent vector is normalized, $\|\mathbf{T}_i\| = 1$. From Assumption 2 and $\partial \boldsymbol{\alpha} / \partial t = \partial \boldsymbol{\gamma} / \partial t$, we can obtain

$$\dot{r}_i(t) - \epsilon_v \leq \|\dot{\mathbf{x}}_i(t)\| \leq \dot{r}_i(t) + \epsilon_v.$$

Therefore, based on Assumption 3 and on (5.3), we can say that

$$\dot{r}_i(t) = u_i(t). \quad (5.6)$$

From Assumption 3, we can say that the updated path of the i th robot is approximately equal to a section of the curve. Then, $\ell_i \approx r_i - r_{i-1}$. The controller from (5.5) can be expressed as,

$$u_i = \Upsilon + K(r_{i+1} + r_{i-1} - 2r_i). \quad (5.7)$$

We rewrite (5.6) and (5.7) 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 equals to $[-2 \ 1 \ 0 \ \dots \ 0 \ 1]$, with $\mathbf{1} = [1, \dots, 1]^T$, and with $\mathbf{r} = [r_1, \dots, r_n]^T$ 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

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

The linear system yields the error $\mathbf{e}$ equals to the average of its initial values $\mathbf{e}(0)$, since the first eigenvalue of the Matrix $\mathbf{A}$ is zero but the others are negative. Moreover, this average is zero because $\mathbf{1}^T \mathbf{e}(0) = \mathbf{1}^T \mathbf{A} \mathbf{r}(0)/2 = 0$. $\square$

Proposition 3. *If the boundary is slow-moving and the team of robots follow the control rule (5.5), every point in the curve will be tracked with frequency*

$$F(t) = \frac{n\Upsilon}{L(t)}. \quad (5.8)$$

Proof. Lets say that robot i completed a cycle from time $t - 1/F$ to t and its path within this time interval is $P(t - 1/F, t)$. The error $|P(t - 1/F, t) - L(t)|$ tends to zero, when Υ increases. Based on Assumption 3, we can say that $L(t) \approx P(t - 1/F, t)$. Therefore, robot i completes a cycle with frequency

$$F_i(t) = \frac{\Upsilon}{L(t)}.$$

By Proposition 2, we know that controller (5.5) drives the robots to an equidistant distribution along the curve. Then, every point in the curve will be visited with frequency:

$$F(t) = nF_i(t) = \frac{n\Upsilon}{L(t)}.$$

$\square$

Based on (5.8), we can state that using our proposed controller, the tracking frequency $F(t)$ increases when the desired robot speed Υ or the number of robots n increases.

We can measure the total variation of the curve by

$$e(t) = \int_0^1 \left\| \frac{\partial \gamma(t, s)}{\partial t} \right\| ds. \quad (5.9)$$

Then, we can relate the tracking frequency and the total variation by the expression

$$\int_{t_0}^{t_0 + \frac{1}{F}} e(t) dt = \int_{t_0}^{t_0 + \frac{1}{F}} \int_0^1 \left\| \frac{\partial \gamma(t, s)}{\partial t} \right\| ds dt.$$

In the experiments section, we will illustrate the impact of the tracking frequency and the variation of the curve on the prediction.

5.3 Cooperative prediction

In this section, we extend that model for distributed on-line prediction with multiple robots.

5.3.1 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,

$$\hat{\gamma}(t, s_0) = \begin{bmatrix} \beta_{00} & \beta_{01} & \dots & \beta_{0n} \\ \beta_{10} & \beta_{11} & \dots & \beta_{1n} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ \vdots \\ t^n \end{bmatrix}. \quad (5.10)$$

Letting $\beta(s_0)$ be the $2 \times (n + 1)$ constant matrix for s_0 , and $\mathbf{f}(t) = [t^0, t^1, \dots, t^n]^T$ the exponents of the variable t , we represent the trajectory of the boundary point as $\hat{\gamma}(t, s_0) = \beta(s_0) \mathbf{f}$. Since the curvature changes smoothly, we can generalize this for every point in the curve $s \in [0, 1]$, by generalizing the matrix β as,

$$\hat{\gamma}(t, s) = \beta(s) \mathbf{f}(t). \quad (5.11)$$

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

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

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

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

With some algebraic manipulation (details in Chapter 4), we can separate the curve function (5.11) 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

$$\hat{\gamma}(t, s) = \mathbf{C}^T \mathbf{h}(t, s). \quad (5.12)$$

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 $\mathbf{C}$.

5.3.2 Estimating the curve function

We use the time t and the curve parameter s as input variables, and the sampled locations $\mathbf{z}(t) = (x, y)$ as the output. For these reasons, we use the trajectory information $\mathcal{D} = \{(t_j, x_j, y_j, s_j) | j = 0, 1, \dots, k\}$ to predict the anomaly behavior by attempting to estimate the parameter matrix $\mathbf{C}$ of (5.12).

We model the problem as a linear system with the form

$$\mathbf{X} \mathbf{C} = \mathbf{Y}, \quad (5.13)$$

where $\mathbf{Y}$ is a matrix with dimension $k \times 2$ that contains each output location (x_i, y_i) , $i = 1, \dots, k$ 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) , $i = 1, \dots, k$ 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 (5.13), of the model function $\hat{\gamma}$ to best fit the data set $\mathcal{D}$. Assuming that $\mathbf{X}$ is a full-rank matrix, we can estimate $\tilde{\mathbf{C}}$ by solving

$$\tilde{\mathbf{C}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}. \quad (5.14)$$

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

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

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.

