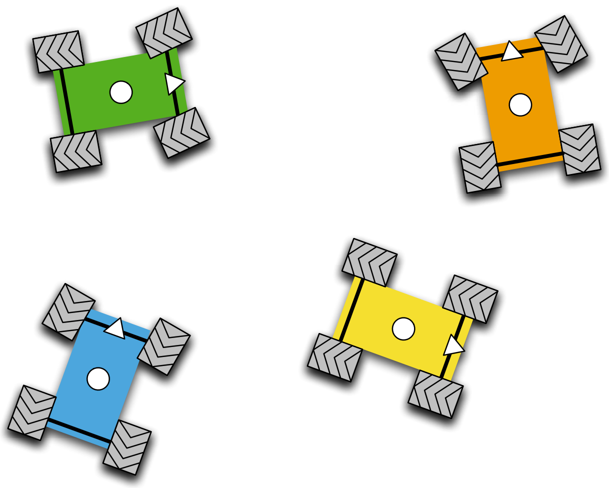
Distributed Control of Robotic Networks

A Mathematical Approach to Motion Coordination Algorithms

Chapter 6: Boundary estimation and tracking

Francesco Bullo
Jorge Cortés
Sonia Martínez

May 20, 2009



PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Contents

Chapter Six

Boundary estimation and tracking

The aim of this chapter is to provide an example of a motion coordination algorithm that can be used in a specific sensing task. This is the task of detection and estimation of an evolving boundary in two dimensions by a robotic sensor network. This type of operation can be of interest in the validation of oceanographic and atmospheric models, as well as for the demarcation of hazardous environments. In the ocean, a boundary can delimit areas where there are abrupt changes in temperature, which can influence the marine biodiversity in those areas. In the atmosphere, a boundary can establish the front of a highly polluting expanding substance. The containment of a spreading fire is another situation that can translate into the specific task of boundary estimation and tracking.

Under full knowledge and centralized computation, various methods exist in the literature to solve the boundary estimation task. A first challenge that we face when designing coordination algorithms for robotic networks is the determination of the extent to which these tasks can be performed in a distributed way and under limited information. In this regard, the algorithm presented in this chapter to track environmental boundaries is distributed, in the sense that it does not require the use of a central station or "fusion center." Our algorithm builds on basic notions from interpolation theory and employs distributed linear iterations and consensus algorithms. The algorithm can be seen as part of a general effort to investigate distributed filters for estimation tasks.

A second challenge is posed by sudden events that may occur when performing sensing tasks, such as the detection of an intruder or an abrupt change in the concentration of some chemical. These events require a specific action on the part of the network. Since the timing of such events is not known *a priori*, this requires coordination algorithms that specify event-driven, asynchronous responses of the robotic network. We deal with this issue by building on the robotic network model proposed in Chapter 3. Our exposition here on boundary estimation relies on Susca (2007) and Susca et al. (2008).

The chapter is organized as follows. The first section extends the synchronous model proposed in Chapter 3 to include the event-driven, asynchronous operation of a robotic network. The second section reviews some basic facts on interpolation theory for boundaries. In the third section, we introduce the ESTIMATE UPDATE AND BALANCING LAW to solve the boundary estimation task and analyzes its correctness. We end the chapter with three sections on, respectively, bibliographic notes, proofs of the results presented in the chapter, and exercises. Throughout the exposition, we make extensive use of polygonal approximations, geometric decompositions, and consensus algorithms. The convergence analysis is based on the LaSalle Invariance Principle and on distributed linear iterations.

6.1 EVENT-DRIVEN ASYNCHRONOUS ROBOTIC NETWORKS

In what follows, we model "event-driven asynchronous" robotic networks. This model describes groups of agents that reset their processor states upon certain asynchronous events, that is, events that do not necessarily happen simultaneously for all agents. For example, a relevant event might be a robot reaching a location or leaving a region. The following event-driven model is convenient to describe our algorithm for boundary estimation, but its applicability extends beyond this particular scenario. Following our discussion of synchronous robotic networks in Section 3.1, the event-driven model consists of the following ingredients: a robotic network, as in Definition 3.2, and an event-driven control and communication law, as defined next.

Definition 6.1 (Event-driven control and communication law). An *event-driven control and communication law* $\mathcal{ECC}$ for a robotic network $\mathcal{S}$ consists of the following sets:

(i) $\mathbb{A}$, a set containing the `null` element, called the *communication alphabet*—elements of $\mathbb{A}$ are called *messages*;

(ii) $W^{[i]}$, $i \in I$, called the *processor state sets*; and

(iii) $W_0^{[i]} \subseteq W^{[i]}$, $i \in I$, sets of *allowable initial values*;

and the following maps:

(i) $(\text{msg-trig}^{[i]}, \text{msg-gen}^{[i]}, \text{msg-rec}^{[i]})$, $i \in I$, called the *message-trigger function*, *message-generation function*, and *message-reception function*, respectively, such that

  (a) $\text{msg-trig}^{[i]} : X^{[i]} \times W^{[i]} \to \{\texttt{true}, \texttt{false}\}$,

  (b) $\text{msg-gen}^{[i]} : X^{[i]} \times W^{[i]} \times I \to \mathbb{A}$,

  (c) $\text{msg-rec}^{[i]} : X^{[i]} \times W^{[i]} \times \mathbb{A} \times I \to W^{[i]}$;

(ii) $(\text{stf-trig}_k^{[i]}, \text{stf}_k^{[i]})$, $i \in I$, $k \in \{1, \dots, K_{\text{stf}}^{[i]}\}$, called the k^{th} *state-transition trigger function* and the k^{th} *(processor) state-transition function*, respectively, such that

   (a) $\text{stf-trig}_k^{[i]} : X^{[i]} \times W^{[i]} \to \{\texttt{true}, \texttt{false}\}$,

   (b) $\text{stf}_k^{[i]} : X^{[i]} \times W^{[i]} \to W^{[i]}$; and

(iii) $\text{ctl}^{[i]} : X^{[i]} \times W^{[i]} \to U^{[i]}$, $i \in I$, called *(motion) control functions.*

If the network $\mathcal{S}$ is uniform and all sets and maps of the law $\mathcal{ECC}$ are independent of the identifier, that is, for all $i \in I$ and $k \in \{1, \dots, K_{\text{stf}} = K_{\text{stf}}^{[i]}\}$,

$$W^{[i]} = W, \quad (\text{stf-trig}_k^{[i]}, \text{stf}_k^{[i]}) = (\text{stf-trig}_k, \text{stf}_k), \text{ctl}^{[i]} = \text{ctl},$$
$$(\text{msg-trig}^{[i]}, \text{msg-gen}^{[i]}, \text{msg-rec}^{[i]}) = (\text{msg-trig}, \text{msg-gen}, \text{msg-rec}),$$

then $\mathcal{ECC}$ is said to be *uniform* and is described by the tuple

$$(\mathbb{A}, W, \{W_0^{[i]}\}_{i \in I}, (\text{msg-trig}, \text{msg-gen}, \text{msg-rec}), \{\text{stf-trig}_k, \text{stf}_k\}_{k=1}^{K_{\text{stf}}}, \text{ctl}). \quad \bullet$$

Observe that a key difference between Definitions 3.9 and 6.1 is that the message-generation and state-transition functions are substituted by sets of maps $(\text{msg-trig}^{[i]}, \text{msg-gen}^{[i]}, \text{msg-rec}^{[i]})$ and $(\text{stf-trig}_k^{[i]}, \text{stf}_k^{[i]})$. A second difference is that the control function depends only upon the current robot position, and not the position at last sample time.

