The International Journal of Robotics Research

http://ijr.sagepub.com/

Decentralized path planning for coverage tasks using gradient descent adaptive control

Daniel E Soltero, Mac Schwager and Daniela Rus

The International Journal of Robotics Research published online 29 October 2013
DOI: 10.1177/0278364913497241

The online version of this article can be found at:
http://ijr.sagepub.com/content/early/2013/10/29/0278364913497241

Published by:

http://www.sagepublications.com

On behalf of:

Multimedia Archives

Additional services and information for *The International Journal of Robotics Research* can be found at:

Email Alerts: http://ijr.sagepub.com/cgi/alerts

Subscriptions: http://ijr.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

>> OnlineFirst Version of Record - Oct 29, 2013

What is This?

Downloaded from ijr.sagepub.com at Massachusetts Institute of Technology on October 30, 2013

Article

The International Journal of Robotics Research
0(0) 1–25
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364913497241
ijr.sagepub.com

Decentralized path planning for coverage tasks using gradient descent adaptive control

Daniel E. Soltero[1], Mac Schwager[2] and Daniela Rus[1]

Abstract

In this paper we propose a new path planning algorithm for coverage tasks in unknown environments that does not rely on recursive search optimization. Given a sensory function that captures the interesting locations in the environment and can be learned, the goal is to compute a set of closed paths that allows a single robot or a multi-robot system to sense/cover the environment according to this function. We present an online adaptive distributed controller, based on gradient descent of a Voronoi-based cost function, that generates these closed paths, which the robots can travel for any coverage task, such as environmental mapping or surveillance. The controller uses local information only, and drives the robots to simultaneously identify the regions of interest and shape their paths online to sense these regions. Lyapunov theory is used to show asymptotic convergence of the system based on a Voronoi-based coverage criterion. Simulated and experimental results, that support the proposed approach, are presented for the single-robot and multi-robot cases in known and unknown environments.

Keywords

Adaptive control, path planning, Voronoi, coverage, unknown environment

1. Introduction

Many applications require robots to perform a given task in an unknown, partially known and/or dynamic environment. Robots operating under such conditions are faced with the problem of deciding where to go to get the most relevant information for their current task. We are particularly interested in the case where robots have to continually monitor or sense regions of interest for a long period of time. One way to solve this task is for the robots to generate closed paths that allow them to sense regions of interest. We will refer to such paths as *Interesting Closed Paths* (IC paths). The robots could then travel these closed paths for long periods of time until the monitoring task is done.

When the environment is known, the IC path planning problem consists of computing closed paths that allow the robots to sense the regions of interest in an optimal way according to some metric. However, when the environment is unknown, the robots must also learn the structure of the environment by identifying which regions are interesting and which are not. Hence, IC paths correspond to paths where the most sensory information can be obtained by the robots. In both cases (known or unknown environment), the key insight is to use a continuous-time control law based on Voronoi coverage, inspired by Schwager et al. (2009) and Cortes et al. (2004). In the unknown environment case, a parameter adaptation law inspired by Schwager et al. (2009) is used to estimate the environment description, which feeds into the Voronoi coverage controller.

In short, the proposed controller works as follows. The robots move along their paths, marking the areas they observe as *interesting* or not. While feeding these observations into a adaptation law that estimates the environment description, the robots simultaneously modify their paths, based on these estimates, by changing the location of the paths' waypoints in order to optimize a cost function related to the *informativeness* of the paths. By informative paths, we mean paths with which the robots are able to sense the regions of interest in the environment. By performing gradient descent on the cost function, the proposed controller attempts to place the paths' waypoints at the centroids of their Voronoi partitions, while also making sure that waypoints within the same path are not too far from one another. The result of this process is a set of paths that enable the

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
[2]Mechanical Engineering Department, Boston University, Boston, MA, USA

Corresponding author:
Daniel Eduardo Soltero, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Stata Center, 32 Vassar Street, Office 32-376, Cambridge, MA 02139, USA.
Email: soltero@mit.edu

robots to sense the regions of interest in the environment and avoid non-interesting regions. An example of the results of our proposed controller can be seen in Figure 1. As will be shown in later sections, this controller can be implemented in a distributed way and works by using a simple control action.

A wide range of applications require robots to move along IC paths in unknown environments in order to concentrate the robots' resources in the regions of interest and avoid wasting them in non-interesting regions. For example, our algorithm could be used to monitor underwater life by allowing a team of robots to estimate the regions where underwater activity is high (e.g. coral reefs), and generate paths that enable the team to visit locations of high underwater activity. Our algorithm can also be useful for city surveillance tasks, where the robots need to monitor high-crime locations; robot vacuum cleaning tasks, where the robots need to visit locations that accumulate dust and avoid locations that remain clean; or estimation tasks, where the robots need to keep an updated mental model of the system's state by visiting dynamic locations and avoiding visiting static locations.

The proposed IC path controller allows the user to have liberty of controlling the robots' speed along the IC paths. Hence, depending on the application, different speed control policies can be used. To provide an example of such liberty, in this paper we combine the IC path controller with a speed control algorithm presented in Smith et al. (2011) in order to provide what is called *persistent monitoring* of an environment. To avoid confusion and distraction from our main contributions, we will refrain from elaborating on persistent monitoring for now, but will do so in a later section.

1.1. Contribution

In Soltero et al. (2012), we presented the single-robot case of the IC path controller in unknown environments, but, due to lack of space, important features about the algorithm were omitted. In this paper we not only elaborate on the results of the single-robot case, but also develop the theory for the multi-robot case in known and unknown environments, provide stability proofs using Lyapunov theory, show simulated as well as experimental results, and highlight some interesting features and limitations of the controller.

1.2. Relation to previous work

Most of the previous work in path planning focuses on reaching a goal while avoiding collision with obstacles (e.g. Piazzi et al., 2007) or on computing an optimal path according to some metric (e.g. Kyriakopoulos and Saridis, 1988; Qin et al., 2004). Other works have focused on probabilistic approaches to path planning (e.g. Kavraki et al., 1996). More relevant to this work is the prior work in adaptive path planning (e.g. Stentz, 1994; Cunningham and Roberts, 2001), which considers adapting the robot's path to changes in the robot's knowledge of the environment or changes in the system's state.

The most relevant previous work is that of informative sensing, where the goal is to calculate paths that provide the most information for robots. The goal of Meliou et al. (2007) was to monitor a spatio-temporal phenomenon at a set of finite locations. The robot observes a subset of locations at any point in time and estimates other locations using a probability distribution, i.e. at each point in time they want to predict the values of all unobserved variables based on information collected up to that point. They solve the problem by planning closed paths that allow them to sense strategic locations according to some sub-modular metric. The work presented by Meliou et al. (2007) applies only to the single-robot case and is solved by search and iterative methods.

Binney et al. (2010) used previously obtained data to choose a path for an autonomous underwater vehicle (AUV) which gives the most additional information. This is done by optimizing over a sub-modular metric, which in their case is mutual information. Again, this solution is based on search and iterative optimization for a single robot, where the path is recursively divided in half and the next midpoint is searched and committed to before continuing to the next iteration.

Singh et al. (2009) used mutual information to obtain informative paths for a collection of robots, subject to constraints on the cost incurred by each robot (e.g. limited battery capacity). This problem is also solved by a search and iterative optimization. In addition, the currently published work is an offline algorithm, i.e. does not adapt the path with new data, and although it treats multiple robots, the algorithm is centralized. It plans the paths for each robot sequentially, each one taking into consideration all previous robots' paths. Other variations on informative path planning include maintaining periodic connectivity in order for robots to share information and synchronize (Hollinger and Singh, 2010), and incorporating complex vehicle dynamics and sensor limitations (Levine et al., 2010).

All of the mentioned previous work in informative sensing and adaptive path planning, as with most in path planning in general, treats the path planning problem as a search and/or recursive problem, with running times that can be very high for large systems, and some of them only providing approximations of optimal solutions due to intractability of the problem. Also, many of them are centralized algorithms since path planning in a distributed way is a very difficult problem. In contrast to all of these previous works, we approach the path planning problem in a way that resembles the train of thought of artificial potential fields (Khatib, 1985) and behavioral dynamics (Large et al., 1999), where a dynamical systems approach is used in path planning and control. The idea behind these approaches is to generate a control law that governs robot behavior/motion based on

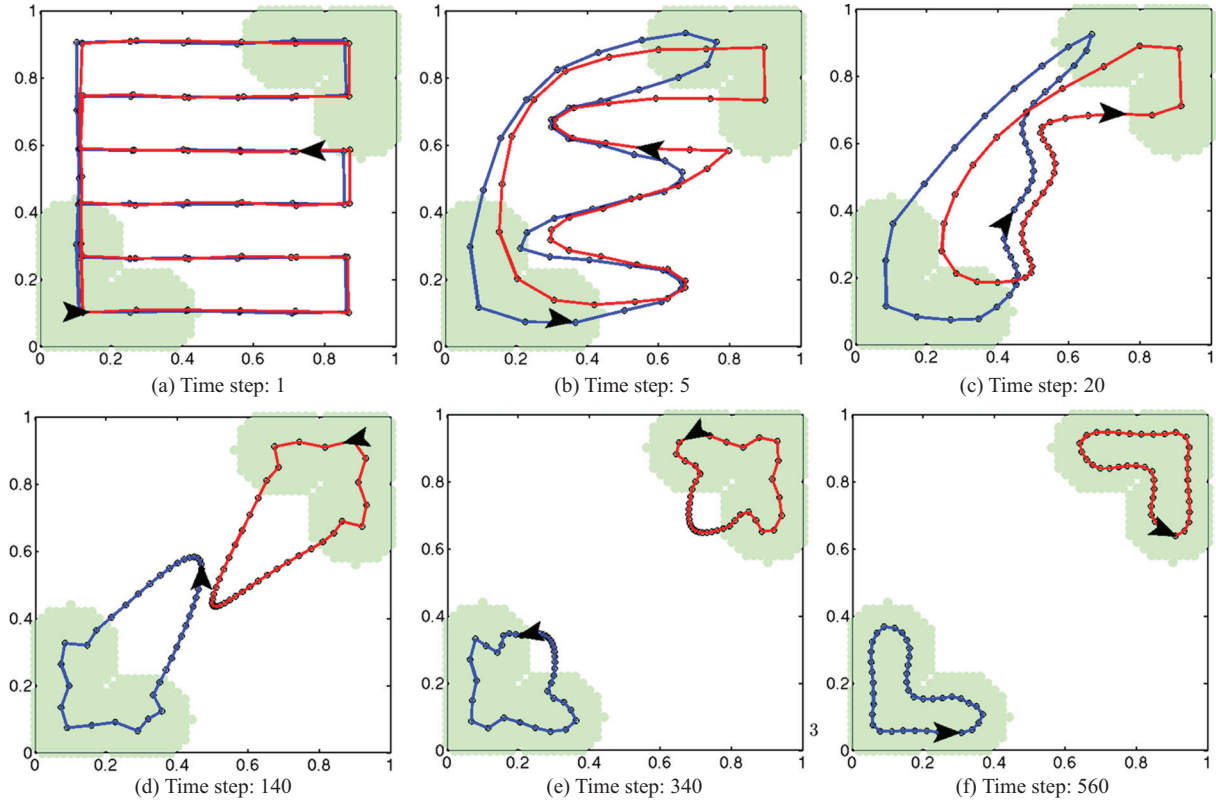


Fig. 1. Simulated example of path shaping process for two robots with the IC path controller. The paths, shown as red and blue lines, connect the waypoints, shown as black circles. The blue path corresponds to robot 1, and the red path corresponds to robot 2. The robots' positions are represented by the black arrow heads. The regions of interest in the environment, which the robots are tasked to cover, are represented by the light green regions. As time passes by, the paths are shaped such that the robots can cover the regions of interest. Each time step corresponds to a period of 0.01 seconds.

behavioral dynamics and competition between constraints. One good feature of these kinds of solutions is that they can be much easier to implement in a decentralized way. In our case, we use a type of artificial potential field based on Voronoi partitions, but this potential field does not govern the robots' behaviors. Rather it governs the behavior of waypoints that define the paths that the robots are following.

We approach the path planning problem as a continuous-time dynamical systems control problem. Furthermore, we focus most of our attention on the problem of path planning in unknown environments, where we use gradient-based adaptive control tools to control the location of waypoints that define robot paths. These gradient-based tools seek to locally optimize a Voronoi-based cost function related to the position of the waypoints. The Voronoi-based nature of our controller is partly inspired by the work of Graham and Cortes (2012), where a form of generalized Voronoi decomposition was used to design a sampling trajectory that minimized the uncertainty of an estimated random field. In our work, we also use a Voronoi decomposition as the basis of our algorithms, but the problem we are treating is different. In our work the field is not random (although unknown) and we are not concerned with minimizing predictive variance, but rather we focus on generating closed paths by optimizing the location of waypoints defining the paths, according to a Voronoi-based coverage criterion in an unknown environment. Because of this, our approach builds upon the work of Schwager et al. (2009) and Cortes et al. (2004), where a group of agents were coordinated to place themselves in static, locally optimal locations to sense an environment that is known (in the case of Cortes et al., 2004) or unknown (in the case of Schwager et al., 2009). In these works, Voronoi partitions were used to position the robots, while a parameter adaptation law allowed the agents to learn the environment model. We build upon these works by letting the waypoints in the robots' paths define a Voronoi decomposition, and using a parameter adaptation law to learn which regions of the environment the robots should focus on sensing. This Voronoi-based adaptive controller provides a simple and distributed way of generating useful paths for the robots, with strong and simple stability guarantees.

The paper is structured as follows. In Section 2 we set up the problem, present locational optimization tools, and present a basis function parameterization of the environment. Section 3 introduces the IC path controller for the case where the environment in known to the robots. We then treat the case of robots operating in unknown environments

in Section 4. After that, in Section 5, we show how controlling the speeds of the robots along the IC paths can be used to perform persistent monitoring tasks in unknown environments, and we provide results from an implementation with two quadrotors. Finally, we talk about some limitations and future work in Section 6 and present our conclusions in Section 7.

2. Problem formulation

We will use the convention of boldface variables to represent vectors. We assume we are given a number of robots whose tasks are to sense an environment while traveling along their individual closed paths, each consisting of a finite number of waypoints. The goal is for the robots to identify the regions within the environment that are of interest and compute paths that allow them to jointly sense these regions. A formal mathematical description of the problem follows.

Let there be N robots identified by $r \in \{1, \dots, N\}$ operating in a convex, bounded area $Q \subset \mathbb{R}^2$. An arbitrary point in Q is denoted $\mathbf{q}$. Robot r is located at position $\mathbf{p}_r(t) \in Q$ and travels along a closed path consisting of a finite number $n(r)$ of waypoints. The $n(r)$ waypoints corresponding to robot r are different from the $n(r')$ waypoints corresponding to robot r', $\forall r' \neq r$. The position of the ith waypoint in robot r's path is denoted by $\mathbf{p}_i^r$, $i \in \{1, \dots, n(r)\}$. Let $\{\mathbf{p}_i^r, \dots, \mathbf{p}_{n(r)}^r\}$ be the configuration of robot r's path and let V_i^r be a Voronoi partition of Q, with the ith waypoint position in robot r's path as the generator point, defined as

$$V_i^r = \{\mathbf{q} \in Q \mid \|\mathbf{q} - \mathbf{p}_i^r\| \leq \|\mathbf{q} - \mathbf{p}_{i'}^{r'}\|, \forall (r', i') \neq (r, i)\} \quad \text{where } r, r' \in \{1, \dots, N\},\ i \in \{1, \dots, n(r)\} \text{ and } i' \in \{1, \dots, n(r')\}, \tag{1}$$

where $\|\cdot\|$ denotes the l^2-norm. We assume that the robot is able to compute the Voronoi partitions based on the waypoint locations, as is common in the literature (Salapaka et al., 2003; Cortes et al., 2004; Schwager et al., 2009).

Since each path is closed, each waypoint $\mathbf{p}_i^r$ along robot r's path has a previous waypoint $\mathbf{p}_{i-1}^r$ and next waypoint $\mathbf{p}_{i+1}^r$ related to it (in the same path), which are referred to as the *neighbor waypoints* of waypoint $\mathbf{p}_i^r$. Note that for each robot r's path, $i + 1 = 1$ for $i = n(r)$, and $i - 1 = n(r)$ for $i = 1$. Once a robot reaches a waypoint in its path, it continues to move to the next waypoint in its path, in a straight line interpolation.[1]

A sensory function, defined as a map $\phi : Q \mapsto \mathbb{R}_{\geq 0}$ (where $\mathbb{R}_{\geq 0}$ refers to non-negative scalars) determines the *interestingness* of the environment at point $\mathbf{q} \in Q$. Each robot r is equipped with a sensor to make a point measurement of $\phi(\mathbf{p}_r(t))$ at its position $\mathbf{p}_r(t)$ at time t. The regions of interest, which the robots are tasked to sense/cover, are defined as follows:

Definition 2.1 (Regions of interest)**.** The regions of interest are the collection of points in the environment where the sensory function is positive, i.e. the set $\{\mathbf{q} \in Q \mid \phi(\mathbf{q}) > 0\}$.