5.3.3 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 expensive computation $(\mathbf{X}_{k+1}^T \mathbf{X}_{k+1})^{-1}$, which has a cubic time complexity with respect to the number of points. In order to make it faster, we extend our method using *Recursive Least Squares* (RLS) 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 contrast, the complexity of RLS is quadratic for each iteration. Additionally, it is memory efficient, since it does not require to store the whole dataset.

In order to make (5.14) iterative, we rewrite it as the operation with the rows of $\tilde{\mathbf{C}}$,

$$\tilde{\mathbf{C}} = \left(\sum_{i=1}^k \mathbf{h}_i \mathbf{h}_i^T \right)^{-1} \sum_{i=1}^k \mathbf{h}_i \mathbf{y}_i^T, \quad (5.15)$$

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 (5.15) defines the correlation matrix $\mathbf{P}$, which can be calculated in a recursive manner,

$$\mathbf{P}(i) = \mathbf{P}(i-1) + \mathbf{h}_i \mathbf{h}_i^T, \quad (5.16)$$

with $\mathbf{P}(1) = \mathbf{h}_1 \mathbf{h}_1^T$, and the *cross-correlation vector* of right sum in (5.15),

$$\mathbf{q}(i) = \mathbf{q}(i-1) + \mathbf{h}_i \mathbf{y}_i^T. \quad (5.17)$$

with $\mathbf{q}(1) = \mathbf{h}_1 \mathbf{y}_1$. Then, (5.14) is rewritten as the multiplication of two iterative terms,

$$\tilde{\mathbf{C}} = \mathbf{P}(k)^{-1} \mathbf{q}(k). \quad (5.18)$$

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,

$$\begin{aligned}\mathbf{P}(k)^{-1} &= (\mathbf{P}(k-1) + \mathbf{h}_k\mathbf{h}_k^T)^{-1} \\ &= \mathbf{P}(k-1)^{-1} - \frac{(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)^T}{1 + \mathbf{h}_k^T\mathbf{P}(k-1)^{-1}\mathbf{h}_k}.\end{aligned}\quad (5.19)$$

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

5.3.4 Method 1: Forgetting old measurements

When the trajectory of each point in the boundary is oscillatory or difficult to approximate by a polynomial, our method can still offer a good approximation for short time horizons. We introduce a *forgetting factor* in order to prioritize the new data over the old sampled points Haykin [2008].

The *forgetting factor* $\lambda \in (0, 1]$ or *exponential weighting factor* determines the rate at which we downweight the the old values as new points are sampled. Taking $\lambda = 1$ uses the whole data-set as in the classical LS method. With $\lambda < 1$, we introduce the forgetting factor in (5.16) and (5.17),

$$\mathbf{P}(i) = \lambda \mathbf{P}(i-1) + \mathbf{h}_i \mathbf{h}_i^T, \quad (5.20)$$

$$\mathbf{q}(i) = \lambda \mathbf{q}(i-1) + \mathbf{h}_i \mathbf{y}_i. \quad (5.21)$$

Then the inverse of the correlation matrix of (5.19) becomes

$$\mathbf{P}(k)^{-1} = \lambda^{-1}\mathbf{P}(k-1)^{-1} - \lambda^{-1}\mathbf{g}(k)(\mathbf{P}(k-1)^{-1}\mathbf{h}_k)^T,$$

where

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

is the *gain vector*. We summarize the iterative process to update the boundary estimation when we add a new sample point, (t_k, s_k, x_k, y_k) in Algorithm 1.

By definition, we know that the robots are configured in a ring topology. The coordination among the robots is an iterative process where the weighted matrix is obtained by a cyclic messaging. Given an initial estimation by robot n , using (5.18),

Algorithm 1: BoundaryEstimation($((t_k, s_k, x_k, y_k), \mathbf{P}(k-1)^{-1}, \mathbf{q}(k-1))$)

```

1  $\mathbf{h}_k = \mathbf{h}(t_k, s_k)$ 
2  $\mathbf{y}_k = [x_k, y_k]^T$ 

3  $\mathbf{g}(k) = \frac{\mathbf{P}(k-1)^{-1}\mathbf{h}_k}{\lambda + \mathbf{h}_k^T \mathbf{P}(k-1)^{-1} \mathbf{h}_k}$ 
4  $\mathbf{q}(k) = \lambda \mathbf{q}(k-1) + \mathbf{h}_k \mathbf{y}_k^T$ 
5  $\mathbf{P}(k)^{-1} = \lambda^{-1} \mathbf{P}(k-1)^{-1} - \lambda^{-1} \mathbf{g}(k)(\mathbf{P}(k-1)^{-1} \mathbf{h}_k)^T$ 

6  $\tilde{\mathbf{C}} = \mathbf{P}(k)^{-1} \mathbf{q}(k)$ 
7  $\hat{\gamma}_k(s, t) = \tilde{\mathbf{C}}^T \mathbf{h}(t, s)$ .
8 return  $(\hat{\gamma}_k(s, t), \mathbf{P}(k)^{-1}, \mathbf{q}(k))$ 

```

based on $k-1$ data points, this robot sends the matrix $\mathbf{P}(k-1)$ and the vector $\mathbf{g}(k-1)$ to the next robot $n-1$. Using Algorithm 1, robot i uses the received information (the matrix $\mathbf{P}(k-1)$ and the vector $\mathbf{g}(k-1)$) from robot $i+1$ to update the estimation of the curve using its local sampled points. Using this method, each robot is able to compute and update its own boundary estimation by only using the matrix $\mathbf{P}$ and the vector $\mathbf{g}$. In this way, the robots get independent information about the curve and join it cooperatively by only using local messages. Then, the estimation circulates along the ring communication network without the need of a central unit.