The event-driven control and communication law models situations in which the agent physical and processor states need to satisfy certain constraints before a message should be sent. For example, in each triplet $(\text{msg-trig}^{[i]}, \text{msg-gen}^{[i]}, \text{msg-rec}^{[i]})$, the map $\text{msg-trig}^{[i]}$ acts as a trigger for agent i to send a message to its neighbors, the map $\text{msg-gen}^{[i]}$ computes the message to be sent, and $\text{msg-rec}^{[i]}$ specifies how agent i updates its processor state when receiving a message. In hybrid systems terminology (van der Schaft and Schumacher, 2000), the map $\text{msg-trig}^{[i]}$ can be seen as a guard map. Similarly, in the pair $(\text{stf-trig}_k^{[i]}, \text{stf}_k^{[i]})$, the map $\text{stf-trig}_k^{[i]}$ acts as a trigger for agent i to update its processor state. If several $\text{stf-trig}_k^{[i]}$ are satisfied at the same time, then the agent can freely choose among the corresponding state transition functions to update the processor state. This freedom means that our dynamical system is described by a set-valued map and leads to non-deterministic evolutions.

The evolution of a robotic network dictated by an event-driven control and communication law is asynchronous: there is no common time schedule for all robots to send messages, receive messages, and update their processor states. Only when an event happens, an agent sends a message or updates its state. The asynchronous event-driven evolution can be loosely described in the following way:

(i) Starting from the initial conditions, the physical state of each agent evolves in continuous time according to the control function $\mathrm{ctl}^{[i]}$.

(ii) At every instant of time $t_1 \in \mathbb{R}_{\geq 0}$ such that the message-trigger function for agent i satisfies $\text{msg-trig}^{[i]}(x^{[i]}(t_1), w^{[i]}(t_1)) = \texttt{true}$, agent i generates a non-null message according to $\text{msg-gen}^{[i]}$ and sends it to all its out-neighbors. At time t_1, each out-neighbor j of agent i receives a messages and processes it according to $\text{msg-rec}^{[j]}$. If multiple messages are received at the same time, then we allow all possible orders of execution of the message-reception function.

(iii) Additionally, at every instant of time $t_2 \in \mathbb{R}_{\geq 0}$ such that one of the state-transition triggers satisfies $\text{stf-trig}_k^{[i]}(x^{[i]}(t_2), w^{[i]}(t_2), y^{[i]}(t_2)) = \texttt{true}$, agent i updates its processor state $w^{[i]}$ according to $\mathrm{stf}_k^{[i]}$. If multiple state transitions are triggered at the same time, then we allow all possible orders of execution of the state-transition functions.

(iv) If one or multiple state-transition and message triggers are equal to `true` at the same time, then we assume that all state transitions take place first, and immediately after the messages are generated and transmitted.

(v) If one or multiple state-transition triggers are equal to `true` at the same time at which messages are received, then we allow all possible orders of execution of the state-transition and message-reception functions.

(vi) In order to avoid the possibility of an infinite number of message transmissions or state transitions in finite time, we introduce a "dwell logic." Let $\delta > 0$ be a *dwell time* common to all agents. For each agent i, if a message was generated at time t_1 by agent i, then no additional message is to be generated before time $t_1 + \varepsilon$ by agent i independently of the value of its message-trigger function. Similarly, for each agent i, if a state-transition function was executed at time t_2 by agent i, then no additional state-transition function is to be executed before time $t_2+\varepsilon$ by agent i independently of the value of its state-transition-trigger function.

Remark 6.2 (Dwell time prevents Zeno behavior). Note that: (1) a dwell time is introduced only for the purpose of properly defining an execution for a general event-drive control and communication law; (2) infinite numbers of message transmissions or state transitions in finite time will not take place during the execution of the algorithm that we present later in the chapter; and (3) we refer the reader interested in comprehensive treatments of the so-called Zeno behavior to van der Schaft and Schumacher (2000); Johansson et al. (1999). •

Finally, as we did in Chapter 3, we now give a formal definition of the asynchronous evolution of an event-driven control and communication law on a robotic network.

Definition 6.3 (Asynchronous event-driven evolution with dwell time). Let $\mathcal{ECC}$ be an event-driven control and communication law for the robotic network $\mathcal{S}$. For $\delta \in \mathbb{R}_{>0}$, the *evolution of* $(\mathcal{S}, \mathcal{ECC})$ *with dwell time* δ from initial conditions $x_0^{[i]} \in X_0^{[i]}$ and $w_0^{[i]} \in W_0^{[i]}$, $i \in I$, is the collection of absolutely continuous curves $x^{[i]} : \mathbb{R}_{\geq 0} \to X^{[i]}$, $i \in I$, and piecewise-constant curves $w^{[i]} : \mathbb{R}_{\geq 0} \to W^{[i]}$, $i \in I$, such that at almost all times,

$$\dot{x}^{[i]}(t) = f\big(x^{[i]}(t), \operatorname{ctl}^{[i]}\big(x^{[i]}(t), w^{[i]}(t)\big)\big), \tag{6.1.1}$$

$$\dot{w}^{[i]}(t) = 0, \tag{6.1.2}$$

with $x^{[i]}(0) = x_0^{[i]}$ and $w^{[i]}(0) = w_0^{[i]}$, $i \in I$, and such that:

(i) For every $i \in I$ and $t_1 \in \mathbb{R}_{>0}$, a message is generated by agent i and received by all its out-neighbors j, that is,

$$y_i^{[j]}(t_1) = \text{msg-gen}^{[i]}\big(x^{[i]}(t_1), w^{[i]}(t_1), j\big),$$
$$w^{[j]}(t_1) = \text{msg-rec}^{[j]}\big(x^{[j]}(t_1), \lim_{t \to t_1^-} w^{[j]}(t), y_i^{[j]}(t_1), i\big),$$

if $\text{msg-trig}^{[i]}(x^{[i]}(t_1), w^{[i]}(t_1)) = \texttt{true}$ and agent i has not transmitted any message to its out-neighbors during the time interval $]t_1 - \delta, t_1[\cap \mathbb{R}_{>0}$. Here, agent j is an out-neighbor of agent i at time t_1 if $(i,j) \in E_{\text{cmm}}\big(x^{[1]}(t_1), \dots, x^{[n]}(t_1)\big)$.

(ii) For every $i \in I$, $k \in \{1, \dots, K_{\text{stf}}^{[i]}\}$, and $t_2 \in \mathbb{R}_{>0}$, the state-transition function $\text{stf}_k^{[i]}$ is executed, that is,

$$w^{[i]}(t_2) = \text{stf}_k^{[i]}\big(x^{[i]}(t_2), \lim_{t \to t_2^-} w^{[i]}(t)\big),$$

if $\text{stf-trig}_k^{[i]}(x^{[i]}(t_2), w^{[i]}(t_2)) = \texttt{true}$ and there has been no execution of $\text{stf}_k^{[i]}$ during the time interval $]t_2 - \delta, t_2[\cap \mathbb{R}_{>0}$. •

This model of event-driven control and communication law and of asynchronous evolution is adopted in the rest of this chapter.

6.2 PROBLEM STATEMENT

In this section, we formalize the network objective. We begin by reviewing some important facts on interpolations of planar boundaries by means of inscribed polygons. After introducing the robotic network model, we make use of notions from the theory of linear interpolations to formalize the boundary estimation task.