Remark 2.2. The interpretation of the sensory function $\phi(\mathbf{q})$ may be adjusted for a broad range of applications. It can be any kind of weighting of importance for points $\mathbf{q} \in Q$. To maintain generality, we treat it as the degree of interestingness of point $\mathbf{q} \in Q$. Note that this sensory function is time-invariant.

2.1. Locational optimization

Building upon the work of Schwager et al. (2009) and Cortes et al. (2004), we now formulate a cost function $\mathcal{H}$ that encodes the performance of the robots in the task of covering/sensing the regions of interest over the region Q. This cost function is a function of the collection of waypoints for all robots (the robots' paths), i.e. $\mathbf{p}_i^r$, $\forall i, r$ and the environment's sensory function $\phi(\mathbf{q})$, $\forall \mathbf{q} \in Q$. We define $\mathcal{H}$ as

$$\mathcal{H} = \sum_{r=1}^{N} \sum_{i=1}^{n(r)} \int_{V_i^r} \frac{W_s}{2} \|\mathbf{q} - \mathbf{p}_i^r\|^2 \phi(\mathbf{q})\, d\mathbf{q} + \sum_{r=1}^{N} \sum_{i=1}^{n(r)} \frac{W_n}{2} \|\mathbf{p}_i^r - \mathbf{p}_{i+1}^r\|^2, \tag{2}$$

where $\|\mathbf{q} - \mathbf{p}_i^r\|$ can be interpreted as the unreliability of measuring the sensory function $\phi(\mathbf{q})$ when robot r is at $\mathbf{p}_i^r$, and $\|\mathbf{p}_i^r - \mathbf{p}_{i+1}^r\|$ can be interpreted as the cost of a waypoint being too far away from a neighboring waypoint. The two parameters W_s and W_n are constant positive scalar weights that the user can specify for the sensing task and neighbor distance, respectively.

This cost function says two things: (1) paths that do not provide good/reliable sensing of the regions of interest are expensive; and (2) having neighboring waypoints (in the same path) be too far apart from each other is expensive. Consequently, by designing a controller that performs a gradient descent on this cost function, we introduce two primitives for the waypoint movements in each path. The first primitive drives the waypoints to move towards their Voronoi centroids in order to properly cover/sense the environment. The second primitive makes each waypoint pull its neighboring waypoints, which couples the waypoints in order to generate closed paths. The relative strength of the first and second primitive is specified by the user's selection of the parameters W_s and W_n, respectively. A formal definition of IC paths follows.

Definition 2.3 (IC paths)**.** A collection of IC paths corresponds to a set of waypoint locations for all robots that locally minimizes (2).

Next we define three properties analogous to mass-moments of rigid bodies. The mass, first mass-moment, and centroid of the Voronoi partition V_i^r are defined as

$$M_i^r = \int_{V_i^r} W_s \phi(\mathbf{q})\, d\mathbf{q}, \;\; \mathbf{L}_i^r = \int_{V_i^r} W_s \mathbf{q} \phi(\mathbf{q})\, d\mathbf{q},$$

$$\mathbf{C}_i^r = \frac{\mathbf{L}_i^r}{M_i^r}, \tag{3}$$

respectively. Also, let $\mathbf{e}_i^r = \mathbf{C}_i^r - \mathbf{p}_i^r$, which is the error between robot r's ith waypoint position and its Voronoi centroid.

From locational optimization (Drezner, 1995), and from differentiation under the integral sign for Voronoi partitions (Pimenta et al., 2008) we have

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i^r} &= -\int_{V_i^r} W_s(\mathbf{q} - \mathbf{p}_i^r)\phi(\mathbf{q})\, d\mathbf{q} \\ &\quad - W_n(\mathbf{p}_{i+1}^r - \mathbf{p}_i^r) - W_n(\mathbf{p}_{i-1}^r - \mathbf{p}_i^r) \\ &= -(\mathbf{L}_i^r - \mathbf{p}_i^r M_i^r) - W_n(\mathbf{p}_{i+1}^r + \mathbf{p}_{i-1}^r - 2\mathbf{p}_i^r) \\ &= -M_i^r(\mathbf{C}_i^r - \mathbf{p}_i^r) - W_n(\mathbf{p}_{i+1}^r + \mathbf{p}_{i-1}^r - 2\mathbf{p}_i^r) \\ &= -M_i \mathbf{e}_i - W_n(\mathbf{p}_{i+1}^r + \mathbf{p}_{i-1}^r - 2\mathbf{p}_i^r). \end{aligned} \tag{4}$$

Assigning to each waypoint dynamics of the form

$$\dot{\mathbf{p}}_i^r = \mathbf{u}_i^r, \tag{5}$$

where $\mathbf{u}_i^r$ is the control input,[2] we propose the following control law, based on gradient descent, for the waypoints to converge to an equilibrium configuration:

$$\begin{aligned}\mathbf{u}_i^r &= \frac{K_i^r}{\beta_i^r}\left(-\frac{\partial H}{\partial \mathbf{p}_i^r}\right) \\ &= \frac{K_i^r(M_i^r \mathbf{e}_i^r + \boldsymbol{\alpha}_i^r)}{\beta_i^r}, \end{aligned} \tag{6}$$

where

$$\begin{aligned}\boldsymbol{\alpha}_i^r &= W_n(\mathbf{p}_{i+1}^r + \mathbf{p}_{i-1}^r - 2\mathbf{p}_i^r), \\ \beta_i^r &= M_i^r + 2W_n, \end{aligned}$$

and K_i^r is a uniformly (potentially time-varying) positive definite matrix. Note, in (6), that the term $\mathbf{e}_i^r$ introduces the first primitive in the waypoints' movements (go towards the Voronoi centroid to provide good sensing of the regions of interest), and the term $\boldsymbol{\alpha}_i^r$ introduces the second primitive (stay close to your neighbor waypoints to create a closed path).

Remark 2.4. The term β_i^r is a positive scalar and has the nice effect of normalizing the weight distribution between sensing and staying close to neighboring waypoints.

2.2. Sensory function parameterization

We assume that the sensory function $\phi(\mathbf{q})$ can be parameterized as a linear combination of a set of known basis functions. That is, we make the following assumption.

Assumption 2.5 (Matching condition). There exists $\mathbf{a} \in \mathbb{R}^m_{\geq 0}$ and $\mathcal{K} : \mathcal{Q} \mapsto \mathbb{R}^m_{\geq 0}$, where $\mathbb{R}^m_{\geq 0}$ is a vector of non-negative entries, such that

$$\phi(\mathbf{q}) = \mathcal{K}(\mathbf{q})^{\mathrm{T}} \mathbf{a}, \tag{7}$$

where the vector of basis functions $\mathcal{K}(\mathbf{q})$ is known by the robots.

Such an assumption is common for adaptive controllers (Schwager et al., 2009).

3. IC path controllers for known environments

In this section we assume that the robots have knowledge of the sensory function $\phi(\mathbf{q})$, which under Assumption 2.5, translates into knowledge of the parameter vector $\mathbf{a}$. We now prove that the proposed control law in (6) causes the collection of robot paths to converge to IC paths, i.e. a locally optimal configuration according to (2).

Theorem 3.1 (Convergence theorem for known environments)**.** *Assuming that the environment is known by the robots, with waypoint dynamics specified by (5) and control law specified by (6), we have*

$$\begin{aligned}&\lim_{t\to\infty} \| M_i^r(t)\, \mathbf{e}_i^r(t) + \boldsymbol{\alpha}_i^r(t) \| = 0, \\ &\quad \forall r \in \{1, \ldots, N\},\ \forall i \in \{1, \ldots, n(r)\}, \end{aligned}$$

Proof. We define a Lyapunov-like function based on the robots' paths, and use Barbalat's lemma (Ioannou and Sun, 1996) to prove asymptotic convergence of the system to a locally optimal equilibrium.

We define a Lyapunov-like function $\mathcal{V}_1$ as

$$\mathcal{V}_1 = \mathcal{H}. \tag{8}$$

Taking the time derivative of $\mathcal{V}_1$, we obtain

$$\begin{aligned}\dot{\mathcal{V}}_1 &= \sum_{r=1}^{N} \sum_{i=1}^{n(r)} \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i^r}^{\mathrm{T}} \dot{\mathbf{p}}_i^r \\ &= \sum_{r=1}^{N} \sum_{i=1}^{n(r)} -(M_i^r \mathbf{e}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r. \end{aligned}$$

Substituting the dynamics specified by (5) and control law specified by (6), we obtain

$$\dot{\mathcal{V}}_1 = \sum_{r=1}^{N} \sum_{i=1}^{n(r)} -\frac{1}{\beta_i^r}(M_i^r \mathbf{e}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (M_i^r \mathbf{e}_i^r + \boldsymbol{\alpha}_i^r). \tag{9}$$

Note that $\dot{\mathcal{V}}_1(t) \leq 0$ since K_i^r is uniformly positive definite and $\beta_i^r > 0$. Since $\dot{\mathcal{V}}_1(t)$ is non-positive and $\mathcal{V}_1 > 0$, this means that $\mathcal{V}_1$ is non-increasing and lower bounded, hence $\lim_{t\to\infty} \mathcal{V}_1(t)$ exists and is finite. Furthermore, it was shown by Schwager et al. (2009, Lemma 1) that $\ddot{\mathcal{V}}_1(t)$ is uniformly bounded, therefore $\dot{\mathcal{V}}_1(t)$ is uniformly continuous. Hence, by Barbalat's lemma, $\dot{\mathcal{V}}_1(t) \to 0$, which implies that $\| M_i^r(t)\, \mathbf{e}_i^r(t) + \boldsymbol{\alpha}_i^r(t) \| \to 0$. □

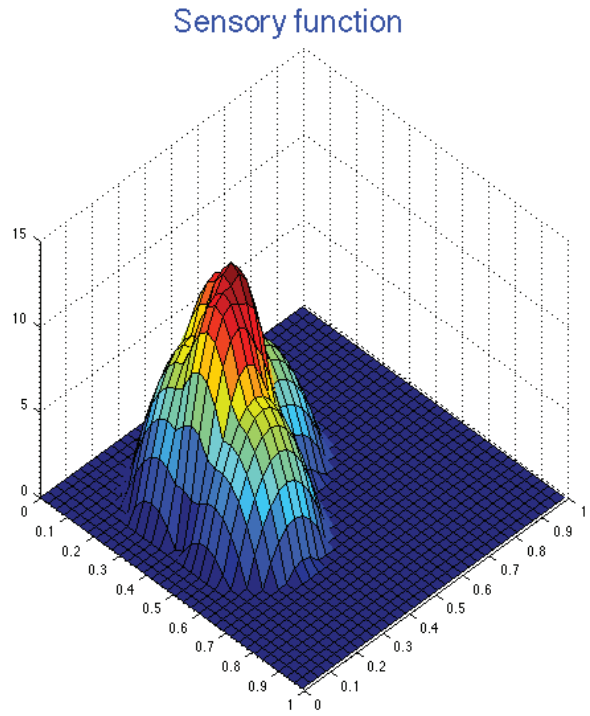


Fig. 2. Environment's sensory function

Remark 3.2. Theorem 3.1 implies that the paths will reach a locally optimal configuration for sensing regions of interest, where the waypoints reach a stable balance between providing good sensing locations and being close to their neighbor waypoints. In addition, this balance can be tuned (short paths versus good coverage) by the proper selection of W_n and W_s.

3.1. Simulated results

In this section we simulate the IC path controller for known environments for the simple case of a single robot, i.e. $N = 1$ and, therefore, $r = 1$. The simulation is performed in MATLAB. Here we present a case for $n(r) = 40$ waypoints. A fixed-time step numerical solver is used to integrate the equations of motion using a time step length of 0.01 seconds. The region Q is taken to be the unit square. The sensory function $\phi(\mathbf{q})$ is parametrized as a Gaussian network with 25 truncated Gaussians, i.e. $\mathcal{K} = [\mathcal{K}_1 \ \cdots \ \mathcal{K}_{25}]^{\mathrm{T}}$, where

$$G_j = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{\|\mathbf{q}-\boldsymbol{\mu}_j\|^2}{2\sigma^2}\right\},$$
$$G_{\mathrm{trunc}} = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{\rho_{\mathrm{trunc}}{}^2}{2\sigma^2}\right\},$$
$$\mathcal{K}_j = \begin{cases} G_j - G_{\mathrm{trunc}}, & \text{if } \|\mathbf{q}-\mu_j\| < \rho_{\mathrm{trunc}}, \\ 0, & \text{otherwise}, \end{cases} \quad (10)$$

$\sigma = 0.4$ and $\rho_{\mathrm{trunc}} = 0.2$. The unit square is divided into an even 5×5 grid and each truncated Gaussian mean $\boldsymbol{\mu}_j$ is chosen so that each of the 25 Gaussians is centered at its corresponding grid square. The environment sensory function $\phi(\mathbf{q})$ is created by selecting the parameter vector $\mathbf{a}$ such that $\mathbf{a}(7) = 80$, $\mathbf{a}(8) = 60$, $\mathbf{a}(12) = 70$, and $\mathbf{a}(j) = 0$ otherwise.[3] The environment's sensory function created with these parameters can be seen in Figure 2.

The parameters for the controller are $K_i^r = 70$, $\forall i$, $W_n = 5$, and $W_s = 150$. The spatial integrals are approximated by discretizing each Voronoi partition into a 7×7 grid and summing contributions of the integrand over the grid. Voronoi regions are computed using a decentralized algorithm similar to that of Cortes et al. (2004).

The initial path is arbitrarily designed to "zig-zag" across the environment (see Figure 3a). The region of interest in the environment is shown as a green-colored region in Figure 3. We can see how the path evolves under the IC path controller in Figure 3. After 25 time steps (Figure 3c), the path already tends to go through the region of interest, while avoiding all other regions. At 200 time steps (Figure 3f), the path clearly only goes through the region of interest. This final IC path corresponds to a local minimum of the Lyapunov-like function $\mathcal{V}_1$.

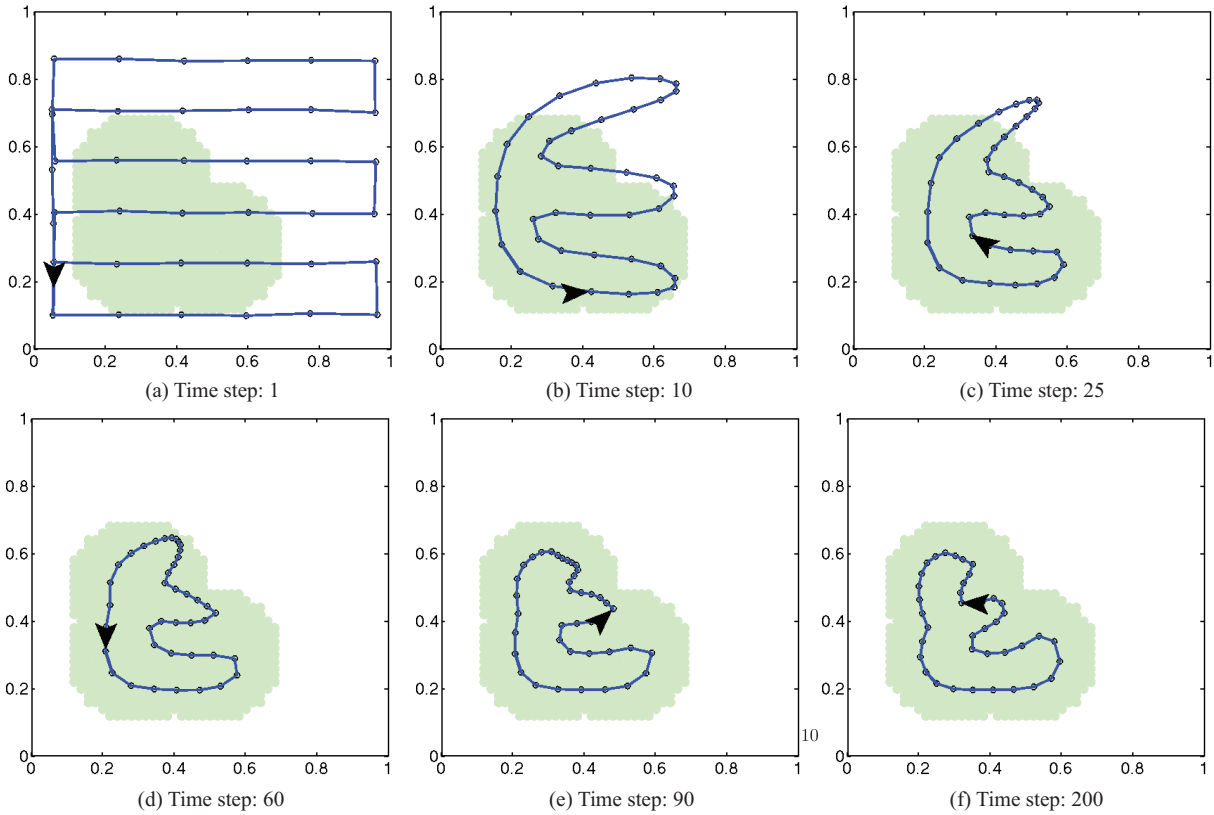
Algorithm 1 IC path controller for the ith waypoint p_i^r in robot r's path in a known environment.