It is important to highlight that the size of the shared packages is fixed and independent of: the number of robots; the number of local measurements; the dynamics of the boundary; and the shape of the boundary.

5.3.5 Method 2: Sliding window

Our second method to integrate the distributed information is the *sliding window*. Our online prediction is based on the last k sampled points (window), and the robots remove (slide) the older points. Then, each robot keeps in memory its own dataset

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

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 (5.17) and (5.19) respectively. Robot n sends the matrix and the vector to robot $n-1$. Then, robot $i < n$ receives the matrix $\mathbf{P}_{i+1}^{-1}(t_k)$ and the vector $\mathbf{q}_{i+1}(t_k)$ from robot $i+1$ and process to update it with its own k values 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

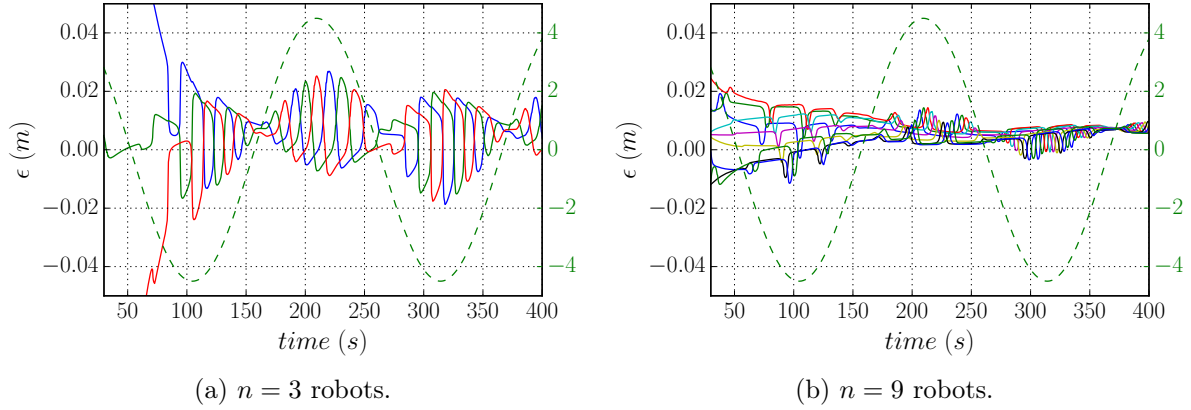


Figure 5.3: Consensus error for $n = 3$ and $n = 9$ number of robots respectively. Each colored line represents a different robot. The dashed line represents the total variation of the boundary (from (5.9)).

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

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

Finally, we get the curve function

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

which integrates the information that is distributed among the n robots. We want to emphasize that the estimation is obtained by exchanging n messages and the size of the message package is fixed, since the 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 .

5.4 Experiments

We want to validate our method through simulations and 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

$$\delta(\Omega_t, \hat{\Omega}_t) = \frac{\mu(\Omega_t \setminus \hat{\Omega}_t) + \mu(\hat{\Omega}_t \setminus \Omega_t)}{\mu(\Omega_t)}, \quad (5.22)$$

takes into account the non-estimated area $\mu(\Omega_t \setminus \hat{\Omega}_t)$ plus the misestimated area $\mu(\hat{\Omega}_t \setminus \Omega_t)$. In order to have a proportion of the error with respect to the actual area, we include the actual boundary area $\mu(\Omega_t)$ into the denominator. Similar metrics to quantify the accuracy in boundary estimation were presented in [Susca et al., 2008; Saldaña et al., 2015, 2016]. In the previous chapter, we show that our method can use a fast single robot to get accurate estimations for analytic boundaries, such as the one presented in [Susca et al., 2008]. In our experiments, we try to estimate a non-linear changing analytic boundary and the boundary of real liquid on a flow using multiple robots. Method 1 can be degenerated when the boundary presents oscillations or non-uniform variations because it takes into account the whole dataset. In contrast, Method 2 offers better estimations for the recent data and it is not affected by the oscillations. For this reason, in the following subsections, we present the results of using the Method 2.

5.4.1 Simulations

In our simulations we want to 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

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

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.

5.4.1.1 Enclosing

Initially, we studied how to enclose the boundary and maintain the robots in equidistant locations. In this simulation, we relaxed the Assumption 3, and the boundary achieves a high velocity in comparison to the robot velocity. However, we can see in Figure 5.3a that the error is still bounded when the boundary achieves its maximum velocity. The dashed line shows the oscillatory behavior of the total variation (from (5.9)). We can see that this oscillatory behavior is coupled with the error associated with robot

estimation. However, the size of the error is reduced as we increase the number of robots (see Figure 5.3b). 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.