6.2.1 Linear interpolations for boundary estimation

Consider a simply connected set Q in $\mathbb{R}^2$, that we term the *body*. We are interested in obtaining a description of the boundary ∂Q of a convex body. In particular, we will consider the *symmetric difference* error metric (Gruber, 1983) to measure the goodness of an approximation to ∂Q. The symmetric difference δ^S between two compact bodies C, $B \subseteq \mathbb{R}^2$ is defined by

$$\delta^S(C, B) = \mathrm{A}(C \cup B) - \mathrm{A}(C \cap B),$$

where, given a set $S \subset \mathbb{R}^2$, $\mathrm{A}(S)$ is its Lebesgue measure. This definition is illustrated in Figure 6.1. We note that the symmetric difference admits alternative definitions (see Exercise E6.1). In what follows, we search for

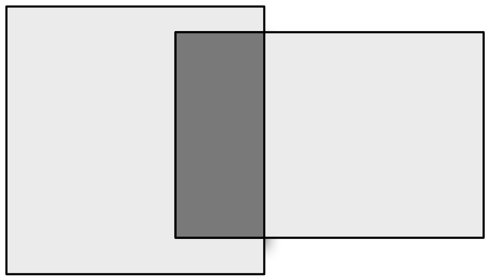


Figure 6.1 The symmetric difference between the two quadrilaterals is the area corresponding to the region colored in light gray.

approximations to a convex body Q by means of *inscribed polygons*. A convex polygon is inscribed in Q if all its vertices belong to the boundary of Q. We denote an inscribed polygon with m vertices by Q_m. The symmetric difference between the body Q and the polygon Q_m takes the simpler form

$$\delta^S(Q, Q_m) = \mathrm{A}(Q) - \mathrm{A}(Q_m).$$

The inscribed polygons that are critical points of δ^S can be characterized as follows.

Lemma 6.4 (Characterization of critical inscribed polygons for the symmetric difference). *Let Q be a convex planar body with a continuously differentiable boundary. Let Q_m be an inscribed polygon with vertices $\{q_1, \dots, q_m\}$ in counterclockwise order. For $i \in \{1, \dots, m\}$, let $t(q_i)$ be the tangent vector to ∂Q at q_i. Then, Q_m is a critical point of δ^S if and only if*

$$t(q_i) \text{ is parallel to } (q_{i+1} - q_{i-1}), \tag{6.2.1}$$

for all $i \in \{1, \dots, m\}$, where $q_0 = q_m$ and $q_{m+1} = q_1$.

Note that the characterization of critical inscribed polygons in Lemma 6.4 can be satisfied not only by polygons that are maximizers, but also by saddle points (see Exercise E6.2). On the other hand, we would like to make use of a characterization that can be extended to nonconvex bodies and that relies as much as possible on local information that agents can collect.

In what follows, we describe the *method of empirical distributions*, based on the asymptotic formula provided in the following lemma. We start with some useful notation. As in Section 1.1, let ∂Q be twice continuously differentiable and let $\gamma_{\text{arc}} : [0, L] \to \partial Q$ be a counterclockwise arc-length parametrization of ∂Q. Additionally, let $\kappa_{\text{signed}} : [0, L] \to \mathbb{R}$, $\kappa_{\text{abs}} : [0, L] \to \mathbb{R}_{\geq 0}$, and $\rho : [0, L] \to \mathbb{R}_{\geq 0}$ be, respectively, the signed curvature, the absolute curvature, and radius of curvature of the boundary. For convex bodies, the following result is proved in McLure and Vitale (1975), and Gruber (1983).

Lemma 6.5 (Optimal polygonal approximation of a convex body). *Let Q be a convex planar body whose boundary is twice continuously differentiable and has strictly positive signed curvature κ_{signed}. If Q_m^* is an optimal approximating polygon of Q, then*

$$\lim_{m \to +\infty} m^2 \delta^S(Q, Q_m^*) = \frac{1}{12} \int_0^L \rho(s)^{2/3} ds.$$

To compute an optimal approximating polygon for a strictly convex body, McLure and Vitale (1975) suggest the following method of empirical distributions. Let $q_1, \dots, q_m$ be consecutive points on ∂Q ordered counterclockwise and, for $i \in \{1, \dots, m\}$, define $s_i \in [0, L]$ by requiring $q_i = \gamma_{\text{arc}}(s_i)$. The positions q_i, $i \in \{1, \dots, m\}$, along ∂Q are said to obey the *method of empirical distributions* if

$$\int_{s_{i-1}}^{s_i} \rho(s)^{2/3} ds = \int_{s_i}^{s_{i+1}} \rho(s)^{2/3} ds \tag{6.2.2}$$

for all $i \in \{1, \dots, m\}$, where we set $s_0 = s_m$ and $s_{m+1} = s_1$. Interpolating polygons computed according to the method of empirical distributions converge to an optimal polygon approximation Q_m^* as $m \to \infty$. Roughly speaking, this property translates into the placement of more interpolation points on those parts of the boundary that have higher curvature. Figure 6.2.1 illustrates an approximating polygon with empirically distributed vertices. As final comment about convex bodies, it is useful to know from Gruber

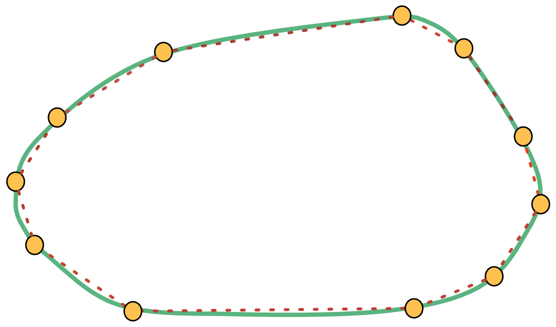


Figure 6.2 Equidistant interpolation points according to the integral in equation (6.2.2). The solid green line represents the boundary and the dashed red line represents the optimal approximating polygon.

(1983) that, for $\alpha > 0$,

$$\int_0^L \rho(s)^\alpha ds = \int_0^L \kappa_{\mathrm{abs}}(s)^{1-\alpha} ds. \tag{6.2.3}$$

Next, we discuss the case of nonconvex bodies whose boundary can be parameterized by a twice continuously differentiable curve. We begin with a definition: given a twice continuously differentiable curve $\gamma : [0, L] \to \mathbb{R}^2$, an *inflection point* of γ is a point $q \in \gamma([0, L])$ with the property that, for $q = \gamma(s_q)$, $\mathrm{sign}(\kappa_{\mathrm{signed}}(s_q - \varepsilon)) \neq \mathrm{sign}(\kappa_{\mathrm{signed}}(s_q + \varepsilon))$ for every $\varepsilon \in \mathbb{R}_{>0}$ sufficiently small. Nonconvex bodies have an arbitrary number of inflection points; we restrict our attention to nonconvex bodies with a finite number of them. Because the radius of curvature of a nonconvex body is unbounded at inflection points, equality (6.2.2) is ill posed in general. Therefore, in order to extend the method of empirical distributions to nonconvex bodies, we introduce the following notions of distance along a boundary. Given two points $q_i = \gamma_{\mathrm{arc}}(s_i)$ and $q_j = \gamma_{\mathrm{arc}}(s_j)$, with $s_i < s_j$, we define

$$\mathcal{D}_{\mathrm{curvature}}(q_i, q_j) = \int_{s_i}^{s_j} \kappa_{\mathrm{abs}}(s)^{1/3} ds,$$
$$\mathcal{D}_{\mathrm{arc}}(q_i, q_j) = s_j - s_i.$$