Require: Ability to calculate Voronoi partition
Require: Knowledge of its two neighboring waypoints $\mathbf{p}_{i-1}^r$ and $\mathbf{p}_{i+1}^r$
1: **loop**
2: Compute the waypoint's Voronoi partition
3: Compute $\mathbf{C}_i$ according to (3)
4: Obtain neighbor waypoint locations $\mathbf{p}_{i-1}^r$ and $\mathbf{p}_{i+1}^r$
5: Compute $\mathbf{u}_i^r$ according to (6)
6: Update $\mathbf{p}_i^r$ according to (5)
7: **end loop**

3.2. Discussion

The IC path algorithm's strength lies in its simplicity and in being capable of decentralized implementation. In a fully decentralized implementation, each waypoint in the robot's path has its own controller and performs its own calculations based only on local data. This can be accomplished by having each robot run a separate control thread for each waypoint in its path that performs the calculations in Algorithm 1.

In order to perform these decentralized calculations, each waypoint control thread must be able to compute the corresponding waypoint's Voronoi partition. There are decentralized ways of calculating such Voronoi partition, as shown by Cortes et al. (2004). In addition, each waypoint control thread must be able to communicate to its neighboring waypoint controllers (corresponding to the previous and next waypoints in the respective path). Since the robot performing the control thread for waypoint $\mathbf{p}_i^r$ knows the location



(a) Time step: 1 (b) Time step: 10 (c) Time step: 25

(d) Time step: 60 (e) Time step: 90 (f) Time step: 200

Fig. 3. Simulated single-robot system in a known environment using the IC path controller. The path is shown as a blue line connecting all waypoints, shown as black circles. The black arrowhead is the simulated robot's position. The region of interest is shown as a light-green-colored region.

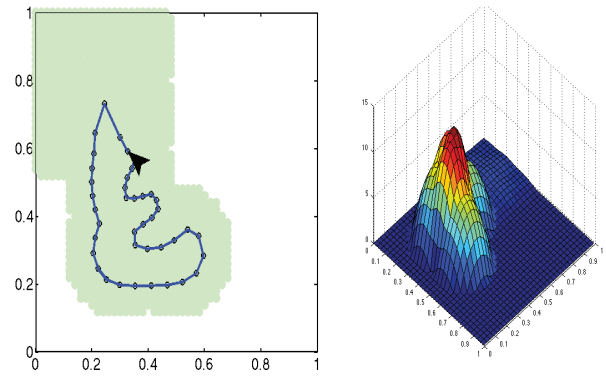
of all other waypoints in its own path, this last requirement is easily satisfied. If the robots do not have the computational power to run a separate control thread for each of their corresponding waypoints, the implementation can still be somewhat decentralized, since each robot can individually perform Algorithm 1 for all of its corresponding waypoints in a serial fashion. As can be seen, this algorithm is very simple and is independent of the number of waypoints in the path. The difficulty of implementing this algorithm mainly lies in the lower-level difficulty of calculating the Voronoi partitions.

Another very useful feature of this controller is that, when the environment is known, it responds to changes in the environment sensory function, i.e. it works for time-varying versions of $\phi(\mathbf{q})$. We tested this claim by transitioning the $\mathbf{a}$ vector described in the previous simulation to a new vector starting after the 200th time step, seen in Figure 3f. The new $\mathbf{a}$ vector is $\mathbf{a}(4) = 40$, $\mathbf{a}(5) = 40$, $\mathbf{a}(9) = 80$, $\mathbf{a}(10) = 60$, and $\mathbf{a}(j) = 0$ otherwise. The process by which $\phi(\mathbf{q})$ changes and the path responds to these changes can be seen in Figure 4. As can be seen, the path successfully reshapes in order to provide the robot with good sensing locations for the new region of interest. It is important to note that this change in behavior is a result of only changing the $\mathbf{a}$ vector through time, i.e. the controller shown in Algorithm 1 was untouched.

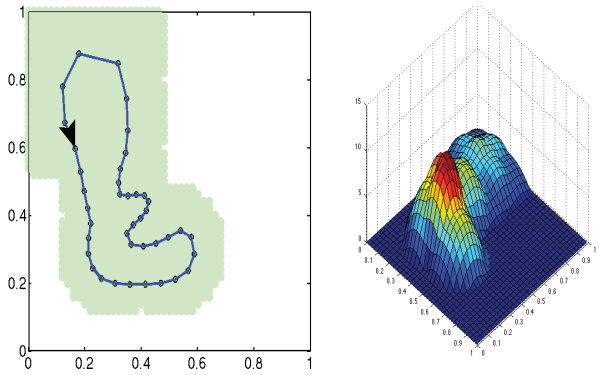
Finally, with respect to the controller, the selection of the parameters W_s and W_n can vary depending on the desired behavior of the system. For example, a higher W_n/W_s ratio will pull waypoints closer together, which might drive the system to a more sub-optimal configuration for sensing. If, in contrast, W_n/W_s is small, the system focuses more on the sensing task, and the waypoints in the path will be more scattered. Although in this work we do not consider vehicle dynamics, one way to indirectly impose vehicle dynamics restrictions is to increase the neighbor distance weight W_n and increase the number of waypoints in the path. This will result in smoother paths.

4. IC path controllers for unknown environments

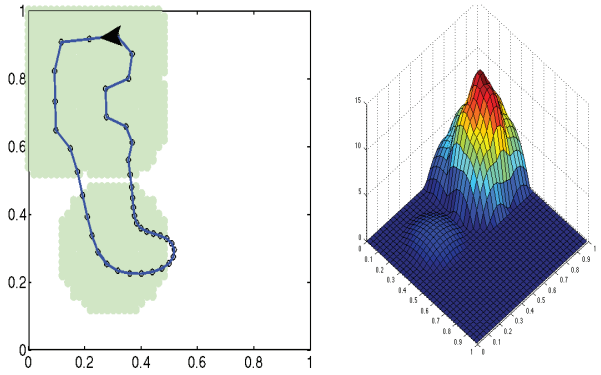
In this section, we relax the assumption that the environment sensory function $\phi(\mathbf{q})$ is known to the robots. Under Assumption 2.5, this directly translates to the robots not knowing the parameter vector $\mathbf{a}$. Therefore, the robots must estimate this parameter vector and the IC path controller must make its decision based on the estimate of $\mathbf{a}$. In order to estimate the parameter vector $\mathbf{a}$, we will use an adaptation law that assumes that the robots have sensors capable of making point measurements of $\phi(\mathbf{q})$ at the robot's location $\mathbf{p}_r(t)$ at time t. That is, $\phi(\mathbf{p}_r(t))$ is known to the robot.



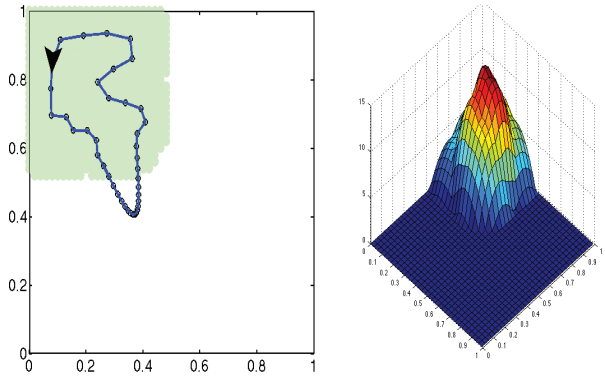
(a) Time step: 205



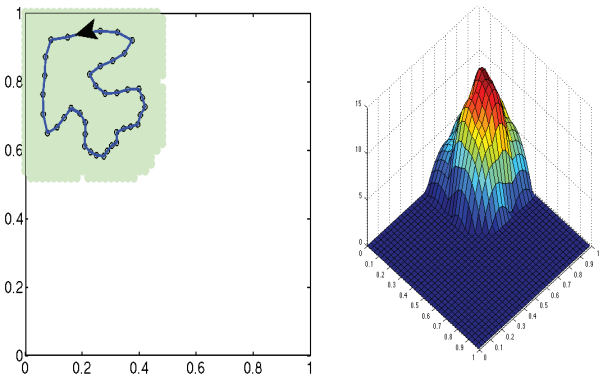
(b) Time step: 220



(c) Time step: 270



(d) Time step: 390



(e) Time step: 700

Fig. 4. Robot path responding properly to a time-varying $\phi(\mathbf{q})$ with the IC path controller when the environment is known to the robot. On the left: the path is shown as a blue line connecting all waypoints (shown as black circles), the black arrowhead is the simulated robot's position, and the regions of interest are shown as light green-colored regions. On the right: the environment's sensory function $\phi(\mathbf{q})$.

Let $\hat{\mathbf{a}}_r(t)$ be robot r's approximation of the parameter vector $\mathbf{a}$. Then,

$$\hat{\phi}_r(\mathbf{q}) = \mathcal{K}(\mathbf{q})^{\mathrm{T}}\hat{\mathbf{a}}_r \tag{11}$$

is robot r's approximation of $\phi(\mathbf{q})$. Building on this, we define the mass-moment approximations as

$$\hat{M}_i^r = \int_{V_i^r} W_s\hat{\phi}_r(\mathbf{q})\, d\mathbf{q}, \quad \hat{\mathbf{L}}_i^r = \int_{V_i^r} W_s\mathbf{q}\hat{\phi}_r(\mathbf{q})\, d\mathbf{q},$$

$$\hat{\mathbf{C}}_i^r = \frac{\hat{\mathbf{L}}_i^r}{\hat{M}_i^r}. \tag{12}$$

In addition, we can define the parameter estimate error for robot r as $\tilde{\mathbf{a}}_r = \hat{\mathbf{a}}_r - \mathbf{a}$, and the sensory function error and mass-moment errors as

$$\tilde{\phi}_r(\mathbf{q}) = \hat{\phi}_r(\mathbf{q}) - \phi(\mathbf{q}) = \mathcal{K}(\mathbf{q})^{\mathrm{T}}\,\tilde{\mathbf{a}}_r, \tag{13}$$

$$\tilde{M}_i^r = \hat{M}_i^r - M_i^r = \int_{V_i^r} W_s\mathcal{K}(\mathbf{q})^{\mathrm{T}}\, d\mathbf{q}\,\tilde{\mathbf{a}}_r, \tag{14}$$

$$\tilde{\mathbf{L}}_i^r = \hat{\mathbf{L}}_i^r - \mathbf{L}_i^r = \int_{V_i^r} W_s\mathbf{q}\mathcal{K}(\mathbf{q})^{\mathrm{T}}\, d\mathbf{q}\,\tilde{\mathbf{a}}_r, \tag{15}$$

$$\tilde{\mathbf{C}}_i^r = \frac{\tilde{\mathbf{L}}_i^r}{\tilde{M}_i^r}. \tag{16}$$

4.1. Adaptive controller

We design a distributed adaptive control law for multiple robots and prove that it causes the set of paths for all robots to converge to a locally optimal configuration according to (2), while causing all of the robots' estimates of the environment's sensory function to converge to the real description (assuming their trajectories are rich enough). The latter is achieved by the robots integrating their sensory measurements along their trajectories and sharing their estimates with each other through a consensus term (Schwager et al., 2009). In order for the robots to properly share their estimates of the sensory function, we must make the following assumption.

Assumption 4.1 (Connected network). When working with more than one robot, i.e. $N > 1$, we assume that the network of robots in the system is a connected network, i.e. that the graph where each robot is a node and an edge represents communication between two nodes is a connected graph. This connected network corresponds to the robots' abilities to communicate with each other.

Since the robots do not know $\phi(\mathbf{q})$, but have estimates $\hat{\phi}_r(\mathbf{q})$ of it, the control law from (6) becomes

$$\mathbf{u}_i^r = \frac{K_i^r(\hat{M}_i^r\hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)}{\hat{\beta}_i^r}, \tag{17}$$

where

$$\boldsymbol{\alpha}_i^r = W_n(\mathbf{p}_{i+1}^r + \mathbf{p}_{i-1}^r - 2\mathbf{p}_i^r),$$
$$\hat{\beta}_i^r = \hat{M}_i^r + 2W_n,$$
$$\hat{\mathbf{e}}_i^r = \hat{\mathbf{C}}_i^r - \mathbf{p}_i^r.$$

Robot r's parameter vector $\hat{\mathbf{a}}_r$ is adjusted according to an adaptation law which is described next. Let

$$\Lambda_r = \int_0^t w_r(\tau)\,\mathcal{K}(\mathbf{p}_r(\tau))\,\mathcal{K}(\mathbf{p}_r(\tau))^{\mathrm{T}}\, d\tau, \tag{18}$$

$$\boldsymbol{\lambda}_r = \int_0^t w_r(\tau)\,\mathcal{K}(\mathbf{p}_r(\tau))\,\phi(\mathbf{p}_r(\tau))\, d\tau, \tag{19}$$

where $w_r(t)$ is a non-negative scalar weight function for data collection (Schwager et al., 2009), whose selection is restricted such that it maintains Λ_r and $\boldsymbol{\lambda}_r$ bounded. We will discuss the selection of this weight function in a later section. Let

$$\mathbf{b}_r = \sum_{i=1}^{n(r)} \int_{V_i^r} W_s\mathcal{K}(\mathbf{q})\,(\mathbf{q} - \mathbf{p}_i^r)^{\mathrm{T}}\, d\mathbf{q}\,\dot{\mathbf{p}}_i^r, \tag{20}$$

$$\dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r} = -\mathbf{b}_r - \gamma(\Lambda_r\hat{\mathbf{a}}_r - \boldsymbol{\lambda}_r) - \zeta\sum_{r'=1}^{N} l_{r,r'}(\hat{\mathbf{a}}_r - \hat{\mathbf{a}}_{r'}), \tag{21}$$

where $\gamma > 0$ is a scalar adaptation gain. The last term in (21) corresponds to a consensus term from Schwager et al. (2009) to allow the robots to share their estimates, where $\zeta > 0$ is a consensus scalar gain, and $l_{r,r'} \geq 0$ can be interpreted as the strength of the communication between robots r and r'.

We restrict the selection of $l_{r,r'}$ such that Assumption 4.1 is satisfied (the network of robots is connected) and such that (21) maintains continuity on the right-hand side, so we can apply Barbalat's lemma in our stability proof. There are a number of definitions for $l_{r,r'}$ that satisfy these restrictions. For example, we can define $l_{r,r'}$ to be the length of the shared Voronoi edge between robots r and r' in a new Voronoi decomposition where each robot is a generator point (Schwager et al., 2009). Another option is to define $l_{r,r'}$ to be a constant for all r, r' (which corresponds to an all-to-all communication between robots). For simplicity of implementation, we will use the latter ($l_{r,r'}$ is a constant) in our simulations and implementations. However, to maintain generality, we will not make further assumptions on the definition of $l_{r,r'}$ through our proofs, other than Assumption 4.1 and continuity.

Since $\mathbf{a}(j) \geq 0$, $\forall j$ (where $\mathbf{a}(j)$ denotes the jth element of $\mathbf{a}$), we enforce $\hat{\mathbf{a}}_r(j) \geq 0$, $\forall r, \forall j$. We do this by using a projection law (Schwager et al., 2009),

$$\dot{\hat{\mathbf{a}}}_r = \Gamma(\dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r} - I_{\mathrm{proj}_r}\dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}), \tag{22}$$

where $\Gamma \in \mathbb{R}^{m\times m}$ is a diagonal positive-definite adaptation gain matrix, and the diagonal matrix I_{proj_r} is defined element-wise as

$$I_{\mathrm{proj}_r}(j) = \begin{cases} 0, & \text{if } \hat{\mathbf{a}}_r(j) > 0, \\ 0, & \text{if } \hat{\mathbf{a}}_r(j) = 0 \text{ and } \dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}(j) \geq 0, \\ 1, & \text{otherwise,} \end{cases} \tag{23}$$

where (j) denotes the jth element for a vector and the jth diagonal element for a matrix.

Theorem 4.2 (Convergence theorem for unknown environments)**.** *Under Assumptions 2.5 and 4.1, with waypoint dynamics specified by (5), control law specified by (17) and adaptive law specified by (22), we have:*

(i) $\lim_{t\to\infty} \|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t)+\boldsymbol{\alpha}_i^r(t)\| = 0, \forall r \in \{1,\dots,N\}$, $\forall i \in \{1,\dots,n(r)\}$;
(ii) $\lim_{t\to\infty} \|\tilde{\phi}_r(\mathbf{p}_r(\tau))\| = 0, \forall r \in \{1,\dots,N\}, \forall \tau \,|\, w_r(\tau) > 0$;
(iii) $\lim_{t\to\infty}(\hat{\mathbf{a}}_r(t) - \hat{\mathbf{a}}_{r'}(t)) = 0, \forall r, r' \in \{1,\dots,N\}$.