5.4.1.2 Predicting

As it was aforementioned, our prediction method is improved if the robot are equidistantly distributed along the boundary. In Figure 5.4, we can see that the prediction method is also affected by the oscillatory behavior of the boundary. However, the predictions are very accurate, even in the worst case for $n = 3$ robots where the error is below 6.0%. The error is reduced rapidly by increasing the number of robots. For this specific boundary, seven robots are enough to keep the prediction error smaller than 1.0%. At this point, we found that increasing the number of robots from $n = 3$ to $n = 7$ significantly increases the estimation accuracy, but the increment is slow for $n > 7$.

We also want to analyze the prediction error for a fixed dataset. In other words, given a set of sampled points, we want to estimate the behavior of the boundary. The prediction error can be seen in Figure 5.5 for a fixed dataset. In this figure, we present predictions with a fixed number of Fourier terms, $M = 10$, and different polynomial degrees $N = 0, 1, 2$. A polynomial degree $N = 0$ means no variation in time, then, estimator tries to fit a fixed boundary that fits with all sampled points. We can see that this error is small at some point, where the least squares find the minimum, but outside that point is the worse estimator. A linear estimator offers a better approximation and also the error changes slowly. This is a good estimator since requires less sampled points and incorporates the time. The last one is $N = 2$, this approximation offers the best estimation, which keeps the error in a very slow percentage during the sampled time. However the error grows faster than the linear estimator for future predictions. This is one of the main problems of the estimator with high-degree polynomials. We also simulated high-degree polynomials but they require a larger number of points in the dataset.

5.4.2 Experiments with actual robots

In this section, we show that our method can be used on actual robots and real boundaries. We use a team of robotic boats in a tank in order to track and predict the shape of a liquid. In our testbed, the environment is composed by a tank with still water and its size is $3 \times 3 \text{ m}^2$ and we use a team of three robotic boats as it is pictured in

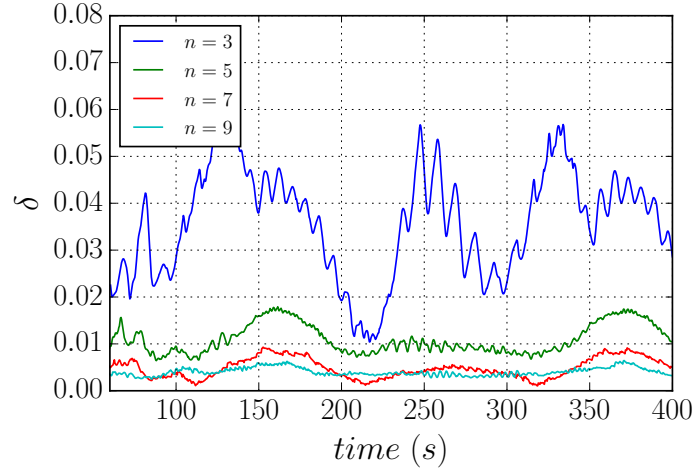


Figure 5.4: Prediction error for different number of robots n . Each colored line denotes the prediction error δ in time.

Figure 5.6. The location of the robots is obtained by using an external motion capture system around the tank. Each robot is driven by two propellers in the back as pictured in the Figure 5.7. We can control these robots by using the differential drive model. Although we assumed omnidirectional motion in the problem statement, we can relax this assumption by using these differential robots which are able to track the boundary and to control their speed. Although, we control the robots based on the differential drive model, in contrast to the ground robots, friction on water is small and the inertia robots can keep the robots moving, even when we turn the motors off. In this way, the

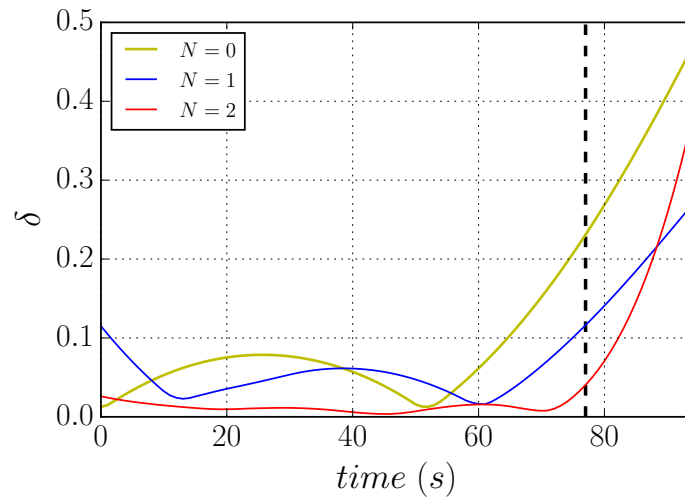


Figure 5.5: Prediction error for a fixed dataset. The vertical dashed line represents the last time that a point was sampled. The continuous lines represent the prediction error for three polynomial degrees, $N = 0, 1, 2$.