Note that the quantity $\mathcal{D}_{\mathrm{arc}}(q_i, q_j)$ is always strictly positive for $q_i \neq q_j$, whereas the quantity $\mathcal{D}_{\mathrm{curvature}}(q_i, q_j)$ vanishes if the points q_i and q_j are connected by a straight line. Additionally, for $\lambda \in [0, 1]$, we define the pseudo-distance $\mathcal{D}_\lambda$ between the vertices q_i and q_j as

$$\mathcal{D}_\lambda(q_i, q_{i+1}) = \lambda \mathcal{D}_{\text{curvature}}(q_i, q_j) + (1-\lambda)\mathcal{D}_{\text{arc}}(q_i, q_{i+1}).$$

The empirical distribution criterion (6.2.2) is substituted by the following one when the boundary is nonconvex. We look for approximations of ∂Q, $\{q_1, \dots, q_m\}$, such that $\mathcal{D}_\lambda(q_{i-1}, q_i) = \mathcal{D}_\lambda(q_i, q_{i+1})$ for all $i \in \{1, \dots, m\}$. This choice has the following interpretation. Taking $\lambda \approx 1$ leads to an interpolation that satisfies a modified method of empirical distributions. The method is modified in the sense that we adopt the distance $\mathcal{D}_{\text{curvature}}$ instead of the integral of the curvature radius; our informal justification for this step is equality (6.2.3). Instead, taking $\lambda \approx 0$ leads to an interpolation that divides the boundary into segments of equal arc-length. A choice of $\lambda \in (0,1)$ leads to a polygon approximation that is midway between these two options. For sufficiently large $\lambda < 1$, the resulting polygon has a higher number of vertices in the portions of the boundary with higher curvature and the distance between any two consecutive interpolation points is guaranteed to be positive.

6.2.2 Network model and boundary estimation task

Next, we formulate a robotic network model and the boundary estimation objective. Assume that Q is a simply connected subset of $\mathbb{R}^2$ with differentiable boundary ∂Q. Consider the network $\mathcal{S}_{\text{bndry}} = (I, \mathcal{R}, E_{\text{cmm}})$, with $I = \{1, \dots, n\}$. In this network, each robot is described by a tuple

$$(\partial Q, [-v_{\min}, v_{\max}], \partial Q, (0, \mathbf{e})), \tag{6.2.4}$$

where $\mathbf{e}$ is the vector field tangent to ∂Q describing counterclockwise motion at unit speed; we assume that unit speed is an admissible speed, that is, we assume that $1 \in [-v_{\min}, v_{\max}]$. We assume that each robot can sense its own location $p^{[i]} \in \partial Q$, $i \in I$, and can communicate with its clockwise and counterclockwise neighbors along ∂Q. In other words, the communication graph E_{cmm} is the ring graph or the Delaunay graph on ∂Q. Later, we shall assume that Q varies in a continuously differentiable way with time, and that, therefore, agents move along its time-varying boundary.

Next, assume that the processor state of each agent contains a set of n_{ip} interpolation points used to approximate ∂Q, that is, the processor state is given by $q^{[i]} \in (\mathbb{R}^2)^{n_{\text{ip}}}$, for $i \in I$. We illustrate the combination of agents and interpolation points along the boundary in Figure 6.3.

For $\varepsilon \in \mathbb{R}_{>0}$ and $\lambda \in [0,1]$, the *boundary estimation task* $\mathcal{T}_{\varepsilon\text{-bndry}} : (\partial Q)^n \times$

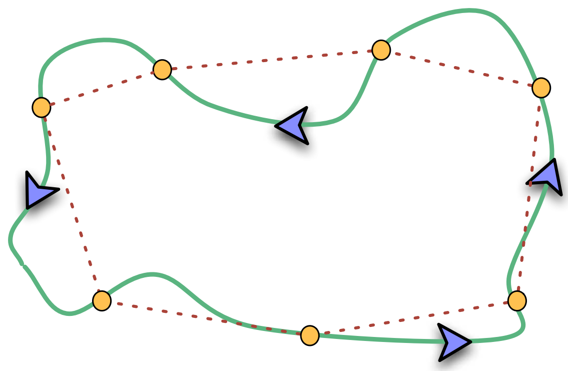


Figure 6.3 Agents and interpolation points on the boundary ∂Q.

$((\mathbb{R}^2)^{n_{\text{ip}}})^n \to \{\texttt{true}, \texttt{false}\}$ for $\mathcal{S}_{\text{bndry}}$ is the coordination task

$$\mathcal{T}_{\varepsilon\text{-bndry}}(p^{[1]}, \dots, p^{[n]}, q^{[1]}, \dots, q^{[n]}) = \texttt{true} \quad \text{if and only if}$$
$$\left|\mathcal{D}_\lambda(q^{[i]}_{\alpha-1}, q^{[i]}_{\alpha}) - \mathcal{D}_\lambda(q^{[i]}_{\alpha}, q^{[i]}_{\alpha+1})\right| < \varepsilon, \quad \alpha \in \{1, \dots, n_{\text{ip}}\} \text{ and } i \in I.$$

Roughly speaking, this task is achieved when the n_{ip} interpolation points are approximately uniformly placed along the boundary according to the counterclockwise pseudo-distance $\mathcal{D}_\lambda$.

A second objective is our desire to space the agents equally far apart along the boundary. As in Example 3.22, for $\varepsilon > 0$, we define the *agent equidistance task* $\mathcal{T}_{\varepsilon\text{-eqdstnc}} : (\partial Q)^n \to \{\texttt{true}, \texttt{false}\}$ to be $\texttt{true}$ if and only if

$$\left|\mathcal{D}_{\text{arc}}(p^{[i-1]}, p^{[i]}) - \mathcal{D}_{\text{arc}}(p^{[i]}, p^{[i+1]})\right| < \varepsilon, \quad \text{for all } i \in I,$$

where $\mathcal{D}_{\text{arc}}$ is the counterclockwise arc-length distance along ∂Q. In other words, $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ is true when, for every agent, the (unsigned) distances to the closest clockwise neighbor and to the closest counterclockwise neighbor are approximately equal.

6.3 ESTIMATE UPDATE AND CYCLIC BALANCING LAW

Here, we propose a coordination algorithm for a robotic network to achieve the boundary estimation task. The algorithm requires individual agents to maintain and continuously update an approximation of the boundary that asymptotically meets the criterion of the method of empirical distributions. The algorithm is an event-driven control and communication law, as defined in Section 6.1. To facilitate the understanding, the algorithm is presented in an incremental way. First, we specify an estimate update law for a single robot. Second, we consider multiple robots cooperatively performing the estimate update law to achieve the boundary estimation task. Third and finally, we introduce a cyclic balancing algorithm to achieve a robot equidistance task.

6.3.1 Single-robot estimate update law

Let Q be a simply connected subset of $\mathbb{R}^2$ with a differentiable moving boundary ∂Q. Consider a single robot described by (6.2.4) that moves along ∂Q. Assume that the processor state contains a set of interpolation points $\{q_1, \dots, q_{n_{\text{ip}}}\}$ used to approximate ∂Q. We begin with an informal description of the SINGLE-ROBOT ESTIMATE UPDATE LAW and we illustrate in Figure 6.4 the two actions characterizing this law:

> *[Informal description]* The agent moves counterclockwise along the moving boundary ∂Q, collecting estimates of its tangent and curvature. Using these estimates, the agent executes the following two actions. First, it updates the positions of the interpolation points so that they take value on the estimate of ∂Q. In other words, as sufficient information is available, each interpolation point q_α, $\alpha \in \{1, \dots, n_{\text{ip}}\}$, is projected onto the estimated boundary. Second, after an interpolation point q_α has been projected, the agent collects sufficient information so that it can locally optimize the location of q_α along the estimate of ∂Q. Here, by an estimate of the time-varying ∂Q, we mean the trajectory of the agent along the moving boundary.