Proof. We define a Lyapunov-like function $\mathcal{V}_2$ based on the robots' paths and sensory function estimates, and use Barbalat's lemma to prove asymptotic convergence of the system to a locally optimal equilibrium. Let $\mathcal{V}_2$ be defined as

$$\mathcal{V}_2 = \mathcal{H} + \frac{1}{2}\sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \tilde{\mathbf{a}}_r. \tag{24}$$

Taking the time derivative of $\mathcal{V}_2$, we obtain

$$\begin{aligned}\dot{\mathcal{V}}_2 &= \sum_{r=1}^{N}\sum_{i=1}^{n(r)} \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i^r}^{\mathrm{T}} \dot{\mathbf{p}}_i^r + \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r \\ &= \sum_{r=1}^{N}\sum_{i=1}^{n(r)} -(M_i^r \mathbf{e}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r + \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r. \end{aligned} \tag{25}$$

From (14), (15), (16), it is easy to check that

$$\mathbf{L}_i^r = M_i^r \mathbf{C}_i^r = M_i^r \hat{\mathbf{C}}_i^r + \tilde{M}_i^r (\hat{\mathbf{C}}_i^r - \tilde{\mathbf{C}}_i^r).$$

Plugging this into (25),

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -(M_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r + (\tilde{\mathbf{L}}_i^r - \tilde{M}_i^r \hat{\mathbf{C}}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r \\ &+ \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r.\end{aligned}$$

Using (14), we have

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -(\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r + (\tilde{\mathbf{L}}_i^r - \tilde{M}_i^r \mathbf{p}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r \\ &+ \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r.\end{aligned}$$

Substituting the dynamics specified by (5) and control law specified by (17), we obtain

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r}(\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r) \\ &+ \sum_{r=1}^{N}\sum_{i=1}^{n(r)} (\tilde{\mathbf{L}}_i^r - \tilde{M}_i^r \mathbf{p}_i^r)^{\mathrm{T}} \dot{\mathbf{p}}_i^r + \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r.\end{aligned}$$

Using (14) and (15),

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r}(\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r) \\ &+ \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \sum_{i=1}^{n(r)} \int_{V_i^r} W_s \mathcal{K}(\mathbf{q}) (\mathbf{q} - \mathbf{p}_i^r)^{\mathrm{T}} \, d\mathbf{q}\, \dot{\mathbf{p}}_i^r \\ &+ \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Gamma^{-1} \dot{\hat{\mathbf{a}}}_r.\end{aligned}$$

Plugging in the adaptation law from (20), (21) and (22),

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r}(\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r) \\ &- \gamma \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} (\Lambda_r \hat{\mathbf{a}}_r - \boldsymbol{\lambda}_r) \\ &- \zeta \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \sum_{r'=1}^{N} l_{r,r'} (\hat{\mathbf{a}}_r - \hat{\mathbf{a}}_{r'}) \\ &- \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} I_{\mathrm{proj}_r} \dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}.\end{aligned} \tag{26}$$

Using (18) and (19), the second term in (26) becomes

$$\begin{aligned}&= -\gamma \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \Big[\int_0^t w_r(\tau)\, \mathcal{K}(\mathbf{p}_r(\tau))\, \mathcal{K}(\mathbf{p}_r(\tau))^{\mathrm{T}} \, d\tau \hat{\mathbf{a}}_r \\ &\quad - \int_0^t w_r(\tau)\, \mathcal{K}(\mathbf{p}_r(\tau))\, \phi(\mathbf{p}_r(\tau))\, d\tau \Big] \\ &= -\gamma \sum_{r=1}^{N} \int_0^t w_r(\tau)\, \tilde{\phi}_r(\mathbf{p}_r(\tau))^{\mathrm{T}} [\hat{\phi}_r(\mathbf{p}_r(\tau)) - \phi(\mathbf{p}_r(\tau))]\, d\tau \\ &= -\gamma \sum_{r=1}^{N} \int_0^t w_r(\tau)\, (\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2 \, d\tau\end{aligned}$$

Plugging this expression back into (26) we obtain

$$\begin{aligned}\dot{\mathcal{V}}_2 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r}(\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r) \\ &- \gamma \sum_{r=1}^{N} \int_0^t w_r(\tau)\, (\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2 \, d\tau \\ &- \zeta \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} \sum_{r'=1}^{N} l_{r,r'} (\hat{\mathbf{a}}_r - \hat{\mathbf{a}}_{r'}) \\ &- \sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} I_{\mathrm{proj}_r} \dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}.\end{aligned} \tag{27}$$

Let $\mathbf{1} = [1,\dots,1]^{\mathrm{T}}$. Then, from Schwager et al. (2009) we can represent the third term in (27) as $-\zeta \sum_{j=1}^{m} \tilde{\boldsymbol{\Omega}}_j^{\mathrm{T}} L(t)\, \hat{\boldsymbol{\Omega}}_j$, where $\boldsymbol{\Omega}_j = \mathbf{a}(j)\,\mathbf{1}$, $\hat{\boldsymbol{\Omega}}_j =$

$[\hat{\mathbf{a}}_1(j), \ldots, \hat{\mathbf{a}}_N(j)]^{\mathrm{T}}$ and $\tilde{\boldsymbol{\Omega}}_j = \hat{\boldsymbol{\Omega}}_j - \boldsymbol{\Omega}_j$. Furthermore, $L(t)$ is the weighted graph Laplacian of the system at time t and is defined element-wise by

$$L(r, r') = \begin{cases} -l_{r,r'}, & \text{for } r \neq r', \\ \sum_{r'=1}^{N} l_{r,r'}, & \text{for } r = r'. \end{cases} \tag{28}$$

This graph Laplacian is positive semidefinite and, since our network of robots is a connected one, we know that it has exactly one zero eigenvalue whose eigenvector is $\mathbf{1}$ (Schwager et al., 2009). Hence, $\mathbf{x}^{\mathrm{T}} L \mathbf{x} = 0$ only if $\mathbf{x} = v\mathbf{1}$, for some $v \in \mathbb{R}$. Consequently, $\boldsymbol{\Omega}_j^{\mathrm{T}} L = \mathbf{a}(j)\, \mathbf{1}^{\mathrm{T}} L = 0,\ \forall j$. Therefore,

$$-\zeta \sum_{j=1}^{m} \tilde{\boldsymbol{\Omega}}_j^{\mathrm{T}} L \hat{\boldsymbol{\Omega}}_j = -\zeta \sum_{j=1}^{m} \hat{\boldsymbol{\Omega}}_j^{\mathrm{T}} L \hat{\boldsymbol{\Omega}}_j,$$

and we have that

$$\begin{aligned} \dot{\mathcal{V}}_2 = & \sum_{r=1}^{N} \sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r} (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} K_i^r (\hat{M}_i^r \hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r) \\ & -\gamma \sum_{r=1}^{N} \int_0^t w_r(\tau) (\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2 \, d\tau \\ & -\zeta \sum_{j=1}^{m} \hat{\boldsymbol{\Omega}}_j^{\mathrm{T}} L \hat{\boldsymbol{\Omega}}_j \\ & -\sum_{r=1}^{N} \tilde{\mathbf{a}}_r^{\mathrm{T}} I_{\mathrm{proj}_r} \dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}. \end{aligned} \tag{29}$$

Denote the four terms in (29) as $\xi_1(t)$, $\xi_2(t)$, $\xi_3(t)$, and $\xi_4(t)$, so that $\dot{\mathcal{V}}_2(t) = \xi_1(t) + \xi_2(t) + \xi_3(t) + \xi_4(t)$. Note that $\xi_1(t) \leq 0$ since K_i^r is uniformly positive definite and $\hat{\beta}_i^r > 0$, $\xi_2(t) \leq 0$ since it is the negative integral of a squared quantity, and it was proven by Schwager et al. (2009) that $\xi_4(t) \leq 0$. In addition, we already know that $L(t) \geq 0,\ \forall t$. Therefore, $\xi_3(t) \leq 0$.

Now consider the time integral of each of these four terms, $\int_0^t \xi_k(\tau)\, d\tau$, $k = 1, 2, 4$. Since each of the terms is non-positive, $\int_0^t \xi_k(\tau)\, d\tau \leq 0,\ \forall k$, and since $\mathcal{V}_2 > 0$, each integral is lower bounded by $\int_0^t \xi_k(\tau)\, d\tau \geq -\mathcal{V}_0$, where $\mathcal{V}_0$ is the initial value of $\mathcal{V}_2$. Therefore, these integrals are lower bounded and non-increasing, and hence $\lim_{t\to\infty} \int_0^t \xi_k(\tau)\, d\tau$ exists and is finite for all k. Furthermore, it was shown by (Schwager et al., 2009, Lemma 1) that $\dot{\xi}_1(t)$ and $\dot{\xi}_2(t)$ are uniformly bounded, therefore $\xi_1(t)$ and $\xi_2(t)$ are uniformly continuous. Hence, by Barbalat's lemma, $\xi_1(t) \to 0$ and $\xi_2(t) \to 0$. This implies (i) and (ii). In addition, it was shown by (Schwager et al., 2009, Lemma 2) that $\dot{\xi}_3(t)$ is uniformly bounded where it exists, and it is not differentiable only in isolated points on $[0, \infty)$ (if the connected network is all-to-all, then it is uniformly bounded everywhere). Therefore $\xi_3(t)$ is uniformly continuous in time. Hence, by Barbalat's Lemma, $\xi_3(t) \to 0$, which implies that $\hat{\boldsymbol{\Omega}}_j^{\mathrm{T}} L \hat{\boldsymbol{\Omega}}_j \to 0,\ \forall j$. From the definition of the weighted graph Laplacian, this means that $\hat{\boldsymbol{\Omega}}_j \to \hat{\mathbf{a}}_{\mathrm{final}}(j)\, \mathbf{1},\ \forall j$, where $\hat{\mathbf{a}}_{\mathrm{final}} \in \mathbb{R}^m_{\geq 0}$ is the final common parameter estimate vector shared by all robots. This implies (iii). □

Remark 4.3. Property (i) from Theorem 4.2 implies that the paths will reach a locally optimal configuration for sensing, i.e. an IC path, where the waypoints reach a stable balance between being close to their neighbor waypoints and providing good sensing locations according to the estimated sensory functions $\hat{\phi}_r(\mathbf{q}), \forall r$.

Remark 4.4. Properties (ii) and (iii) from Theorem 4.2 together imply that $\tilde{\phi}_r(\mathbf{q}),\ \forall r$ will converge to zero for all points on any robot's trajectory while the weight $w_r(t)$ is positive, but not necessarily for all of the environment. This means that the robots will learn the true sensory function for all of the environment if the union of their trajectories is rich enough while their weights are positive. Therefore, we can design the initial waypoint locations such that, between all robots, most of the unknown environment is explored (see Figure 1a).

4.2. Data collection weight functions

The adaptation law mentioned above includes a data collection weight function $w_r(t)$ in the calculations of Λ_r and $\boldsymbol{\lambda}_r$. The restriction on this weight function is that it must maintain Λ_r and $\boldsymbol{\lambda}_r$ bounded in order for our stability proof to hold. Schwager et al. (2009) gave a few options for such a weight. One such option that is applicable to the problem in this paper is to use a decaying exponential weight of the form $w_r(t, \tau) = \exp\{-(t - \tau)\}$. This weight function introduces a forgetting factor into our adaptation, which might be desirable if the environment is changing very slowly.

Another clear option is to simply use a constant for a certain amount of time, and then set the weight to zero in order for Λ_r and $\boldsymbol{\lambda}_r$ to remain bounded. This essentially enables the robots to spend time sampling the environment sensory function and, after a finite amount of time, stop sampling. Their estimates of the sensory function will be based only on this finite amount of sampled points. Since we assume that the sensory function is time invariant, this option is a reasonable and simple one for the weight function. Hence, for our simulations and implementations we use this option and define our weight function as

$$w_r(t) = \begin{cases} w, & \text{if } t < \tau_{w_r}, \\ 0, & \text{otherwise}, \end{cases} \tag{30}$$

where w is a positive constant scalar, and τ_{w_r} is some positive time at which robot r stops sampling the environment's sensory function in order to maintain Λ_r and λ_r bounded.

Algorithm 2 IC path controller for unknown environments: layer 1 for robot r.

Require: $\phi(\mathbf{q})$ can be parametrized as in (7)
Require: $\mathbf{a} \geq 0$
Require: Waypoints move slow enough that the robot can reach them and travel its path
Require: The network of robots is connected.
Require: Robot r knows the waypoints that correspond to its own path, i.e. location of $\mathbf{p}_i^r$, $\forall i \in \{1, \ldots, n(r)\}$.
Initially robot r is moving towards $\mathbf{p}_i^r$
Initialize Λ_r and $\boldsymbol{\lambda}_r$ to zero
Initialize $\hat{\mathbf{a}}_r$ to some non-negative value
loop
  if robot r reached $\mathbf{p}_i^r$ **then**
    move towards $\mathbf{p}_{i+1}^r$
  else
    keep moving towards $\mathbf{p}_i^r$
  end if
  Make measurement $\phi(\mathbf{p}_r)$ at robot r's position $\mathbf{p}_r$
  Obtain $\hat{\mathbf{a}}_{r'}$, $\forall r'$ that can communicate to robot r
  Update $\hat{\mathbf{a}}_r$ according to (22)
  Update Λ_r and $\boldsymbol{\lambda}_r$ according to (18) and (19)
end loop

Algorithm 3 IC path controller for unknown environments: layer 2 for robot r's ith waypoint.

Require: The waypoint controller knows $\hat{\mathbf{a}}_r$ from Algorithm 2
Require: The waypoint controller can calculate its Voronoi partition
Require: Knowledge of its two neighboring waypoint positions $\mathbf{p}_{i-1}^r$ and $\mathbf{p}_{i+1}^r$
loop
  Compute the waypoint's Voronoi partition V_i^r
  Compute $\hat{\mathbf{C}}_i^r$ according to (12)
  Obtain neighbor waypoint locations $\mathbf{p}_{i-1}^r$ and $\mathbf{p}_{i+1}^r$
  Compute $\mathbf{u}_i^r$ according to (17)
  Update $\mathbf{p}_i^r$ according to (5)
end loop

4.3. Additional note on the adaptation law

As mentioned in Remark 4.4 the robots' estimates of the sensory function will converge to the real description for the locations that they visit while the weight function is positive, but not necessarily for other locations. Hence, in order for the robots to perform the sensing/coverage task in an acceptable way, it is a good idea to provide the robots with initially rich trajectories through the environment. In our simulations and experiments we provide such rich trajectories by giving the robots initial paths that pass through most of the environment ("zig-zagging" through the environment) and we allow the robots to travel their initial paths once with the IC path controller, but with $K_i^r = 0$, $\forall i, r$. This allows the robots to obtain enough samples of $\phi(\mathbf{q})$ in order to ensure that their estimates will converge to the correct values. If, in contrast, the robots are allowed to move the waypoints before building a proper estimate of $\phi(\mathbf{q})$, then the paths could evolve in a way that never allows the robots to obtain a rich enough trajectory to estimate the sensory function.

This initial pass by the robots through their paths without editing them and using the adaptation law to make estimates of the sensory function is referred to as the *learning phase*. Once the robots obtain enough samples of the sensory function in order to make proper estimates of it, the robots stop taking samples in order to maintain Λ_r and $\boldsymbol{\lambda}_r$ bounded. According to (30), this translates to τ_{w_r}, $\forall r$ is the time when the learning phase ends. In our simulations and experiments we end the learning phase when all robots have gone once through their initial paths. At this point, the robots turn up the controller gains K_i^r and start what we refer to as the *path shaping phase*, where the waypoints' locations are edited to minimize the cost function.

The reader should note that this is purely an implementation detail and it is not, by any means, a requirement of the adaptive controller that we propose in this paper. For example, the separation of these two phases is not necessary if we initialize the parameter estimates $\hat{\mathbf{a}}_r$ to a relatively high non-zero value, for all robots. By doing so, the robots initially think that all of the environment is a region of interest, so initially they will want to travel through all of the environment and will shape their paths to do so. Only after traveling through all of the environment (while learning the true sensory function through our adaptation law) will the robots shape their paths to only cover the true regions of interest. We have simulated this scenario, and the results are practically identical to the phase separation technique. In our simulations and experiments, we will use the phase separation technique, simply because learning the sensory function while all waypoints are not allowed to move (i.e. the learning phase) is performed faster than when the waypoints move simultaneously (i.e. no phase separation).

In any case, our adaptation law allows the robots to perform estimates of the sensory function online in a decentralized way with asymptotic convergence to the true sensory function guaranteed for all robots if the union of their trajectories is rich enough. This online adaptation law can especially be beneficial in some scenarios where the sensory function varies slowly in time. For such cases, a decaying exponential weight function for data collection (as specified in Section 4.2) can allow the robots to change their estimates as the sensory function changes. However, this case of time-varying sensory functions is outside the scope of this paper and will not be discussed further.

4.4. Decentralized algorithm

The IC path controller for unknown environments can be implemented in a decentralized way using the architecture shown Figure 5, which is divided into two layers.