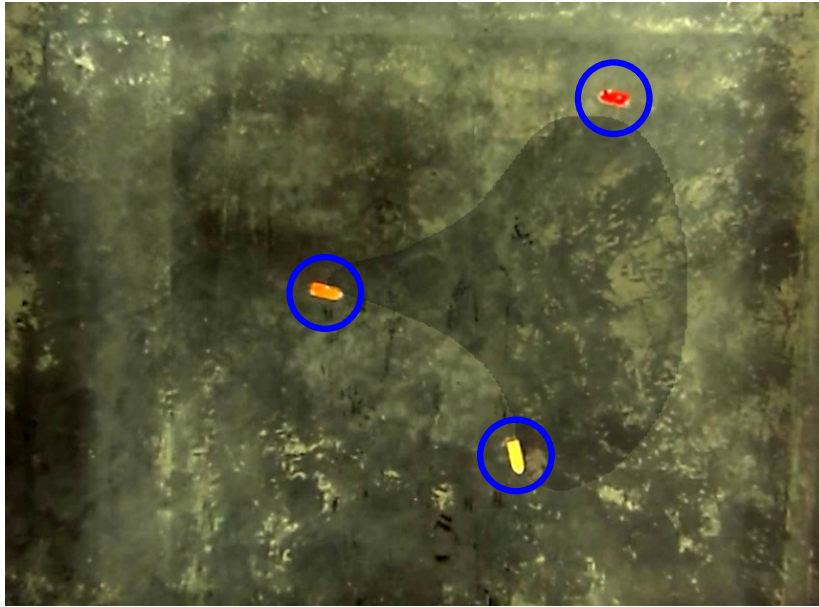


Figure 5.6: Top view of the robotic boats in the tank. The robots are highlighted by the blue circles and the tracked region is represented by the darker area.

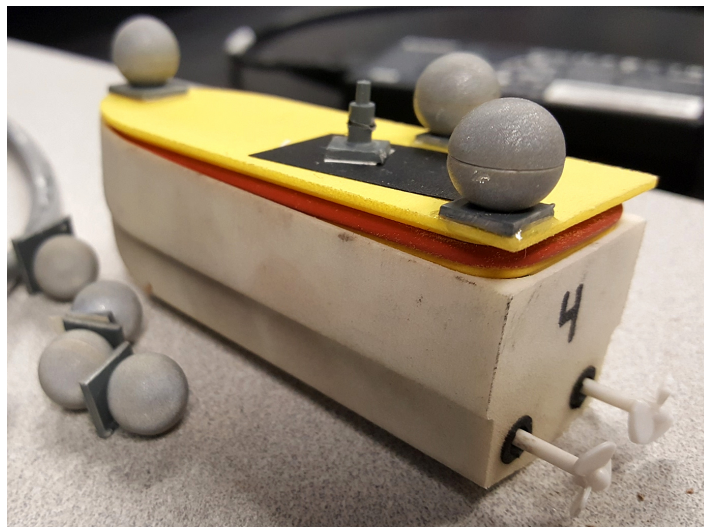


Figure 5.7: A robotic boat. It is driven by two propellers in the back. The reflective markers on the top are used for localization.

real behavior is different over the expectation because we did not include inertia nor friction in our model. There are other conditions that make the robots behave different as expected such as: wave disturbances generated by other robots; or water within the boat that changes the center of mass. However, the feedback controller maintains a good enough robot motion as we will describe as follows.

The boundary was previously recorded. It delimits the area of a viscous liquid on a dynamic fluid. In Figure 5.8, we can see three different snapshots of the experiment. We

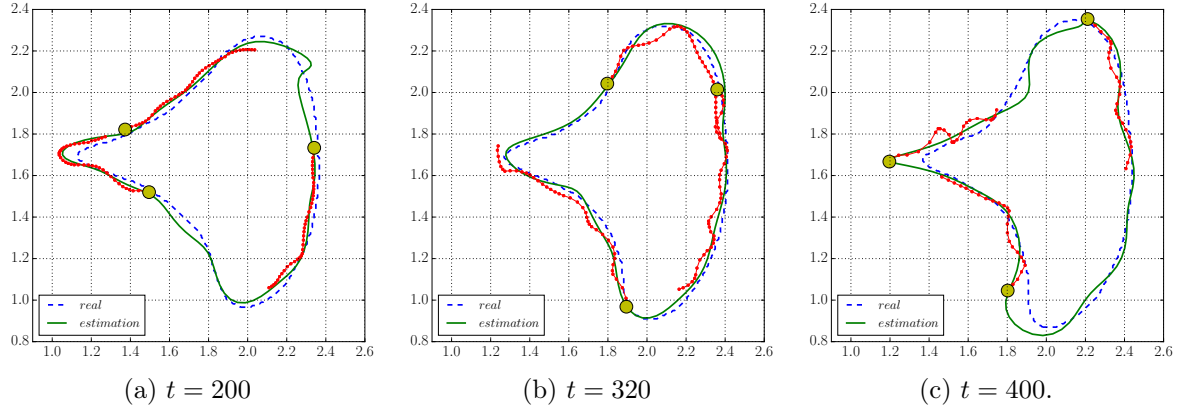


Figure 5.8: Snapshots of the experiment. The robot locations and their trajectories are represented by the yellow disks and the dotted line respectively. The real boundary is represented by the dashed line and the estimation by the continuous green line.

can see that all boundary points change their locations at different rates and directions. In the tracking process, the boats have difficulties to follow the boundary because the inertia makes the robots slide on the water and it is difficult to compensate with its small actuators. When the robots get outside the boundary, they present some oscillations before returning to the proper tracking. We would like to increase the speed of the robots in order to improve the prediction accuracy, but it reduces the quality of the tracking task by increasing the oscillations. Even though the boats were not able to exactly move along the boundary during the tracking process, we obtained estimations that matches closely to the real dynamic boundary. In Figure 5.8a and 5.8c, we can see on the left side that the robots had difficulties to move through the sections with high curvature (around the point $(1.2, 1.7)$). 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 in Figure 5.8b that the estimation is approximately equal to the real boundary even after inaccurate samples.