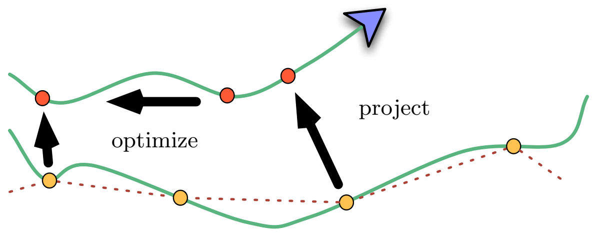


Figure 6.4 The two actions characterizing the SINGLE-ROBOT ESTIMATE UPDATE LAW.

Next, we begin our detailed description of the algorithm by specifying what variables the agent maintains in its memory. The processor state of the agent consists of the following variables:

(i) A counter `nxt` taking values in $\{1, \dots, n_{\text{ip}}\}$ that specifies which interpolation point the agent is going to project next.

(ii) A boundary representation comprised of the pairs

$$\{(q_\alpha, v_\alpha) \in (\mathbb{R}^2)^2 \mid \alpha \in \{1, \dots, n_{\mathrm{ip}}\}\},$$

where q_α is the position of the interpolation point α and v_α represents the tangent vector of ∂Q at q_α.

(iii) A curve of the form $\texttt{path} : [0, t] \to \mathbb{R}^2$. This curve is the trajectory followed by the agent from initial time until present time t. We assume that the agent updates the variable `path` continuously. We let $\mathcal{C}(\mathbb{R}^2)$ be the set of planar curves, that is, twice differentiable functions from an interval to $\mathbb{R}^2$. With this notation, we may write $\texttt{path} \in \mathcal{C}(\mathbb{R}^2)$.

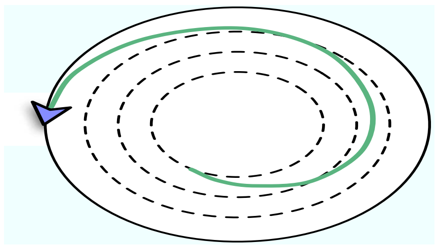


Figure 6.5 The agent moves along the time-varying boundary ∂Q, here depicted as a sequence of growing ellipses, and its trajectory is an approximation to ∂Q.

Remark 6.6 (Boundary approximation). For simplicity, we assume that, at every instant of time, the agent is located exactly on top of the boundary. This assumption implies that if the boundary is time-invariant, then the agent trajectory `path` is locally equal to ∂Q. Furthermore, if the boundary is slowly time-varying, then the agent's trajectory `path` is an estimate of the moving boundary ∂Q, as illustrated in Figure 6.5. •

The agent updates its processor state according to the following two rules:

Rule #1: When and how to project onto ∂Q the interpolation point $q_{\texttt{nxt}}$. Let $q_{\texttt{nxt}}$ denote the interpolation point about to be projected and let $v_{\texttt{nxt}}$ denote the corresponding tangent vector. The projection takes place when the agent crosses the line, denoted by $\texttt{line}_{\texttt{nxt}}$, that passes through $q_{\texttt{nxt}}$ and is perpendicular to $v_{\texttt{nxt}}$. At this crossing time, we define the updated values for the interpolation point `nxt`, denoted by $q^+_{\texttt{nxt}}$, to be the point on `path` where the agent's trajectory crosses the line $\texttt{line}_{\texttt{nxt}}$. This projection operation is illustrated in Figure 6.6. We refer to this operation by the map $\texttt{perp-proj} : (\mathbb{R}^2)^2 \times \mathcal{C}(\mathbb{R}^2) \to \mathbb{R}^2$; in other words, we write

$$q^+_{\texttt{nxt}} := \texttt{perp-proj}(q_{\texttt{nxt}}, v_{\texttt{nxt}}, \texttt{path}).$$

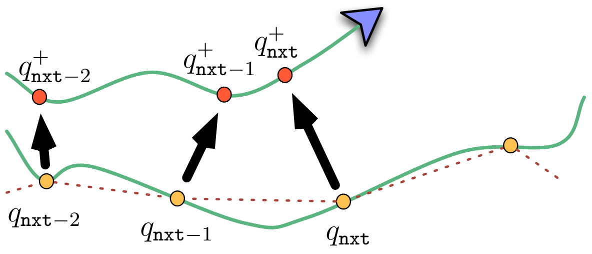


Figure 6.6 The projection of interpolation point $q_{\mathtt{nxt}}$ onto the curve `path`.

Rule #2: When and how to optimize the interpolation point $q_{\mathtt{nxt}-1}$. The local optimization of the interpolation point $(\mathtt{nxt}-1)$ takes place immediately after the projection of the interpolation point `nxt` onto the estimated boundary. The interpolation point $(\mathtt{nxt}-1)$ is moved along the curve `path` in order to balance its two pseudodistances to its clockwise and counterclockwise neighboring interpolation points (recall that `path` is an estimate of the boundary ∂Q as discussed in Remark 6.6). Specifically, we define the map `cyclic-balance` : $(\mathbb{R}^2)^3 \times \mathcal{C}(\mathbb{R}^2) \to \mathbb{R}^2$ by

$$\mathtt{cyclic\text{-}balance}(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}, \mathtt{path}) \text{ is point } q^* \text{ in the curve } \mathtt{path}$$
$$\text{such that } \mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q^*) = \frac{3}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) + \frac{1}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}})\,.$$

This map is illustrated in Figure 6.7. With this definition, we update the interpolation $(\mathtt{nxt}-1)$ to be

$$q^+_{\mathtt{nxt}-1} := \mathtt{cyclic\text{-}balance}(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}, \mathtt{path}).$$

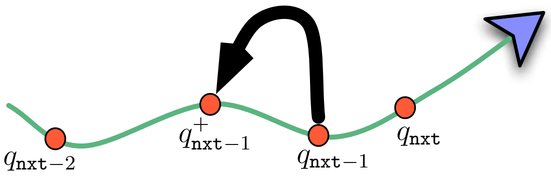


Figure 6.7 Optimal placement of the interpolation point $q_{\mathtt{nxt}-1}$ along the curve `path`.

Remark 6.7 (Balancing property of the optimal placement). The optimal placement $q^+_{\mathtt{nxt}-1}$ can be equivalently defined by

$$\mathcal{D}_\lambda(q^+_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}) = \frac{1}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) + \frac{3}{4}\mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}),$$

so that it achieves the balancing property that

$$\begin{bmatrix} \mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q^+_{\mathtt{nxt}-1}) \\ \mathcal{D}_\lambda(q^+_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} \mathcal{D}_\lambda(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}) \\ \mathcal{D}_\lambda(q_{\mathtt{nxt}-1}, q_{\mathtt{nxt}}). \end{bmatrix}$$

This iteration is the same as the cyclic balancing system with parameter $k = 1/4$ studied in Exercises E1.30, E5.5, and E6.3. •

Finally, we define one last useful operation. Given a point on the curve path, it is useful to be able to compute the tangent of the curve path at the point q. Specifically, given a point q on the curve path, we shall write

$$v := \mathtt{tangentat}(\mathtt{path}, q).$$

In summary, the Single-Robot Estimate Update Law is formally described as follows:

`Robot:` single robot moving at constant speed along ∂Q, continuously recording its trajectory