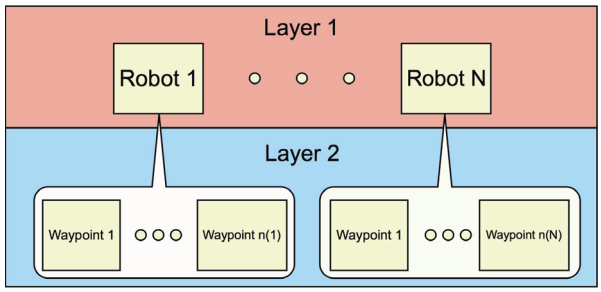
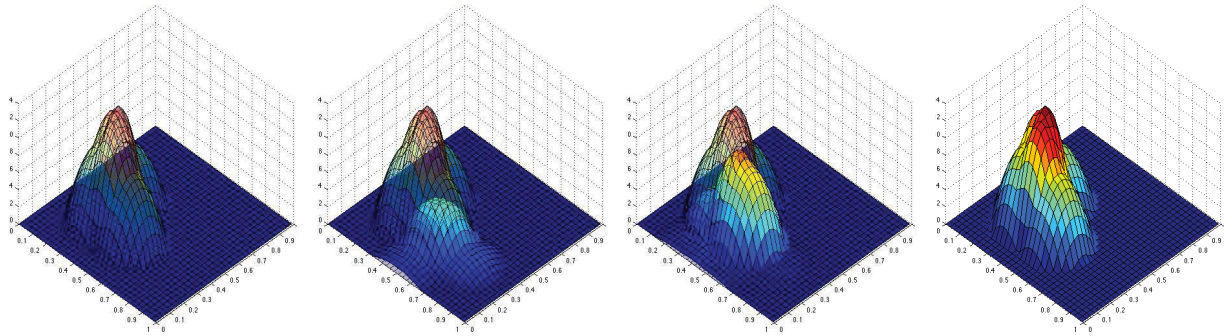


Fig. 5. Architecture for the multi-robot system.



(a) Learning time step: 1 (b) Learning time step: 45 (c) Learning time step: 49 (d) Learning time step: 180

Fig. 6. Simulated environment during the learning phase. The translucent data represents the true sensory function $\phi(\mathbf{q})$ which is unknown to the robot. The solid colored data represents the robot's sensory function estimate $\hat{\phi}_r(\mathbf{q})$. As time passes, the robot builds the correct estimate of the sensory function.

The first layer (top of Figure 5) corresponds to the robots traveling through their paths, and sampling and estimating the environment sensory function. A practical algorithm for this first layer is shown in Algorithm 2 and can be executed in a decentralized way by each robot, in the sense that each robot can execute this algorithm independently, only sharing its current parameter estimate vector to the other robots that are within communication range (i.e. $l_{r,r'} > 0$).

The second layer (bottom of Figure 5) corresponds to the waypoints being moved and placed in locally optimal locations to create IC paths. For this layer, we interpret each waypoint as having its own control thread that runs Algorithm 3. This algorithm can be implemented in a decentralized way in the sense that each waypoint control thread can execute this algorithm independently, only sharing information with their neighboring waypoint control threads and enough information for all waypoint control threads to compute their Voronoi partitions. If the robots have multiple cores, or multiple processors, the computations for each waypoint control thread can be readily parallelized for improved efficiency.[4] Alternatively, if the robots do not have the computational power to run in parallel a separate control thread for each of their waypoints, another decentralized implementation can be done if each robot individually performs its adaptation law (Algorithm 2) and the waypoint control law (Algorithm 3) in a sequential manner for all of its corresponding waypoints.

4.5. Simulated results for a single robot

In this section we present a simulation performed in MATLAB for the single-robot case (i.e. $N = 1$, $r = 1$) in an unknown environment. The robot has $n(r) = 58$ waypoints. The environment's sensory function for this simulation is identical to that described in Section 3.1. The estimated parameter vector $\hat{\mathbf{a}}_r$, and Λ_r and $\boldsymbol{\lambda}_r$ are initialized to zero. The parameters for the controller are $W_n = 5$, $W_s = 150$, $\Gamma =$ identity, $\gamma = 2000$, and $w = 30$.

The initial path was designed to "zig-zag" across the environment (see Figure 9a) to provide a rich initial trajectory for the robot to sample the environment. We present our results in the two phases described in Section 4.3: learning phase and path shaping phase. The learning phase corresponds to the robot traveling through its full path once, without reshaping it, in order to obtain enough samples to properly estimate $\phi(\mathbf{q})$. The path shaping phase corresponds to when the waypoints are moved to reshape the path into an IC path, and starts when the learning phase is done.

4.5.1. Learning phase During this phase $K_i^r = 0$, $\forall i$, so the path does not change. The robot travels its entire path once, measuring the environment sensory function $\phi(\mathbf{q})$ as it travels and using the adaptation law from (22) to generate its estimate $\hat{\phi}_r(\mathbf{q})$. This process can be seen in Figure 6. As the robot travels its path, the adaptation law causes $\tilde{\phi}_r(\mathbf{q}) \to 0$, $\forall \mathbf{q} \in Q$, as can be seen from Figure 6 where the robot's estimate $\hat{\phi}_r(\mathbf{q})$ (solid-colored data) converges to the

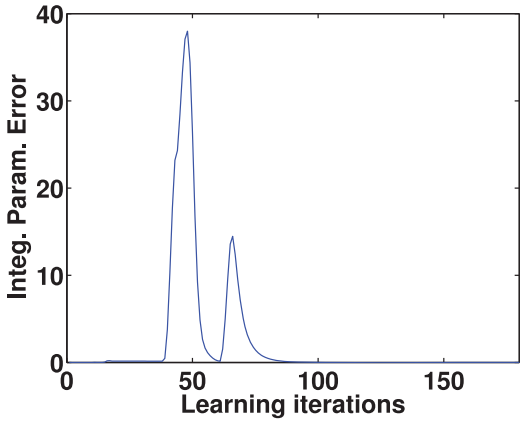


Fig. 7. Integral parameter error.

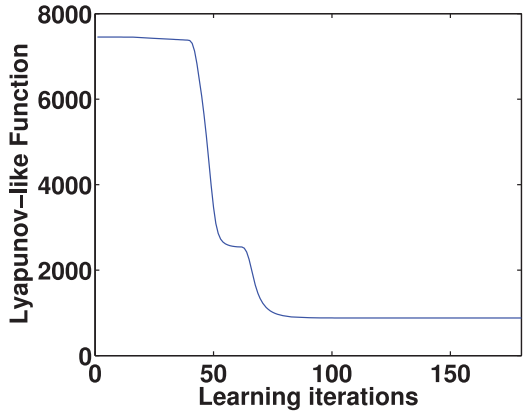


Fig. 8. Lyapunov-like function.

real environment's sensory function $\phi(\mathbf{q})$ (translucent data) for all of the space. This means that the robot's trajectory was rich enough to generate accurate estimates of the environment's sensory function. Numerically, this is shown in Figure 7, where $\lim_{t\to\infty} \int_0^t w(\tau)\left(\tilde{\phi}_r(\mathbf{p}_r(\tau))\right)^2 d\tau = 0$, in accordance with Theorem 4.2(ii). Finally, for this learning phase, we see in Figure 8 that the Lyapunov-like function $\mathcal{V}_2$ is monotonically non-increasing, which supports our theory.

4.5.2. Path shaping phase Once the robot travels through its path once (sampling and building an estimate $\hat{\phi}_r$), the waypoints' control gains are set to $K_i^r = 50$, $\forall i$. We can see how the path evolves under the IC path controller in Figure 9. We can see that, since the robot built a proper estimate of $\phi(\mathbf{q})$ in the learning phase, the path shapes itself properly to cover the region of interest. In addition, since the environment is identical to that in Section 3.1, the resulting IC path should be similar to that in Section 3.1 (assuming that the initial path is similar). Comparing Figures 3f and 9f we can see that this is true. Note that, since there are more waypoints and the controller gain is smaller, the system takes more time to converge to the final configuration in the latter case.

Figure 10 shows the true and estimated mean waypoint position errors, where the estimated error refers to the quantity $\|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t) + \boldsymbol{\alpha}_i^r(t)\|$. As shown, $\lim_{t\to\infty} \|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t) + \boldsymbol{\alpha}_i^r(t)\| = 0$, in accordance with Theorem 4.2(i). Figure 11 shows the Lyapunov-like function $\mathcal{V}_2$ monotonically non-increasing during this path shaping phase and settling to a local minimum, meaning that the path is driven to a locally optimal configuration according to (24). The initial value of this function in the path shaping phase is the final value of the function in the learning phase.

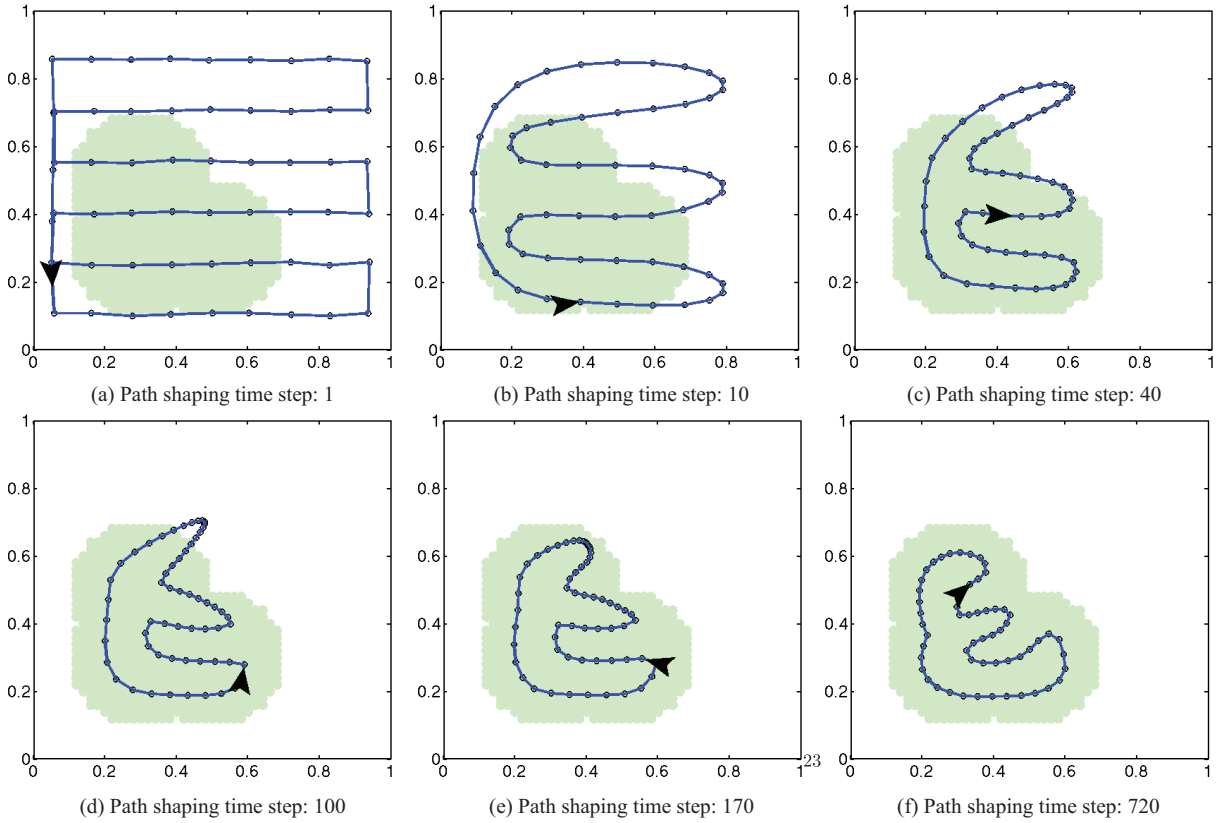
4.6. Simulated results for multiple robots

In this section we present a simulation of a 10-robot system in an unknown environment using our IC path controller. This simulations, like all others, was done in a quasi-decentralized way, in which the decentralized controllers shown in Algorithms 2 and 3 were used, but in a single laptop computer. We present a case for $N = 10$ robots and $n(r) = 28$ waypoints, $\forall r$. The region Q is taken to be the unit square, and the basis function $\mathcal{K}$ is defined in (10) with 25 truncated Gaussians, $\sigma = 0.4$ and $\rho_{\text{trunc}} = 0.2$. The parameter vector $\mathbf{a}$ is defined as $\mathbf{a}(j) = 60$, $\forall j$. The environment's sensory function $\phi(\mathbf{q})$ created with these parameters can be seen in Figure 12d. Note that $\phi(\mathbf{q})$ is approximately a uniform constant positive value, for all $\mathbf{q}$.

The parameter vector $\hat{\mathbf{a}}_r$, Λ_r and $\boldsymbol{\lambda}_r$, for all r are initialized to zero. The controller parameters are $\Gamma = \text{identity}$, $\gamma = 2000$, $W_n = 70$, $W_s = 500$, and $w = 10$. All robots were assumed to maintain communication, hence $l_{r,r'}(t)\,\zeta = 10$, $\forall r, r'$, $\forall t$.

The initial paths can be seen in Figure 17a, where all robots have practically the same "zig-zagging" initial path across the environment, providing rich initial trajectories for them to sample the environment.[5] We present results in two separate phases: (1) learning phase; and (2) path shaping phase.

4.6.1. Learning phase During this phase $K_i^r = 0$, $\forall i, r$, so the paths do not change. The robots travel their entire paths once, sampling $\phi(\mathbf{q})$ as they travel and using the adaptation law from (22) to estimate it. This process is seen in Figure 12 for one of the 10 robots. We can see from Figure 13 that the mean over all robots of $\int_0^t w_r(\tau)\,(\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2\, d\tau$ converges to zero, in accordance with Theorem 4.2(ii). Also, Figure 14 shows the consensus error converging to zero in accordance with Theorem 4.2(iii). The combination of these two results is reflected in Figure 15 (or, equivalently, Figure 12), where $\tilde{\phi}_r(\mathbf{q}) \to 0$, $\forall \mathbf{q} \in Q$ for one of the robots. Since the consensus error converges to zero then we can conclude that the adaptation laws cause $\tilde{\phi}_r(\mathbf{q}) \to 0$, $\forall \mathbf{q} \in Q$, $\forall r$. This means that the union of all of the robots' trajectories was rich enough to generate accurate estimates for all of the environment's sensory function. Finally, for this learning phase, we see in Figure 16 that the Lyapunov-like function $\mathcal{V}_2$ is monotonically non-increasing.



(a) Path shaping time step: 1 (b) Path shaping time step: 10 (c) Path shaping time step: 40

(d) Path shaping time step: 100 (e) Path shaping time step: 170 (f) Path shaping time step: 720

Fig. 9. Simulated single-robot IC path controller for an unknown environment. The path is shown as a blue line connecting all waypoints, shown as black circles. The black arrowhead is the simulated robot's position. The region of interest is shown as a light-green-colored region. As time passes, the robot's path converges to a configuration that only focuses on the region of interest.

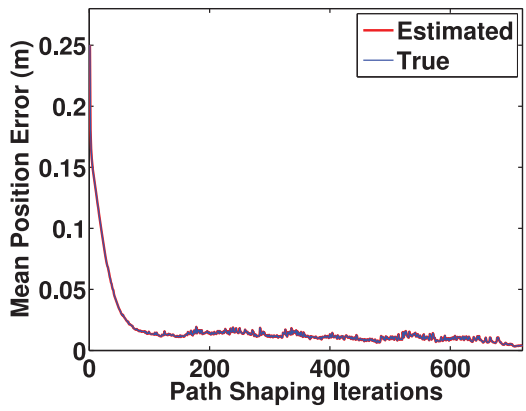


Fig. 10. Mean waypoint position error.

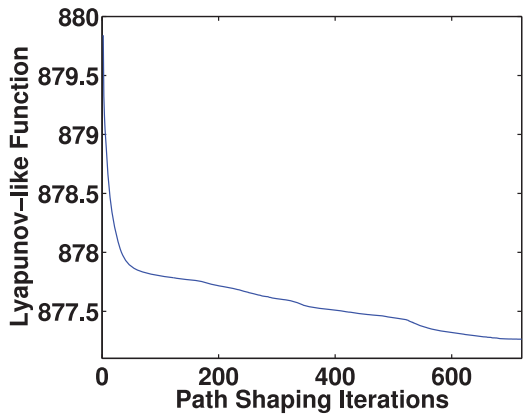
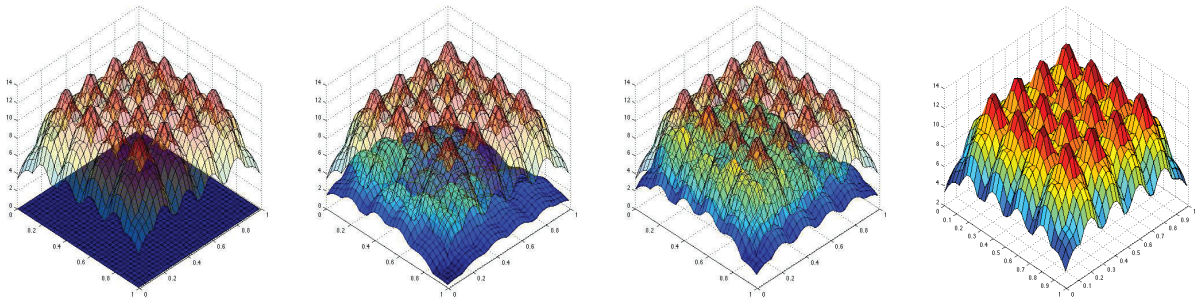


Fig. 11. Lyapunov-like function.