The error of the estimation during the experiment is presented in Figure 5.9. We can see that it presents some fluctuations due to the oscillatory movements, but in general the error is under 13%.

5.5 Discussion

In this chapter, 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 mea-

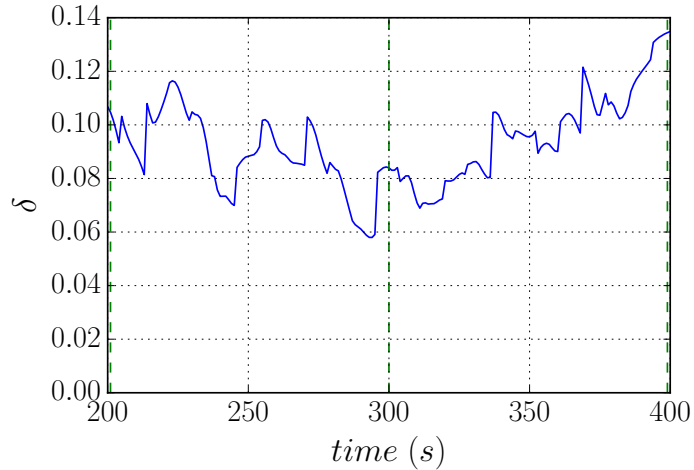


Figure 5.9: Prediction error using three robots. The vertical lines represent the time where the snapshots were taken.

surements 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.

Our method coordinates a team of n robots to sample and predict boundary behaviors. The distributed sampled information is integrated by using recursive least squares and a ring communication topology. 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. In general, the required number of robots to sample the boundary directly depends on the relationship between the speed of the boundary and speed of the robots. We also found in the experiments that there is a number of robots n where adding more robots does not the significantly increment the estimation accuracy. Unfortunately, we do not know the necessary number of robots to track a boundary a-priori because we do not have previous information about the boundary dynamics. However, we might be able to progressively add robots and check how the estimation accuracy is improved.

We assumed that the robots can maintain communication in a ring topology. However, there are cases such as u-shaped or flat elongated boundaries, where it is difficult to maintain the ring topology using wireless communication networks. For this reason, we would like to study alternative coordination methods to integrate the distributed information in sparse topologies.

As we showed in the experiments, our method is a promising candidate for real

environments because it offers accurate estimations even in the presence of noisy measurements or motion errors.

Chapter 6

Conclusion Remarks

During the first stage of this doctorate program, we have been developing two related methods for the scientific problem that we chose to tackle. The problem is about how to monitor and predict environmental phenomena that can be circumscribed by boundaries.

The first method addresses the subproblem of coordinating and controlling multiple robots for the detection of multiple dynamic anomalies in the environment. The main contributions of our approach are: *(i)* an algorithm for the coordination of a team of robots that are tasked to find and track multiple anomalies. This is an important feature for real world applications, since in some contexts, such as fire in a forest or algae bloom in a lake, more than one anomaly may appear or disappear in time. In previous works, we have found that the problem of searching for multiple anomalies is approached in an uncoordinated way, where an anomaly is detected but not tracked. Our approach allows for autonomous on-line allocation and deallocation of robots during the tracking process, depending on the size of the anomalies, which may dynamically vary. In our approach, a growing anomaly demands closer attention and demands more robots to be allocated throughout the mission execution. Meanwhile, if an anomaly is shrinking, tracking robots may be reallocated to explore other areas with higher priority; *(ii)* Efficient exploration. The exploration under uncertainty maintains the history of the sensed areas to reduce redundancy and to allow for full-map coverage.

The second method is focused on the estimation and prediction of boundary behaviors with a single robot. Our contributions are two fold. *(i)* Most of the related approaches relies on linear interpolation of the robot's positions in order to approximate the boundary by a polygon. Without previous knowledge of the boundary dynamics, we approached the problem in a different manner. We modeled the evolving boundary by a position-specific parametric analytical continuous function, where the parameters

can be estimated using the sampled punctual locations. *(ii)* Our proposal not only estimates the current closed-curve state, but it can also predict the future boundary behavior without previous knowledge of the phenomena.

Both methods presented good performance results, but one independent of the other and vice-versa. For instance, the prediction method is currently focused on a single robot and a coordinated multi-robot version is not developed yet. Finally, we propose a method to coordinate a team of n robots to cooperatively predict environmental boundaries. 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 and surfaces like lava on a volcano. Additionally, we would like to make our method resilient to wrong information in the network. For example, a single robot with a broken GPS can damage the prediction of the whole team.