`Event-driven Algorithm:` Single-Robot Estimate Update Law

`Processor State:` $w = (\mathtt{nxt}, \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}, \mathtt{path})$, where

`nxt`	$\in \{1, \dots, n_{\text{ip}}\}$,	initially equal to index of interpolation point closest to robot moving counterclockwise
$\{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}$	$\subset \mathbb{R}^2 \times \mathbb{R}^2$,	initially counterclockwise along boundary
`path`	$\in \mathcal{C}(\mathbb{R}^2)$,	continuously recording agent's trajectory

```
% A state transition is triggered when the agent crosses a certain line
function stf-trig($p, w$)
1: $\mathtt{line}_{\mathtt{nxt}} :=$ line through point $q_{\mathtt{nxt}}$ perpendicular to direction $v_{\mathtt{nxt}}$
2: if $p \in \mathtt{line}_{\mathtt{nxt}}$ then
3:     return true
4: else
5:     return false
```

```
% The current interpolation point and tangent vector are projected and the
   previous interpolation point is optimized along the new boundary
function stf($p, w$)
1: $\{(q^+_\alpha, v^+_\alpha)\}_{\alpha=1}^{n_{\text{ip}}} := \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}$
2: $q^+_{\mathtt{nxt}} := \mathtt{perp\text{-}proj}(q_{\mathtt{nxt}}, v_{\mathtt{nxt}}, \mathtt{path})$
3: $q^+_{\mathtt{nxt}-1} := \mathtt{cyclic\text{-}balance}(q_{\mathtt{nxt}-2}, q_{\mathtt{nxt}-1}, q^+_{\mathtt{nxt}}, \mathtt{path})$
4: $v^+_{\mathtt{nxt}} := \mathtt{tangentat}(\mathtt{path}, q^+_{\mathtt{nxt}})$
5: $v^+_{\mathtt{nxt}-1} := \mathtt{tangentat}(\mathtt{path}, q^+_{\mathtt{nxt}-1})$
6: return $(\mathtt{nxt}+1, \{(q^+_\alpha, v^+_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}, \mathtt{path})$
```

The SINGLE-ROBOT ESTIMATE UPDATE LAW may be improved in a number of ways; here, we present some important algorithmic clarifications.

Remarks 6.8 (Content and representation of the variable `path`).

(i) The above discussion assumes that, with the information provided by the variable `path`, the agent can compute the tangent vector and the curvature along its trajectory in order to perform the calculation of the pseudodistance $\mathcal{D}_\lambda$.

(ii) It is not necessary for the agent to keep in the variable `path` its entire trajectory since initial time. In fact, when the agent updates the location of the interpolation point $(\texttt{nxt} - 1)$ in instruction `4:` of the state-transition function, it is sufficient that `path` contains the trajectory of the agent starting from interpolation point $(\texttt{nxt} - 2)$ until the current agent position. This "limited-length" requirement may be implemented as follows: the variable `path` is a curve of the form $\texttt{path} : [\texttt{t}_{\texttt{path}}, t] \to \mathbb{R}^2$, the variable t denotes the present time, the variable $\texttt{t}_{\texttt{path}}$ is initially set equal to 0, and the instruction

$$\texttt{t}_{\texttt{path}} := t^* \text{ such that } q_{\texttt{nxt}-1} = \texttt{path}(t^*),$$

is executed before instruction `6:` in the state-transition function.

(iii) In a realistic implementation, the `path` variable and its first two derivatives are to be represented with finite resolution over a discrete time domain. The interested reader is referred to Susca et al. (2008) for a discussion of the SINGLE-ROBOT ESTIMATE UPDATE LAW algorithm with a realistic implementation of the `path` variable in a finite-length finite-resolution manner. •

Remark 6.9 (Timeout for the projection of interpolation points). In the definition of the SINGLE-ROBOT ESTIMATE UPDATE LAW, we have implicitly assumed that the agent crosses the line through $q_{\texttt{nxt}}$ perpendicular to $v_{\texttt{nxt}}$. This is certainly the case if ∂Q is static or slowly time-varying. However, if ∂Q changes drastically, it is conceivable that the agent never crosses the line through $q_{\texttt{nxt}}$ perpendicular to $v_{\texttt{nxt}}$. In other words, it can happen that the state-transition trigger function is always `false`. This situation can be prevented by prescribing a timeout such that, if the agent has not crossed the line after a certain time has elapsed, then the interpolation point `nxt` is projected onto its trajectory anyway. Formally, let t^* be implicitly defined by

$$\mathcal{D}_\lambda(q^+_{\texttt{nxt}-1}, p(t)) = 2\mathcal{D}_\lambda(q_{\texttt{nxt}-1}, q_{\texttt{nxt}}).$$

If no crossing has happened at time t, then $q^+_{\texttt{nxt}}$ is set equal to the point on `path` that is closest to $q_{\texttt{nxt}}$. The tangent vector $v^+_{\texttt{nxt}}$ is set equal to

$\texttt{tangentat}(\texttt{path}, q^+_{\texttt{nxt}})$. This projection is well-defined and has the property that if ∂Q is time-invariant, then $q^+_{\texttt{nxt}} = q_{\texttt{nxt}}$. The algorithm described by Susca et al. (2008) explicitly incorporates this timeout. •

6.3.2 Cooperative estimate update law

In the previous section, we presented an event-driven algorithm for a single robot to monitor a boundary. We consider the robotic network $\mathcal{S}_{\text{bndry}}$ with ring communication topology, described in Section 6.2.2, and we develop a parallel version of the SINGLE-ROBOT ESTIMATE UPDATE LAW that allows the network to monitor the boundary efficiently (i.e., with more accuracy than a single robot could). We begin with an informal description of the COOPERATIVE ESTIMATE UPDATE LAW:

> *[Informal description]* Each agent moves counterclockwise along the moving boundary ∂Q, has its individual copy of the boundary representation, including the interpolation points, and executes the SINGLE-ROBOT ESTIMATE UPDATE LAW. Because the agents are spatially distributed, each agent updates its individual boundary representation separately. On top of the SINGLE-ROBOT ESTIMATE UPDATE LAW, the agents run a communication protocol that transmits the updated interpolation points along the ring topology. Specifically, every time an agent updates two interpolation points (using the state-transition function of the SINGLE-ROBOT ESTIMATE UPDATE LAW and, thus, the functions `perp-proj` and `cyclic-balance`), this agent transmits these updated interpolation points to its clockwise and counterclockwise neighbors. In turn, the neighbors record the updates in their individual boundary representation.