4.6.2. Path shaping phase After the robots travel through their paths once, sampling $\phi(\mathbf{q})$, the waypoint control gains are set to $K_i^r = 90$, $\forall i, r$. We can see how the paths evolve under the action of the IC path controller in Figure 17. Performance-wise, we see in Figure 18 that the quantity $\|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t) + \boldsymbol{\alpha}_i^r(t)\|$ converges to zero, $\forall i, r$ in accordance with Theorem 4.2(i). Finally, Figure 19 shows the Lyapunov-like function $\mathcal{V}_2$ monotonically non-increasing and approaching a limit during this phase.

Note that the sensory function $\phi(\mathbf{q})$ generates a region of interest that is equal to the whole environment, as seen by the light-green-colored region in Figure 17. Furthermore, the environment is approximately uniform in sensory information, i.e. every point in the environment has approximately the same sensory information. Owing to this, the union of all robots' paths covers all of the environment evenly (see Figure 17f).



(a) Learning time step: 1 (b) Learning time step: 15 (c) Learning time step: 22 (d) Learning time step: 137

Fig. 12. Simulated environment during the learning phase. The translucent data represents the true sensory function $\phi(\mathbf{q})$ which is unknown to the robot. The solid colored data represents the robot's sensory function estimate $\hat{\phi}_r(\mathbf{q})$ for one of the 10 robots. As time passes, the robot builds the correct estimate of the sensory function. Since all robots converge to the same estimate, due to the consensus term in the adaptation law, all robots converge to the same correct estimate of the real sensory function.

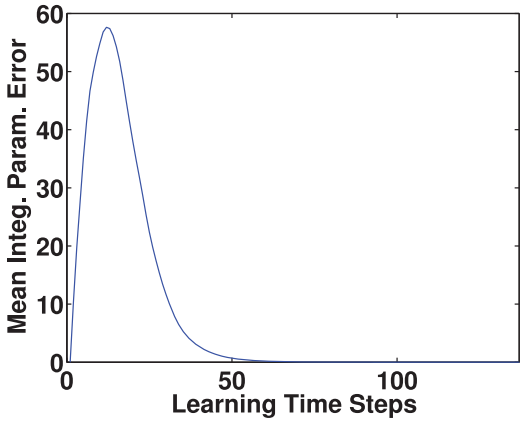


Fig. 13. Integral parameter error.

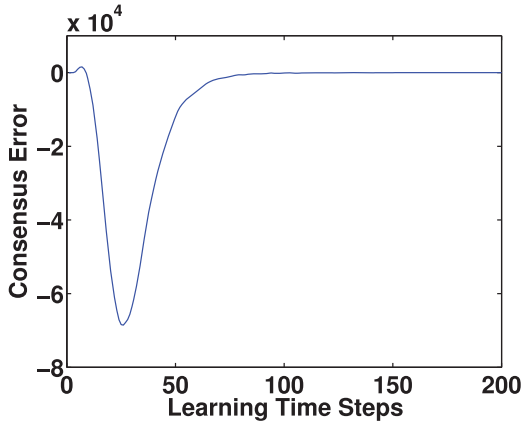


Fig. 14. Consensus error.

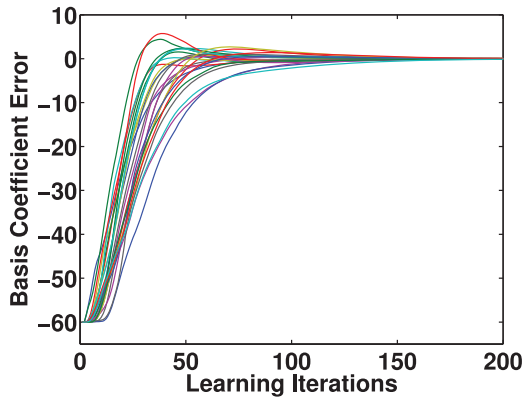


Fig. 15. Basis coefficient error.

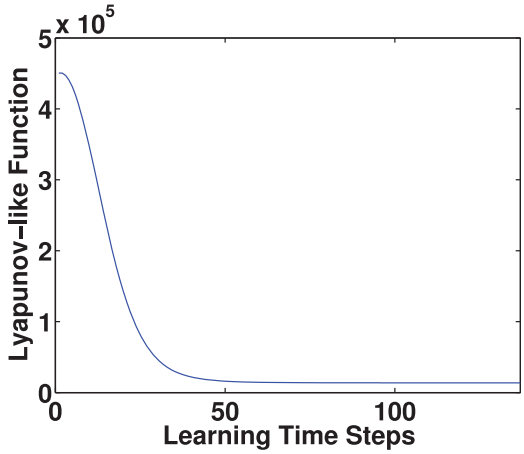


Fig. 16. Lyapunov-like function.

4.7. Discussion

The IC path controller drives the robots to work together to sense/cover the regions of interest in an unknown environment. In the case seen in Figure 17, since the region of interest is the entire environment, we see that the robots' paths spread out throughout the entire environment to cover it uniformly. This sense of teamwork seen in the robots and their paths comes as a direct consequence of the coupling that exists between them in the Voronoi decomposition. Owing to this, a robot's path will not tend to go over the same spot that another robot's path is already passing through because they have different Voronoi partitions. However, it is important to point out that the IC paths generated by our controller depend on the parameters W_n and W_s inserted

into the controller and the initial paths assigned to each robot. We now compare some cases where such variations produce big differences in the final IC paths.

4.7.1. Variations of neighbor distance weight W_n Due to the coupling of waypoints within each path, if the value of the neighbor distance weight W_n (or, equivalently, the ratio W_n/W_s) is high enough the paths will tend to shape in a way that creates short paths for each robot. We now present two cases with the same controller parameters, with the exception of the value of W_n. The "high W_n" case corresponds to $W_n = 30$ and the "low W_n" case corresponds to $W_n = 3$. All other parameters are the same for both cases: $W_s = 150$, $N = 2$, $n(r) = 40\ \forall r$, $\mathbf{a}(j) = 80$, for $j \in \{1, 2, 3, 16, 17, 18\}$, and $\mathbf{a}(j) = 0$ for all other j, $\sigma = 0.4$, $\rho_{\text{trunc}} = 0.2$, $K_i^r = 90\ \forall i, r$, $\Gamma =$ identity, $\gamma = 2000$, and $w = 30$, and $l_{r,r'}(t)\,\zeta = 20,\ \forall r, r',\ \forall t$.

The regions of interest created by these parameters are shown as light-green-colored regions in Figures 20 and 21. The initial paths for both cases are practically the same and are shown in Figure 20. The final paths, however, are very different for each case and are shown in Figure 21. The difference in final paths is mainly caused by the difference in the neighbor distance weight W_n. For both cases the IC path controller generates paths that allow the robots to sense the regions of interest. However, for the high W_n case, such a high gain provides highly attractive forces between neighboring waypoints. This causes the paths to tend to be short because the neighbor waypoints want to be close to each other. Therefore, the system evolves in a way that the environment is covered with low-length paths. In contrast, in the low W_n case, there are low attractive forces for neighboring waypoints. This causes the waypoints to be more distant to each other and focus more on the coverage/sensing task, rather than generating short paths. Depending on the application, the weights W_n and W_s can be selected to generate appropriate results. As an example, if collision is an issue, one way to possibly avoid collisions is to have a high W_n so that paths tend to not intersect each other.

4.7.2. Variations of initial paths For all of the controllers that have been shown in previous sections, any initial paths for the robots can be selected. However, depending on the initial paths, the system may achieve a different set of IC paths corresponding to a different local minimum of the cost function. In this section, we provide an example to show how the initial paths can affect the local equilibrium at which the system settles to.

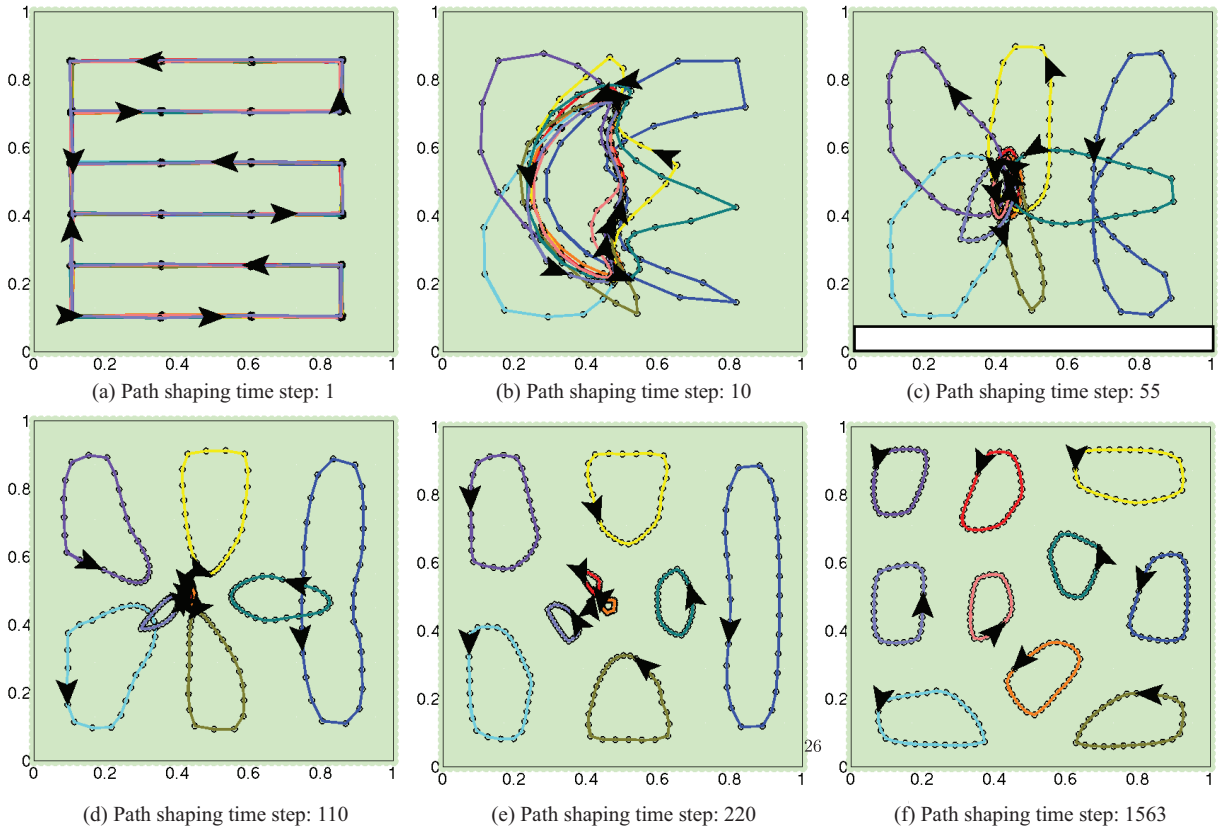
The example consists of two cases of a system with the same environment, parameters and number of robots, but with different initial paths for the robots. The parameters are: $W_s = 50$, $W_n = 3$, $N = 2$, $\mathbf{a}(j) = 25$, for $j \in \{7, 9, 11, 15, 16, 20\}$, and $\mathbf{a}(j) = 0$ for all other j, $\sigma = 0.18$, $\rho_{\text{trunc}} = 0.15$, $K_i^r = 90\ \forall i, r$, $\Gamma =$ identity, $\gamma = 500$, and $w = 3$, and $l_{r,r'}(t)\,\zeta = 20,\ \forall r, r',\ \forall t$.

The regions of interest created by these parameters are shown as light-green-colored regions in Figures 22 and 23. The initial paths are the only difference between the two cases and these are shown in Figure 22. Consequently, the resulting final paths are very different for each case and are shown in Figure 23. More accurately, the difference in final paths is caused by a different local minimum in the cost function. From this example, it is clear that the initial paths for the robots can greatly affect the outcome of the final path shapes. Therefore, if some structure of the environment is known, it can be used to design initial paths that are more beneficial to the outcome of the system.

5. Controlling the speed of the robots along the IC paths

The IC path controller only affects the locations of waypoints defining the robots' paths. The user has the liberty of controlling the speed at which the robots travel the IC paths in order to achieve a particular task. As an example of the usefulness of exploiting this freedom, we consider the problem of *persistent monitoring*. In persistent monitoring we wish for the robots, each assumed to have a finite sensor footprint, to gather information in a dynamic environment so as to guarantee a bound on the difference between the robots' current models of the environment and the actual state of the environment for all time and over all locations. Since their sensors have finite footprints, the robots cannot collect the data about all of the environment at once. Consequently, as data about a dynamic region becomes outdated, the robots must return to that region repeatedly to collect new data. More generally, if different parts of the environment change at different rates, the robots must visit these areas in proportion to their rates of change to ensure a bounded uncertainty in the estimation. The algorithm presented by Smith et al. (2011) calculates the speed of the robots at each point along a given path in order to perform a persistent monitoring task, i.e. to prevent the robots' model of the environment from becoming too outdated. In this section, we use the IC path controller, along with the speed controller from Smith et al. (2011) to produce IC trajectories online with provable performance guarantees for persistent monitoring in unknown environments.

More specifically, Smith et al. (2011) defined the persistent monitoring problem as an optimization problem whose goal is to keep a time changing environment, modeled as an *accumulation function*, bounded everywhere. The accumulation function grows where it is not covered by any of the robots' sensors, indicating a growing need to collect data at that location, and shrinks where it is covered by any of the robots' sensors, indicating a decreasing need for data collection. This accumulation function can be thought of as the amount of dust in a cleaning task, as the difference between the robots' mental model and reality in a mapping or estimation task, or as the chance of crime in a surveillance



(a) Path shaping time step: 1 (b) Path shaping time step: 10 (c) Path shaping time step: 55

(d) Path shaping time step: 110 (e) Path shaping time step: 220 (f) Path shaping time step: 1563

Fig. 17. Simulated 10-robot system using the IC path controller. Each robot follows a path with a unique color. All robots have approximately the same initial path, as seen in (a). The black arrowheads represent the simulated robots. The region of interest, shown as the light-green-colored region, is all of the environment since $\phi(\mathbf{q})$ is non-zero throughout the environment.

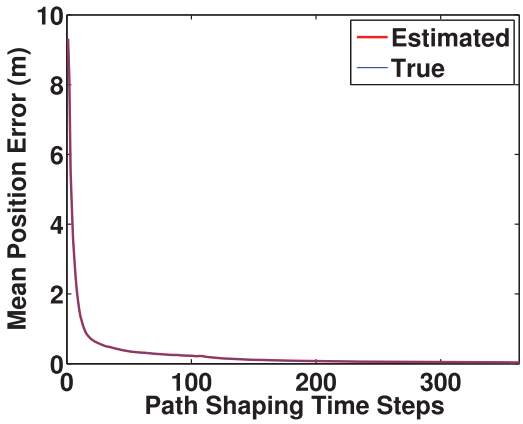


Fig. 18. Mean waypoint position error.

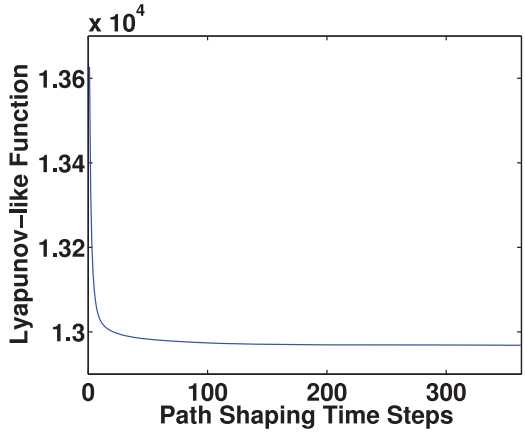


Fig. 19. Lyapunov-like function.

task. In order to apply our IC path algorithm to the persistent monitoring problem, we must treat our environment's sensory function $\phi(\mathbf{q})$ as a rate of growth for the accumulation function in the environment. The goal of a persistent monitoring task is to maintain this accumulation function bounded for the entire environment and for all time.

We assume that each robot is equipped with a sensor with a finite footprint $F_r(\mathbf{p}_r) = \{\mathbf{q} \in Q : \|\mathbf{q} - \mathbf{p}_r\| \leq \rho\}$ when robot r is at location $\mathbf{p}_r$, where ρ is a constant positive scalar.[6] This sensor can correspond, for example, to a camera in a surveillance or monitoring task, or a vacuum cleaner in a cleaning task.