Bibliography

- Barat, C. and Rendas, M.-J. (2003). Benthic boundary tracking using a profiler sonar. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 830--835. IEEE.
- Bertozzi, A. L., Kemp, M., and Marthaler, D. (2005). *Determining Environmental Boundaries: Asynchronous Communication and Physical Scales*, pages 25--42. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Bertuccelli, L. F. and How, J. P. (2006). Search for dynamic targets with uncertain probability maps. In *2006 American Control Conference*, pages 6 pp.--. ISSN 0743-1619.
- Borowiak, D. (2001). Linear models, least squares and alternatives. *Technometrics*, 43(1):99--99.
- Bruemmer, D. J., Dudenhoeffer, D. D., McKay, M. D., and Anderson, M. O. (2002). A robotic swarm for spill finding and perimeter formation. Technical report, DTIC Document.
- Casbeer, D., Beard, R., McLain, T., Li, S., and Mehra, R. (2005). Forest fire monitoring with multiple small uavs. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3530--3535. Ieee.
- Casbeer, D., Kingston, D., Beard, R., and McLain, T. (2006). Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351--360.
- Chung, C. F. and Furukawa, T. (2009). Coordinated pursuer control using particle filters for autonomous search-and-capture. *Robotics and Autonomous Systems*, 57(6-7):700 -- 711. ISSN 0921-8890.

- Clark, J. and Fierro, R. (2005). Cooperative hybrid control of robotic sensors for perimeter detection and tracking. *Proceedings of the 2005, American Control Conference, 2005.*, pages 3500–3505.
- Clark, J. and Fierro, R. (2007). Mobile robotic sensors for perimeter detection and tracking. *ISA transactions*, 46(1):3–13. ISSN 0019-0578.
- 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., and Fierro, R. (2007). Decentralized cooperative control - a multivehicle platform for research in networked embedded systems. *Control Systems, IEEE*, 27(3):58–78. ISSN 1066-033X.
- Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.
- Duttagupta, S., Ramamritham, K., and Kulkarni, P. (2011). Tracking dynamic boundaries using sensor network. *Parallel and Distributed Systems, IEEE Transactions on*, 22(10):1766–1774.
- Euler, J., Horn, A., Haumann, D., Adamy, J., and Stryk, O. (2012). Cooperative n-boundary tracking in large scale environments. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 1–6. IEEE.
- Fahad, M., Saul, N., Guo, Y., and Bingham, B. (2015). Robotic simulation of dynamic plume tracking by unmanned surface vessels. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2654–2659.
- Fiorelli, E. and Leonard, N. E. (2007). Exploring scalar fields using multiple sensor platforms: Tracking level curves. *2007 46th IEEE Conference on Decision and Control*, pages 3579–3584.
- Franchi, A., Stegagno, P., Di Rocco, M., and Oriolo, G. (2010). Distributed target localization and encirclement with a multi-robot system. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, pages 151–156.
- Grocholsky, B., Keller, J., Kumar, V., and Pappas, G. (2006). Cooperative air and ground surveillance. *Robotics Automation Magazine, IEEE*, 13(3):16–25. ISSN 1070-9932.

- Haykin, S. S. (2008). *Adaptive filter theory*. Pearson Education India.
- Hol, J. D., Schon, T. B., and Gustafsson, F. (2006). On resampling algorithms for particle filters. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 79--82. IEEE.
- Hsieh, C. H., Jin, Z., Marthaler, D., Nguyen, B. Q., Tung, D. J., Bertozzi, A. L., and Murray, R. M. (2005). Experimental validation of an algorithm for cooperative boundary tracking. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1078--1083 vol. 2.
- Jin, Z. and Bertozzi, A. L. (2007a). Environmental boundary tracking and estimation using multiple autonomous vehicles. In *Decision and Control, 2007 46th IEEE Conference on*, pages 4918--4923. IEEE.
- Jin, Z. and Bertozzi, A. L. (2007b). Environmental boundary tracking and estimation using multiple autonomous vehicles. In *2007 46th IEEE Conference on Decision and Control*, pages 4918--4923. ISSN 0191-2216.
- Joshi, A., Ashley, T., Huang, Y. R., and Bertozzi, A. L. (2009). Experimental validation of cooperative environmental boundary tracking with on-board sensors. In *Proceedings of the 2009 Conference on American Control Conference, ACC'09*, pages 2630--2635, Piscataway, NJ, USA. IEEE Press.
- Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4):321--331.
- Kemp, M., Bertozzi, A. L., and Marthaler, D. (2004). Multi-uuv perimeter surveillance. In *2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No.04CH37578)*, pages 102--107.
- Kiusalaas, J. (2013). *Numerical methods in engineering with Python 3*. Cambridge university press.
- Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1398--1404.
- Li, S., Guo, Y., and Bingham, B. (2014). Multi-robot cooperative control for monitoring and tracking dynamic plumes. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 67--73.