Next, we give a more detailed description of the algorithm. We assume that each robot i has a processor state with its local copy of $w^{[i]}$ containing a counter $\texttt{nxt}^{[i]}$, a boundary representation $\{(q^{[i]}_\alpha, v^{[i]}_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}$, where n_{ip} is equal for all robots, and its $\texttt{path}^{[i]}$. The COOPERATIVE ESTIMATE UPDATE LAW is formally described as follows:

`Robotic Network:` $\mathcal{S}_{\text{bndry}}$, first-order agents moving at unit speed along ∂Q with absolute sensing of own position, communicating with clockwise and counterclockwise neighbors

`Event-driven Algorithm:` COOPERATIVE ESTIMATE UPDATE LAW

`Alphabet:` $\mathbb{A} = \{1, \dots, n_{\text{ip}}\} \times (\mathbb{R}^2)^2 \times (\mathbb{R}^2)^2 \cup \{\texttt{null}\}$

```
Processor State, function stf-trig, and function stf
                        same as in SINGLE-ROBOT ESTIMATE UPDATE LAW
% A transmission is triggered right after the interpolation points are updated
function msg-trig($p, w$)
1: return stf-trig($p, w$)

% The updated interpolation points (and reference label) are transmitted
function msg-gen($p, w, i$)
1: return $\big(\texttt{nxt}, (q_{\texttt{nxt}-1}, v_{\texttt{nxt}-1}), (q_{\texttt{nxt}-2}, v_{\texttt{nxt}-2})\big)$

% The received updated interpolation points are stored
function msg-rec($p, w, y, i$)
1: $\{(q_\alpha^+, v_\alpha^+)\}_{\alpha=1}^{n_{\text{ip}}} := \{(q_\alpha, v_\alpha)\}_{\alpha=1}^{n_{\text{ip}}}$
2: $(\texttt{nxtrec}, y_1, y_2) := y$
3: $(q_{\texttt{nxtrec}-1}^+, v_{\texttt{nxtrec}-1}^+) := y_1$
4: $(q_{\texttt{nxtrec}-2}^+, v_{\texttt{nxtrec}-2}^+) := y_2$
5: return $(\texttt{nxt}, \{(q_\alpha^+, v_\alpha^+)\}_{\alpha=1}^{n_{\text{ip}}}, \texttt{path})$
```

We conclude this section with an important clarification. We begin with a useful definition and then give two related remarks.

Definition 6.10 (Two-hop separation). A group of $n \geq 2$ agents implementing the COOPERATIVE ESTIMATE UPDATE LAW is *two-hop separated along the interpolation points* if $\texttt{nxt}^{[i-1]} \leq \texttt{nxt}^{[i]} - 2$ for all $i \in I$ at all times during the execution of the algorithm. •

Remarks 6.11 (Well-posedness of the Cooperative Estimate Update Law).

(i) The inequality $\texttt{nxt}^{[i-1]} \leq \texttt{nxt}^{[i]} - 2$ guarantees that each robot can correctly implement the algorithm. Indeed, if this inequality is violated, then the `cyclic-balance` function performed by robot i during the state-transition function is invoked with an interpolation point $q_{\texttt{nxt}-2}^{[i]}$ which does not take value in the curve $\texttt{path}^{[i]}$ (because the boundary might be time-varying). This inequality therefore guarantees that the algorithm is well-posed. Assuming two-hop separation guarantees that the projection and optimization events happen in the following order: each interpolation point $\texttt{nxt}^{[i]}$ is projected and later optimized by robot i, strictly before it is projected by robot $(i-1)$.

(ii) The two-hop separation property is easily seen to hold when (1) the number of interpolation points n_{ip} is much larger than the num-

"Distributed Control of Robotic Networks" by F. Bullo, J. Cortés and S. Martínez

ber of robots n, (2) the robots are approximately equidistant along ∂Q, and (3) the distances between pairs of consecutive interpolation points are much less than the length of ∂Q divided by n. •

6.3.3 Cyclic balancing algorithm for agent equidistance task

Here, we propose a motion coordination controller to achieve the agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ among the agents moving along the boundary. This task also leads to the orderly interactions mentioned in the last remark. Specifically, we extend the Cooperative Estimate Update Law to include a motion coordination component that makes the agents achieve the agent equidistance task while moving at approximately unit speed along the boundary. The control design is straightforward: for robot i at position $p^{[i]}$ moving in continuous time with speed $v^{[i]}$ along ∂Q, we define

$$v^{[i]} = 1 + k_{\text{prop}}\big(\mathcal{D}_{\text{arc}}(p^{[i]}, p^{[i+1]}) - \mathcal{D}_{\text{arc}}(p^{[i-1]}, p^{[i]})\big), \qquad (6.3.1)$$

where $k_{\text{prop}} \in \mathbb{R}_{>0}$ is a fixed control gain. In other words, the agent speeds up or slows down depending upon whether it is closer to the following or to the preceding agent, respectively. This simple motion control law is the continuous-time analog of the cyclic balancing system described in Exercise E1.30; recall that this system was adopted also in the Single-Robot Estimate Update Law for the purpose of balancing pseudodistances among interpolation points.

To handle the lower and upper bounds constraints on the velocity, that is, the constraint $v \in [-v_{\min}, v_{\max}]$, we introduce a saturation function in the design (6.3.1). Specifically, we implement

$$v^{[i]} = \text{sat}_{[v_{\min}, v_{\max}]}\Big(1 + k_{\text{prop}}\big(\mathcal{D}_{\text{arc}}(p^{[i]}, p^{[i+1]}) - \mathcal{D}_{\text{arc}}(p^{[i-1]}, p^{[i]})\big)\Big), \qquad (6.3.2)$$

where the saturation function $\text{sat}_{[a,b]} : \mathbb{R} \to [a, b]$, for $a < b$, is defined by

$$\text{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x < a, \\ x, & \text{if } x \in [a, b], \\ b, & \text{if } x > b. \end{cases}$$

The difficulty in implementing controller (6.3.2) in the Cooperative Estimate Update Law is how to measure the counterclockwise arc-length distance between robots. To tackle this difficulty, let us begin with a useful observation. Given the interpolation points $\{q_1, \dots, q_{n_{\text{ip}}}\}$ and two points on the boundary r_1, r_2, assume that sufficient information is available to compute the indices $[r_1]$ and $[r_2]$ of the counterclockwise-closest interpolation points from r_1, r_2, respectively. With this notation and the assumption, the counterclockwise pseudodistance from r_1 to r_2 and one of its possible approximations are as follows:

$$\mathcal{D}_{\mathrm{arc}}(r_1, r_2) = \mathcal{D}_{\mathrm{arc}}(r_1, q_{[r_1]}) + \sum_{\alpha=[r_1]}^{[r_2]-2} \mathcal{D}_{\mathrm{arc}}(q_\alpha, q_{\alpha+1}) + \mathcal{D}_{\mathrm{arc}}(q_{[r_2]-1}, r_2) \tag{6.3.3}$$

$$\approx \mathrm{dist}_2(r_1, q_{[r_1]}) + \sum_{\alpha=[r_1]}^{[r_2]-2} \mathrm{dist}_2(q_\alpha, q_{\alpha+1}) + \mathrm{dist}_2(q_{[r_2]-1}, r_2). \tag{6.3.4}$$

Based on this equality and on this approximation, we propose two methods to implement the controller (6.3.2). One may implement either of the following:

(i) The approximation proposed in (6.3.4); this approximated pairwise counterclockwise arc-length distance may be computed with the information available to the agents in the COOPERATIVE ESTIMATE UPDATE LAW.

(ii) The exact computation proposed in (6.3.3); in order to perform this computation, however, the robots require more information. The processor state is required to store a collection of arc-length distances $\mathcal{D}_{\mathrm{arc}}(q_\alpha, q_{\alpha+1})$, $\alpha \in \{1, \dots, n_{\mathrm{ip}}\}$ that are measured by the agents as they move, and that are maintained accurate via communication. In the interest of brevity, we omit a detailed discussion of this point here.

Finally, independently of the computation or approximation of the arc-length distances, the implementation of controller (6.3.2) requires each agent to have a continuous-time estimate of the location of its clockwise and counterclockwise neighbors: this information may be acquired by either a dedicated message-exchanging protocol or, possibly, by proximity sensors mounted on the robots. In the interest of brevity, we omit a detailed discussion of this point here.

6.3.4 Correctness of the estimate update and cyclic balancing law

We call the ESTIMATE UPDATE AND BALANCING LAW the combination of the COOPERATIVE ESTIMATE UPDATE LAW with the cyclic balancing control law (6.3.2), with exact arc-length distance computation between robots.