Following the approach from Soltero et al. (2012), the IC path controller for persistent monitoring should incorporate the stability criterion for a persistent monitoring task given the speeds of each robot at each point along their paths, referred to as the *speed profiles*, such that the control action increases the *stability margin* of the persistent monitoring task through time. Since the robots do not know the

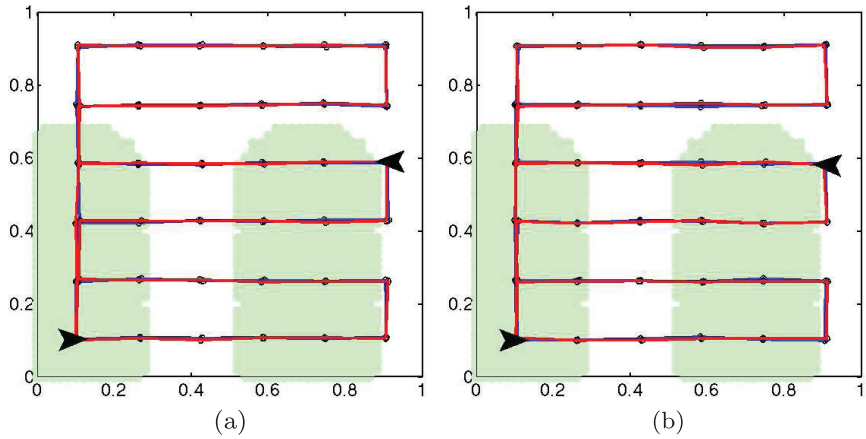


Fig. 20. Initial paths for comparing neighbor distance weight: (a) high W_n; (b) low W_n.

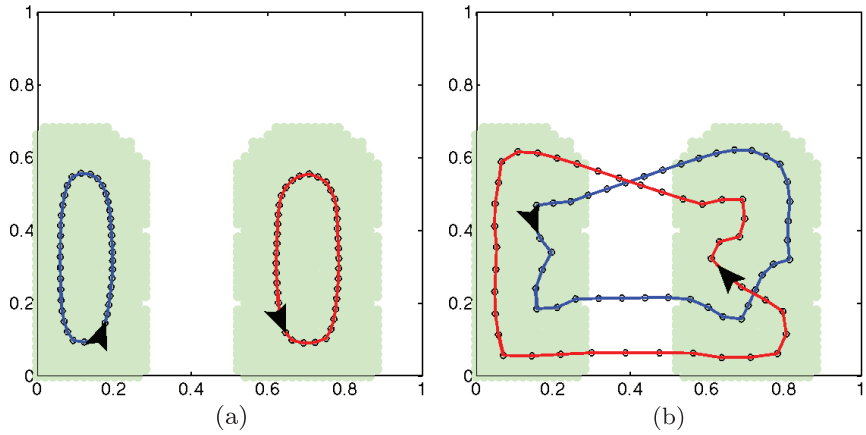


Fig. 21. Final paths for comparing neighbor distance weight: (a) high W_n; (b) low W_n.

accumulation function's rate of growth, but estimate it, each robot r's IC path controller will work towards improving the estimated stability criterion of the persistent monitoring task, defined by Smith et al. (2011) as

$$\hat{s}_r(\mathbf{q},t) = \hat{\phi}_r(\mathbf{q},t) - \sum_{r'=1}^{N} \frac{\tau_c^{r'}(\mathbf{q},t)}{T_{r'}(t)} c_{r'}(\mathbf{q}) < 0, \quad \forall \mathbf{q} \mid \hat{\phi}_r(\mathbf{q},t) > 0, \tag{31}$$

where $\hat{\phi}_r(\mathbf{q},t)$ (the estimated sensory function at time t) is robot r's estimated rate at which the environment's accumulation function grows at point $\mathbf{q}$ (referred to as the *growth rate*), the constant scalar $c_r(\mathbf{q})$ is the *consumption rate* at which the accumulation function shrinks when robot r's sensor is covering point $\mathbf{q}$, $T_r(t)$ is the time it takes robot r to complete its path at time t, and $\tau_c^r(\mathbf{q},t)$ is the time robot r's sensor covers point $\mathbf{q}$ along the path at time t. These two last quantities are calculated with the speed profiles. Robot r's estimated stability margin at time t is defined as $\hat{S}_r(t) = -(\max_{\mathbf{q}} \hat{s}_r(\mathbf{q},t))$, and a *stable persistent task* is one in which S (the true version of $\hat{S}_r$) is positive, which means the robots are able to maintain the environment's accumulation function bounded at all points $\mathbf{q}$. Note that only estimated regions of interest, i.e. points $\mathbf{q}$ that satisfy $\hat{\phi}_r(\mathbf{q},t) > 0$, are considered in (31) since it is not necessary to persistently sense a region that has no sensory interest.

Smith et al. (2011) gave a linear program which calculates the speed profile for each robot's path at time t that maximizes $\hat{S}_r(t)$ (or $S(t)$ for ground-truth). With the speed profiles obtained with this linear program, and using (31), we can formulate an IC path controller for persistent monitoring that drives both the robots' paths and speed profiles in a direction to, not only perform stable persistent monitoring tasks, but locally optimize these tasks. Hence, from this point on, we assume the maximizing speed profile for $\hat{S}_r(t)$ is known by each robot r and used to obtain $\hat{s}_r(\mathbf{q},t)$, $\forall \mathbf{q}$, $\forall t$.

In order to incorporate the persistent monitoring stability criterion into our IC path controller, we must re-define the waypoints dynamics as

$$\dot{\mathbf{p}}_i^r = I_i^r \mathbf{u}_i^r, \tag{32}$$

where $\mathbf{u}_i^r$ is defined in (17),

$$I_i^r = \begin{cases} 0, & \text{if } \frac{\partial \hat{S}_r}{\partial \mathbf{p}_i^r}^{\mathrm{T}} \mathbf{u}_i^r < 0 \text{ and } t - t_u^{r,i} > \tau_{\text{dwell}}, \\ 1, & \text{otherwise}, \end{cases} \tag{33}$$

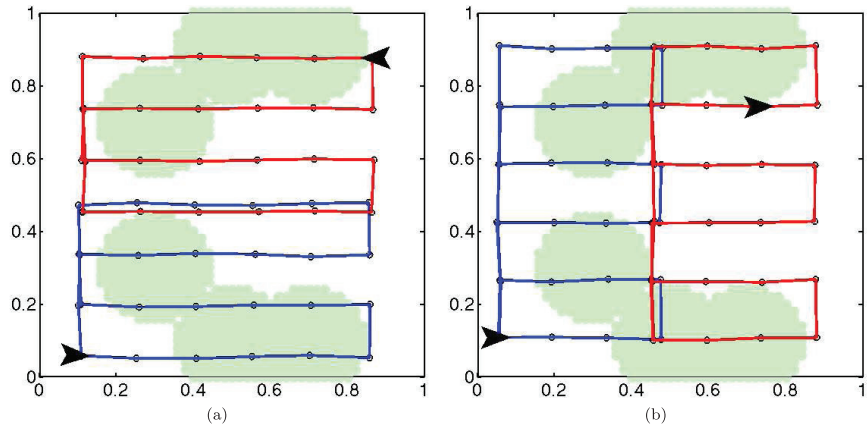


Fig. 22. Initial paths for comparing initial path variations: (a) case 1; (b) case 2.

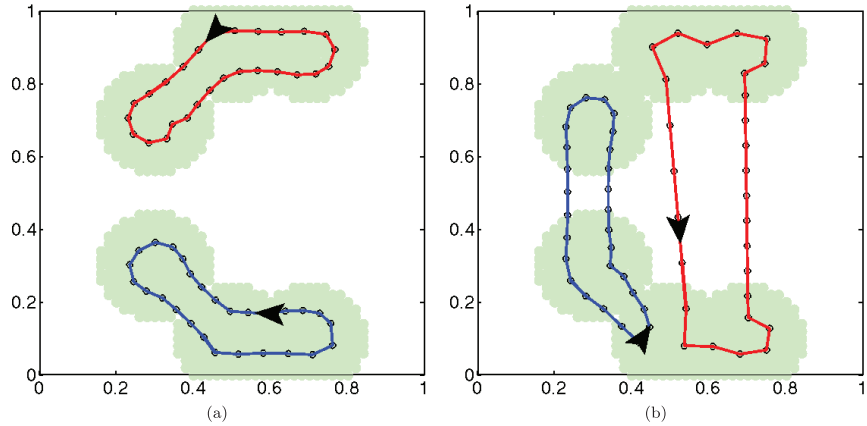


Fig. 23. Final paths for comparing initial path variations: (a) case 1; (b) case 2.

τ_{dwell} is a design parameter, and $t_u^{r,i}$ is the most recent time at which I_i^r, switched from zero to one (switched "up"). The formulation of this controller ensures that the estimated stability margin $\hat{S}_r(t)$ is non-decreasing for all r.

Remark 5.1. The quantity $\frac{\partial \hat{S}_r}{\partial \mathbf{p}_i^r}(t)$ refers to $\frac{\partial \hat{S}_r}{\partial \mathbf{p}_i^r}(t+\epsilon)$, for positive $\epsilon \to 0$, when $\arg\max_{\mathbf{q}} \hat{s}_r(\mathbf{q}, t-\epsilon) \neq \arg\max_{\mathbf{q}} \hat{s}_r(\mathbf{q}, t+\epsilon)$ (Soltero et al., 2012).

Remark 5.2. It is very important to note that I_i^r is a Boolean function that is used to shut off the control action on the waypoints if their movement is not beneficial to the task that is desired. The definition of I_i^r provided in (33) is for purposes of persistent monitoring. Different Boolean functions can be used depending on the task.

We now prove that the system under the new IC controller for persistent monitoring is stable.

Theorem 5.3 (Convergence theorem for persistent monitoring in unknown environments using the IC path controller)**.** *Under Assumptions 2.5 and 4.1, with waypoint dynamics specified by (32), control law specified by (17), and adaptive law specified by (22), we have:*

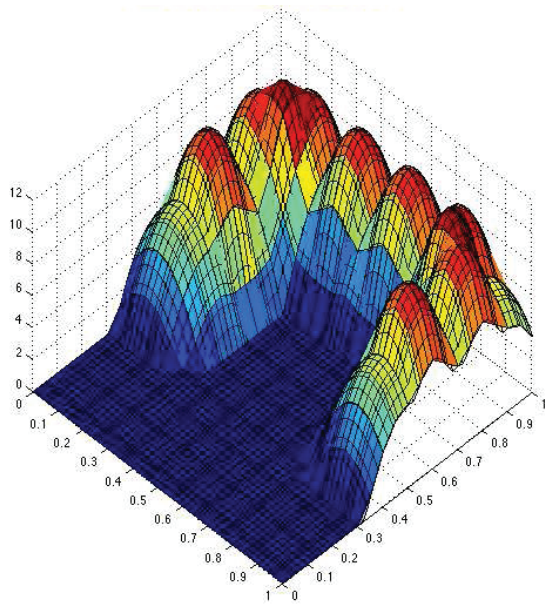


Fig. 24. Sensory function.

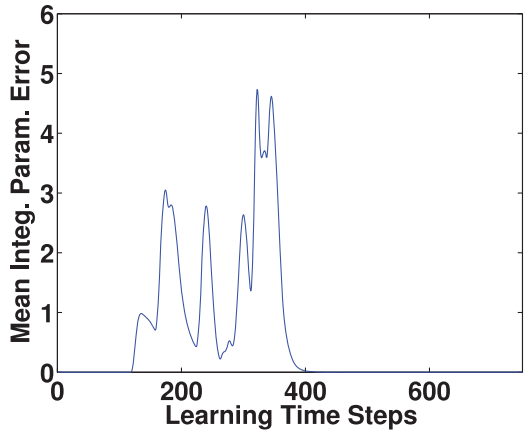


Fig. 25. Integral parameter error.

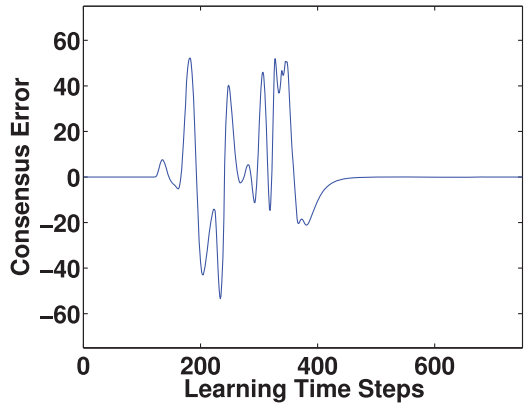


Fig. 26. Consensus error.

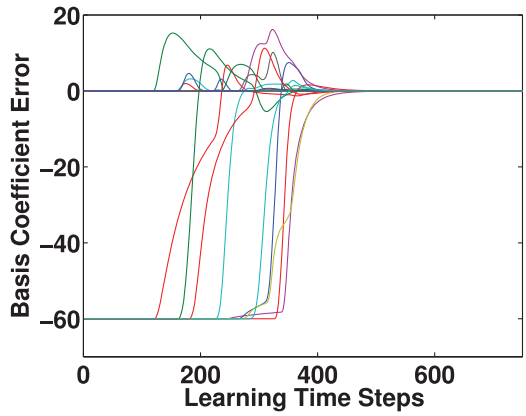


Fig. 27. Basis coefficient error.

(i) $\lim_{t\to\infty} I_i^r(t)\,\|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t)+\boldsymbol{\alpha}_i^r(t)\,\| = 0,$
$\forall r \in \{1,\ldots,N\}, \forall i \in \{1,\ldots,n(r)\};$
(ii) $\lim_{t\to\infty} \|\tilde{\phi}_r(\mathbf{p}_r(\tau))\| = 0,$
$\forall r \in \{1,\ldots,N\},\ \forall \tau \mid w_r(\tau) > 0;$
(iii)$\lim_{t\to\infty}(\hat{\mathbf{a}}_r(t) - \hat{\mathbf{a}}_{r'}(t)) = 0, \forall r, r' \in \{1,\ldots,N\}.$

Proof. Let $\mathcal{V}_3$ be a new Lyapunov-like function, and let $\mathcal{V}_3 = \mathcal{V}_2$ from (24). Then, following the procedure from Section 4.1, but with $\dot{\mathbf{p}}_i^r$ defined by (32), we obtain

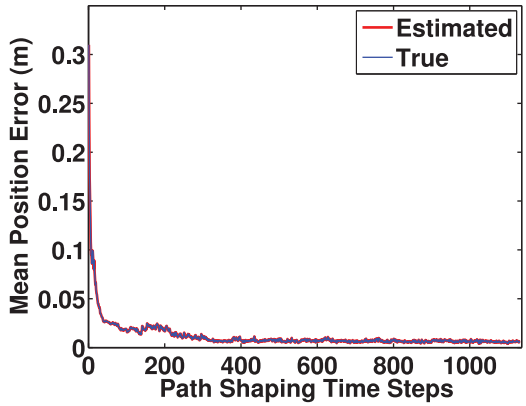


Fig. 28. Mean waypoint position error.

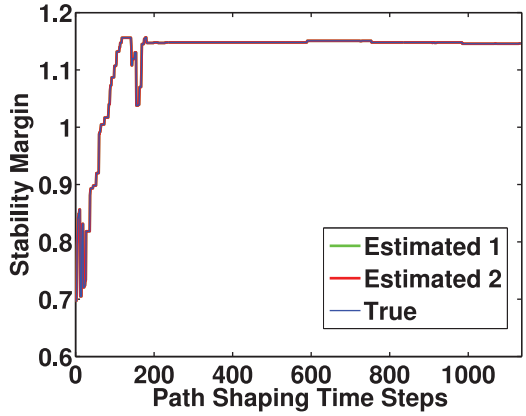


Fig. 29. Stability margin for persistent monitoring task.

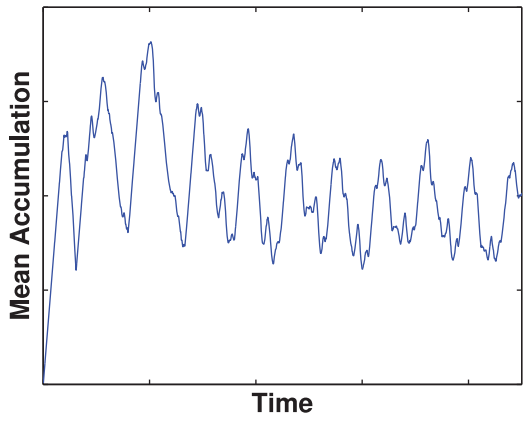


Fig. 30. Mean accumulation function value.

$$\begin{aligned}\dot{\mathcal{V}}_3 = &\sum_{r=1}^{N}\sum_{i=1}^{n(r)} -\frac{1}{\hat{\beta}_i^r}(\hat{M}_i^r\hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)^{\mathrm{T}} I_i^r K_i^r (\hat{M}_i^r\hat{\mathbf{e}}_i^r + \boldsymbol{\alpha}_i^r)\\ &-\gamma\sum_{r=1}^{N}\int_0^t w_r(\tau)(\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2\,d\tau\\ &-\zeta\sum_{j=1}^{m}\hat{\boldsymbol{\Omega}}_j^{\mathrm{T}} L\hat{\boldsymbol{\Omega}}_j\\ &-\sum_{r=1}^{N}\tilde{\mathbf{a}}_r^{\mathrm{T}} I_{\mathrm{proj}_r}\dot{\hat{\mathbf{a}}}_{\mathrm{pre}_r}.\end{aligned} \tag{34}$$

From Section 4.1, we know that second and third term in (34) converge to zero, implying (ii) and (iii), respectively, and it was shown by Soltero et al. (2012) that the first term converges to zero, implying (i). □

Remark 5.4. The stability margin can theoretically worsen while I_i^r, for some i, r, cannot switch from one to zero because it is waiting for $t - t_u^{r,i} > \tau_{\text{dwell}}$. However, τ_{dwell} can be selected arbitrarily small and, in practice, any computer will enforce a τ_{dwell} due to discrete time steps. Therefore, it is not a practical restriction. As a result, intuitively, Theorem 5.3(i) means that $\lim_{t\to\infty} \|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t) + \boldsymbol{\alpha}_i^r(t)\| = 0$ only if this helps the persistent monitoring task. Otherwise $\lim_{t\to\infty} I_i^r(t) = 0$, meaning that the persistent task will not benefit if the ith waypoint in robot r's path moves.