- Marthaler, D. and Bertozzi, A. (2003a). Collective motion algorithms for determining environmental boundaries. In *SIAM Conference on Applications of Dynamical Systems*.
- Marthaler, D. and Bertozzi, A. (2003b). Tracking environmental level sets with autonomous vehicles. *Journal of the Electrochemical Society*, 129:2865.
- Matveev, A. S., Hoy, M. C., Ovchinnikov, K., Anisimov, A., and Savkin, A. V. (2015). Robot navigation for monitoring unsteady environmental boundaries without field gradient estimation. *Automatica*, 62:227 – 235. ISSN 0005-1098.
- Matveev, A. S., Semakova, A. A., and Savkin, A. V. (2017). Tight circumnavigation of multiple moving targets based on a new method of tracking environmental boundaries. *Automatica*, 79:52 – 60. ISSN 0005-1098.
- Mellucci, C., Menon, P., Edwards, C., and Challenor, P. (2017). Experimental validation of boundary tracking using the suboptimal sliding mode algorithm. In *2017 American Control Conference*, pages 4878–4883.
- Menon, P. P., Edwards, C., Shtessel, Y. B., Ghose, D., and Haywood, J. (2014). Boundary tracking using a suboptimal sliding mode algorithm. In *53rd IEEE Conference on Decision and Control*, pages 5518–5523. ISSN 0191-2216.
- Menon, P. P. and Ghose, D. (2013). Boundary mapping of 3-dimensional regions. In *2013 American Control Conference*, pages 2984–2989. ISSN 0743-1619.
- Mottaghi, R. and Vaughan, R. (2006). An integrated particle filter and potential field method for cooperative robot target tracking. In *IEEE International Conference on Robotics and Automation. ICRA 2006.*, pages 1342--1347. IEEE.
- Nagarathna and 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)*, pages 1–6.
- Nagarathna, Valli, S., and Manjunath, D. (2014). A spatio-temporal model for estimation and efficient tracking of dynamic boundaries. In *Communications (NCC), 2014 Twentieth National Conference on*, pages 1–6.
- Nguyen, B., Chuang, Y.-L., Tung, D., Hsieh, C., Jin, Z., Shi, L., Marthaler, D., Bertozzi, A., and Murray, R. (2005). Virtual attractive-repulsive potentials for cooperative control of second order dynamic vehicles on the caltech mvwt. In *American*

- Control Conference, 2005. Proceedings of the 2005*, pages 1084–1089 vol. 2. ISSN 0743-1619.
- Sabattini, L., Secchi, C., and Fantuzzi, C. (2012). Closed-Curve Path Tracking for Decentralized Systems of Multiple Mobile Robots. *Journal of Intelligent & Robotic Systems*, pages 109–123. ISSN 0921-0296.
- Sabattini, L., Secchi, C., Fantuzzi, C., and Possamai, D. D. M. (2010). Tracking of Closed-Curve Trajectories for Multi-Robot Systems. pages 6089–6094.
- Saldaña, D., Assunção, R., and 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*, pages 1262–1267.
- Saldaña, D., Assunção, R., and Campos, M. (2016). Predicting environmental boundary behaviors with a mobile robot. *IEEE Robotics and Automation Letters*, 1(2):1133–1139. ISSN 2377-3766.
- Saldana, D., Javanmard Alitappeh, R., Pimenta, L., Assunção, R., and Campos, M. F. (2016). Dynamic perimeter surveillance with a team of robots. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1262–1267. IEEE.
- Shen, Z., He, Z., Li, S., Wang, Q., and Shao, Z. (2017). A multi-quadcopter cooperative cyber-physical system for timely air pollution localization. *ACM Trans. Embed. Comput. Syst.*, 16(3):70:1–70:23. ISSN 1539-9087.
- Sibley, G., Rahimi, M., and Sukhatme, G. (2002). Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In *IEEE International Conference on Robotics and Automation. Proceedings. ICRA 2002.*, volume 2, pages 1143–1148. IEEE.
- Simmons, G. F. (1963). *Topology and modern analysis*, volume 3. McGraw-Hill New York.
- Srinivasan, S., Ramamritham, K., and 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)*, pages 147–158.
- Srivastava, M. and Wu, Y. (1993). Comparison of ewma, cusum and shiryayev-roberts procedures for detecting a shift in the mean. *The Annals of Statistics*, pages 645–670.

- Susca, S., Bullo, F., and Martinez, S. (2008). Monitoring Environmental Boundaries With a Robotic Sensor Network. *IEEE Transactions on Control Systems Technology*, 16(2):288--296. ISSN 1063-6536.
- Susca, S., Bullo, F., and Martinez, S. (2009). Gradient algorithms for polygonal approximation of convex contours. *Automatica*, 45(2):510--516. ISSN 00051098.
- Temam, R. (2001). *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc.
- Turgeman, A. and Werner, H. (2017). Mission control - combined solutions for source seeking and level curve tracking in a time-varying field. In *2017 American Control Conference*, pages 4268--4273.
- White, B., Tsourdos, A., Ashokaraj, I., Subchan, S., and Zbikowski, R. (2007). Contaminant cloud boundary monitoring using uav sensor swarms. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6761.
- Zhang, F. and Leonard, N. E. (2005). Generating contour plots using multiple sensor platforms. In *IEEE Swarm Intelligence Symposium*, pages 309--316.
- Zhang, F. and Leonard, N. E. (2010). Cooperative filters and control for cooperative exploration. *IEEE Transactions on Automatic Control*, 55(3):650--663. ISSN 0018-9286.
- Zhang, G., Fricke, G. K., and Garg, D. P. (2013). Spill detection and perimeter surveillance via distributed swarming agents. *Mechatronics, IEEE/ASME Transactions on*, 18(1):121--129.