5.1. Implementation and results

We have simulated the IC controller for persistent monitoring by multiple robots in a MATLAB environment for many test cases, and we performed a physical implementation of a system comprised of two quadrotors. In this section, we present such implementation.

The implementation was performed in a quasi-decentralized way, meaning that although the algorithm is a distributed one, the computations for all waypoints and all robots were executed serially in a single computer. Hence, some of the communication procedures between distributed nodes were bypassed at the expense of a longer running time since the computations were performed serially. We used a collision avoidance algorithm for persistent monitoring from Soltero et al. (2011) in order to prevent the quadrotors from colliding.

We present a case for $N = 2$ robots and $n(r) = 44$ waypoints, $\forall r$. A fixed-time step numerical solver is used with a time step of 0.01 seconds and $\tau_{\text{dwell}} = 0.009$. The region Q is taken to be the unit square. The sensory function $\phi(\mathbf{q})$ is parametrized as a Gaussian network as in Section 3.1, with $\mathcal{K}$ defined in (10), $\sigma = 0.4$ and $\rho_{\text{trunc}} = 0.2$. The parameter vector $\mathbf{a}$ is defined as $\mathbf{a}(j) = 60$, for $j \in \{3, 4, 5, 10, 15, 20, 23, 24, 25\}$, and $\mathbf{a}(j) = 0$ otherwise. The environment's sensory function (growth rates) created with these parameters can be seen in Figure 24.

The parameters $\hat{\mathbf{a}}_r$, Λ_r and $\boldsymbol{\lambda}_r$, for all r are initialized to zero. The controller parameters are $\Gamma = \text{identity}$, $\gamma = 3000$, $W_n = 6$, $W_s = 150$, $w = 3$ and $\rho = 0.12$. The two robots were assumed to maintain communication, hence $l_{r,r'}(t)\,\zeta = 20$, $\forall r, r'$, $\forall t$. The environment is discretized into a 10×10 grid and only points in this grid with $\hat{\phi}_r(\mathbf{q}) > 0$ are used as points of interest in (31). These points are a subset of the regions of interest, and are shown as green dots in Figure 31. This discretization is only used in (31) and, by using this discretized version of the environment, the running time for experiments is greatly reduced. For more sensitive systems, this grid can be refined.

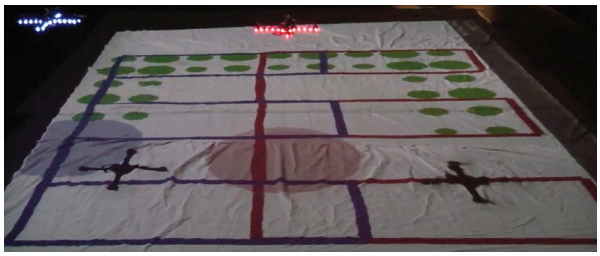
Following the rules of a persistent monitoring task, the environment's accumulation function grows at a growth rate $\phi(\mathbf{q})$ at point $\mathbf{q}$ and is consumed by robot r at a consumption rate of $c_r(\mathbf{q}) = 10$, if $\mathbf{q} \in F_r(\mathbf{p}_r(t))$. This accumulation function is represented by the green dots in Figure 31, where their sizes represent the values of the accumulation function at those points of interest. Again, the goal of a persistent monitoring task is to maintain the sizes of these green dots bounded. The environment and, therefore, the sensor measurements are simulated, but projected onto the floor for visualization, as seen in Figure 31.

The initial paths can be seen in Figure 31a, where both robots have a "zig-zagging" initial path across a portion of the environment, and, between both robots, most of the environment is initially traversed, providing rich initial trajectories for them to sample the environment. We present results in two separate phases: (1) learning phase; and (2) path shaping phase.

5.1.1. Learning phase In this phase $K_i^r = 0$, $\forall i, r$. The robots travel their entire paths once, sampling the environment growth rates as they travel and using the adaptation law from (22) to estimate these rates. We can see in Figure 25 that the mean over both robots of $\int_0^t w_r(\tau)\,(\tilde{\phi}_r(\mathbf{p}_r(\tau)))^2\,d\tau$ converges to zero, in accordance with Theorem 5.3(ii). Also, Figure 26 shows the consensus error converging to zero in accordance with Theorem 5.3(iii). The combination of these two results is reflected in Figure 27, where $\tilde{\phi}_r(\mathbf{q}) \to 0$, $\forall \mathbf{q} \in Q$ for one of the robots. Since the consensus error converges to zero then we can conclude that the adaptation laws cause $\tilde{\phi}_r(\mathbf{q}) \to 0$, $\forall \mathbf{q} \in Q$, $\forall r$.

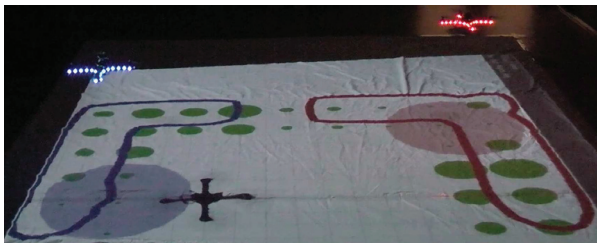
5.1.2. Path shaping phase In this phase $K_i^r = 90$, $\forall i, r$. We can see how the paths evolve under the action of this controller in Figure 31, which shows snapshots of the multi-robot implementation at different time steps. Figure 28 shows quantity $I_i^r(t)\,\|\hat{M}_i^r(t)\,\hat{\mathbf{e}}_i^r(t) + \boldsymbol{\alpha}_i^r(t)\|$ converging to zero, $\forall i, r$ in accordance with Theorem 5.3(i). Figure 29 shows the persistent monitoring task's stability margin increasing through time as the robots' paths become IC paths for persistent monitoring. The chattering in the stability margin is due to the discretization of the system, and can be reduced by reducing the length of the time steps. Finally, Figure 30 shows the mean over all points of interest of the environment's accumulation function value over time. As shown, this value initially increases on average due to the initialization of the system. Later, it starts to decrease and reaches an approximate steady-state behavior that is bounded from above. This, along with the stability margin from Figure 29, corresponds to the locally optimal final configuration of the system for persistent monitoring.

The hardware implementation was run more than 10 times, generating IC paths that are practically identical to their simulated counterparts. Since the calculations for this implementation were performed in serial, the paths reshaped much more slowly than they would in the case of a fully decentralized implementation. This experiment



(a) Time step: 0

(b) Time step: 50



(c) Time step: 375

Fig. 31. Hardware implementation of IC path controller for persistent monitoring with two quadrotors. Three snapshots of the path shaping phase at different time step values are shown. The paths, shown as the blue and red lines, connect all the waypoints corresponding to each robot. The green dots are a sample of the environment's regions of interest and are used in the calculation of the stability margin for the persistent monitoring task. The size of a green dot represents the value of the accumulation function at that point. The robots are the blue-lit and red-lit quadrotors, and the sensor footprints for the persistent monitoring task are represented by the colored circles under the robots' positions.

required approximately 3 hours to achieve a final IC path. If the implementation was performed in a fully decentralized way (each waypoint having its own control thread running in parallel), then the running time would be greatly reduced. For all cases that were studied (with or without persistent monitoring), theoretically,[7] it takes less than 20 seconds for the paths to converge to the final IC paths.

6. Limitations and future work

Although the IC path controller has very positive features such as simplicity and decentralized nature of the controller, it does have some limitations that must be mentioned. First, as has been mentioned previously, the environment's sensory function $\phi(\mathbf{q})$ must be time-invariant if the environment is unknown. The reason is that the adaptation law in the IC path controller can only ensure proper estimates of the environment's sensory function for the points that the robots have sampled throughout their travels. If the sensory function is time-varying and changes in a way that the robots are not able to sense this change (for example, if it changes somewhere away from the IC paths), then the adaptation law will not account for these changes and the controller will not respond. If, however, we make the assumption that fluctuations in the sensory function only happen sufficiently near the robots' IC paths, then the controller will respond appropriately if we use the forgetting factor in the data collection weight function, as described in Section 4.2.

Unlike typical search/iterative path planning methods where you can optimize a cost function encoding specific constraints, our IC path controller cannot directly encode constraints in the robots' paths, such as length of the paths, curvature limitations, etc. Our controller can indirectly encode some of these constraints by properly selecting the parameters W_s and W_n, but there is no specific rule for determining the optimal values for such parameters. As with most problems in engineering, the user has to decide if the power of a traditional search/iterative path planner outweighs the simplicity and decentralized nature of our controller.

The IC path controllers presented here do not account for collisions when operating with multiple robots mainly because they have no information about the robots' trajectories; they only deal with generating paths. It is only when controlling the velocity of the robots that enough information is acquired to perform collision avoidance. In this paper, when we consider the case of controlling the robot's speed, we use a collision avoidance algorithm developed in previous work, along with some heuristics that dealt with the time-varying nature of the paths. A possible future direction for this work could be to better integrate a collision avoidance behavior in the robots under some assumptions on the robots' speeds along the generated paths. For example, shaping the paths not only as a function of the sensory function, but also as a function of the robots' trajectories along such paths.

The IC path controllers assume that a fixed number of waypoints are assigned to each robot. However, one could imagine that dynamically changing the number of these waypoints could improve the performance and rate of convergence in the system. The theory behind this scenario has yet to be explored and developed.

Vehicle dynamics are not considered in this paper. Hence, a vehicle with non-holonomic constraints might not be able to travel the paths generated by the IC path controller. Future work could address this issue and add constraints between neighboring waypoints that encode vehicle dynamics.

Finally, future work could focus on characterizing the sensitivity of the system under our controller to initial conditions and controller parameters, particularly to initial paths and the weight W_n. We have observed that when initial waypoint locations are very close to each other for multiple robots, the system can be very sensitive, and extremely small perturbations in the initial paths can lead to very different final paths. This is due to the sensitivity in the Voronoi partitions when generator points are too close to each other.

7. Conclusion

This paper presents a novel gradient-based path planning algorithm based on a dynamical systems approach. This approach deviates from the standard search/iterative methods that are typically used to treat the path planning problem. By doing so, we generated a very simple controller that can be implemented in a decentralized way where each waypoint in a robot's path has its own control thread. The main difficulty of implementing this algorithm lies in the lower-level difficulty of calculating the Voronoi partitions. The proposed controller is guaranteed to reach a locally optimal solution, unlike other search/iterative methods that can only provide an approximate solution.

The proposed controller works by introducing two primitives to the waypoints' motions. The first primitive drives the waypoints towards their Voronoi centroids, which encodes in the robots being able to cover/sense the regions of interest. The second primitive drives the waypoints to stay close to their two neighboring waypoints, i.e. the previous and next waypoints in the respective paths. This encodes in the waypoints forming closed paths. The strength of the two primitives can be customized by properly selecting the parameters W_s and W_n. The combination of the two primitives drives the waypoints to locations that correspond to locally optimal paths according to a Voronoi-based coverage cost function.

We developed the theory for the general multi-robot case. We first derived and showed results for the case where the environment is known. For this case, we saw that our assumption that the environment's sensory function $\phi(\mathbf{q})$ is time-invariant can be relaxed, since the controller responds to time-varying sensory functions. Later, we treated the unknown environment case, in which an adaptation law was used to estimate the environment's sensory function. The estimates generated from this adaptation law were proven to converge to the real sensory function for all points in any of the robot's trajectory, while the data collection weight w_r was positive.

The IC path controllers provide a strong tool for robots to generate useful paths in unknown environments for any sensing task such as cleaning and surveillance. Our controller has the useful feature that the speed of a robot's travel along its path is customizable to the specific task that the user wants achieved. As an example of this usefulness, we combined the IC path controller with a speed controller designed for persistent monitoring tasks. We proved the stability of the system under this combined controller and physically implemented it in a two-robot system tasked with performing a persistent monitoring task in an unknown environment. The controller was able to generate IC paths that were specifically useful for persistent monitoring since they locally optimized the stability criterion for the task.

Funding

This work was supported in part by ONR (MURI award N00014-09-1-1051), an NSF Graduate Research Fellowship (award 0645960) and The Boeing Company.

Notes

1. How a robot moves between two waypoints can be defined arbitrary by the user and can potentially be constrained by the robot's dynamics. We use a straight line interpolation due to its simplicity.
2. We assume that under the influence of the control input, the waypoints do not outrun the robot, i.e. that the waypoints move slow enough for the robot to reach them and travel its path.
3. In our simulation and experiments we create a sensory function $\phi(\mathbf{q})$ by selecting a parameter vector $\mathbf{a}$. In a real-world scenario this is not possible, i.e. the sensory function $\phi(\mathbf{q})$ is inherent to the environment and we simply assume that it can be parametrized by a parameter vector $\mathbf{a}$ and a basis function vector $\mathcal{K}$.
4. The robots may also be able to massively parallelize this computation using a GPU, for example.
5. All robots having the same initial path is not necessary for the robots' estimates of $\phi(\mathbf{q})$ to converge. What is necessary is that the union of their trajectories is rich enough.
6. Any footprint shape can be used, and the footprint size does not have to be the same for all robots. For simplicity, we use a circular footprint with same size for all robots.
7. In this theoretical case, each waypoint has a control thread running in parallel with the other waypoints' and the robots' adaptation law threads, and all threads have a loop period of 0.01 seconds.

References

Binney J, Krause A and Sukhatme GS (2010) Informative path planning for an autonomous underwater vehicle. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4791–4796.

Cortes J, Martinez S, Karatas T and Bullo F (2004) Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation* 20(2): 243–255.

Cunningham CT and Roberts RS (2001) An adaptive path planning algorithm for cooperating unmanned air vehicles. In: *2001 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, pp. 3981–3986.

Drezner Z (1995) Facility location: A survey of applications and methods. In: *Springer Series in Operations Research*. New York: Springer-Verlag.

Graham R and Cortes J (2012) Adaptive information collection by robotic sensor networks for spatial estimation. *IEEE Transactions on Automatic Control* 57(6): 1404–1419.

Hollinger G and Singh S (2010) Multi-robot coordination with periodic connectivity. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4457–4462.

Ioannou PA and Sun J (1996) *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall.

Kavraki LE, Svestka P, Latombe JC and Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: *1985 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 500–505.

Kyriakopoulos KJ and Saridis GN (1988) Minimum jerk path generation. In: *1988 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 364–369.

Large EW, Christensen HI and Bajcsy R (1999) Scaling the dynamic approach to path planning and control: competition among behavioral constraints. *The International Journal of Robotics Research* 18(1): 37–58.

Levine D, Luders B and How JP (2010) Information-rich path planning with general constraints using rapidly-exploring random trees. In: *AIAA Infotech@Aerospace Conference*, Atlanta, GA, paper AIAA-2010-3360.

Meliou A, Krause A, Guestrin C and Hellerstein JM (2007) Nonmyopic informative path planning in spatio-temporal models. In: *Proceedings of the 22nd National Conference on Artificial Intelligence*, *AAAI'07*, vol. 1, pp. 602–607.

Piazzi A, Guarino Lo Bianco C and Romano M (2007) η^3-splines for the smooth path generation of wheeled mobile robots. *IEEE Transactions on Robotics* 23(5): 1089–1095.

Pimenta LCA, Schwager M, Lindsey Q, et al. (2008) Simultaneous coverage and tracking (SCAT) of moving targets with robot networks. In: *Proceedings of the Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR 08)*.

Qin YQ, Sun DB, Li N and Cen YG (2004) Path planning for mobile robot using the particle swarm optimization with mutation operator. In: *2004 International Conference on Machine Learning and Cybernetics*, vol. 4. pp, 2473–2478.

Salapaka S, Khalak A and Dahleh MA (2003) Constraints on locational optimization problems. In: *2003 IEEE Conference on Decision and Control*, vol. 2, pp. 1741–1746.

Schwager M, Rus D and Slotine JJ (2009) Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research* 28(3): 357–375.

Singh A, Krause A, Guestrin C and Kaiser WJ (2009) Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research* 34(1): 707–755.

Smith SL, Schwager M and Rus D (2011) Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Transactions on Robotics* 99: 1–17.

Soltero DE, Schwager M and Rus D (2012) Generating informative paths for persistent sensing in unknown environments. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2172–2179.

Soltero DE, Smith SL and Rus D (2011) Collision avoidance for persistent monitoring in multi-robot systems with intersecting trajectories. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3645–3652.

Stentz A (1994) Optimal and efficient path planning for partially-known environments. In: *1994 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, pp. 3310–3317.