We call the APPROXIMATE ESTIMATE AND BALANCING LAW the combination of the COOPERATIVE ESTIMATE UPDATE LAW with the cyclic balancing control law (6.3.2), with (1) finite-resolution finite-length representation of the `path` variable in the robots, and (2) approximate arc-length distance computation between robots.

We state the properties of these laws in the following theorem, whose proof is postponed to Section 6.6.

Theorem 6.12 (Correctness of the exact and approximate laws). *On the network $\mathcal{S}_{\text{bndry}}$, along evolutions with the two-hop separation property:*

(i) the ESTIMATE UPDATE AND BALANCING LAW *achieves the boundary estimation task $\mathcal{T}_{\varepsilon\text{-bndry}}$ and the agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ for any $\varepsilon \in \mathbb{R}_{>0}$ if the boundary is time-independent; and*

(ii) the APPROXIMATE ESTIMATE AND BALANCING LAW *achieves the boundary estimation task $\mathcal{T}_{\varepsilon\text{-bndry}}$ and the agent equidistance task $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ for some $\varepsilon \in \mathbb{R}_{>0}$ if the boundary varies in a continuously differentiable way and sufficiently slowly with time, and its length is upper bounded.*

Remark 6.13 (Error induced by the evolution of the boundary and its discretization). In the second statement in the theorem, the constant ε depends upon the rate of change of the boundary and upon the accuracy of the various approximations made in the algorithm. •

6.4 SIMULATION RESULTS

In order to illustrate the performance of the algorithms, we include here different simulation results of the APPROXIMATE ESTIMATE AND BALANCING LAW. In the first simulation, the boundary to be estimated is time invariant, while in the second it is time-varying.

Time-invariant boundary

As a first simulation, we assume that $n = 3$ agents aim to approximate the time-invariant boundary ∂Q described by the closed curve

$$\gamma(\theta) = \big(2 + \cos(5\theta) + 0.5\sin(2\theta)\big) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \quad \theta \in [0, 2\pi].$$

The control gain is $k_{\text{prop}} = 0.05$. The minimum and maximum velocities are $v_{\min} = 0.5$ and $v_{\max} = 2$. The number of interpolation points is $n_{\text{ip}} = 30$.

Pseudodistances are computed with $\lambda = \frac{10}{11}$. The simulation time is 50 seconds. At initial time, the interpolation points are selected to be the positions of the agents and other randomly distributed points on the boundary. Finally, each robot maintains a discretized representation of its trajectory `path` with a resolution of 0.01 seconds.

The behavior of the Approximate Estimate and Balancing Law is illustrated in Figure 6.8. The left- and right-hand figures correspond to the positions of the interpolation points and the agents at the initial and final configurations, respectively. In the right-hand figure one can see the approximating polygon and how close it is to the actual boundary.

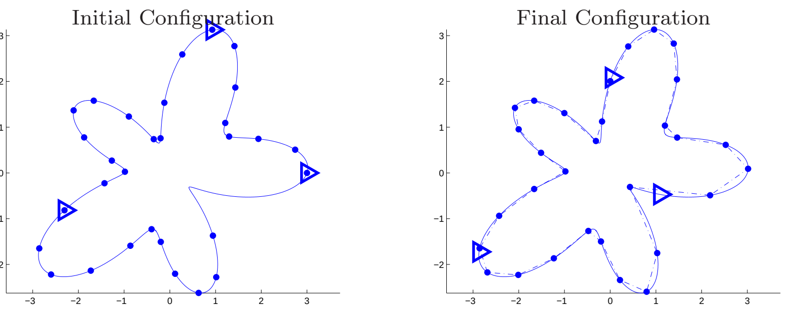


Figure 6.8 The Approximate Estimate and Balancing Law for a time-invariant boundary: initial and final configuration of agents (drawn as triangles) and interpolation points. The right-hand figure shows also the approximating polygon.

Figure 6.9 illustrates the convergence of the algorithm. Although the Approximate Estimate and Balancing Law uses an approximated version of the pseudodistances between interpolation points and of the arc-length distance between agents, we illustrate the performance of the algorithm by plotting the exact versions of the pseudodistances and arc-length distances. Regarding the boundary estimation task, the left-hand figure illustrates how the quantity $\max_{\alpha\in\{1,\dots,n_{\text{ip}}\}} \mathcal{D}_\lambda(q_\alpha, q_{\alpha+1}) - \min_{\alpha\in\{1,\dots,n_{\text{ip}}\}} \mathcal{D}_\lambda(q_\alpha, q_{\alpha+1})$ does indeed decrease towards zero, even though it does not vanish because of the adopted approximations. Regarding the equidistance task, the right figure illustrates how the agents become uniformly spaced along the boundary. Again, the arc-length distances converge toward a common steady-state value, even though the convergence is not exact because of the adopted approximations.

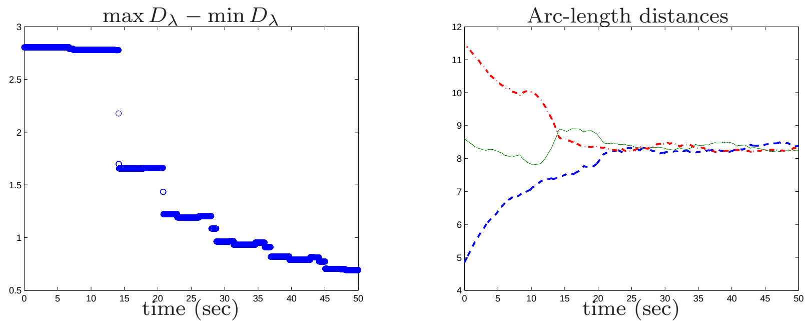


Figure 6.9 The Approximate Estimate and Balancing Law for a time-invariant boundary: the left-hand figure shows the largest minus the smallest pseudodistance between neighboring interpolation points. The right-hand figure shows the three arc-length distances between the three agents.

Slowly time-varying boundary

As a second simulation, we assume that $n = 4$ agents aim to approximate the time-varying boundary ∂Q described by the time-varying closed curve

$$\gamma(\theta, t) = \left(2\frac{t_{\text{final}} - t}{t_{\text{final}}} + \Big(2 + \cos(5\theta) + 0.5\sin(2\theta)\Big)\frac{t}{t_{\text{final}}}\right)\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix},$$

with $\theta \in [0, 1]$, $t \in [0, t_{\text{final}}]$, and $t_{\text{final}} = 200$ seconds, as shown in Figure 6.10. The parameters and initial conditions of the Approximate Estimate and Balancing Law are the same as in the time-invariant case. The four plots in Figure 6.10 show the positions of the interpolation points and of the agents at the four time instants 0, 50, 100, and 200 seconds, respectively. The last plot also illustrates how close the approximating polygon is to the actual boundary. From the frames in Figure 6.10, it is clear that the agents can adapt as ∂Q changes.

6.5 NOTES

For a discussion of hybrid systems we refer to van der Schaft and Schumacher (2000). Other relevant references include Lygeros et al. (2003), Goebel et al. (2004), and Sanfelice et al. (2007).

Many methods are currently available (Mehaute et al., 1993) for the approximation of planar curves; this fact is largely motivated by computational and signal-processing applications. Among them, the use of interpolated curves is a standard and important approach. In their most simple version, interpolations provide polygonal approximations of curves, with generalizations that make use of splines, or combinations of functions in a certain basis. In particular, the problem of characterizing the polygons that optimally ap-

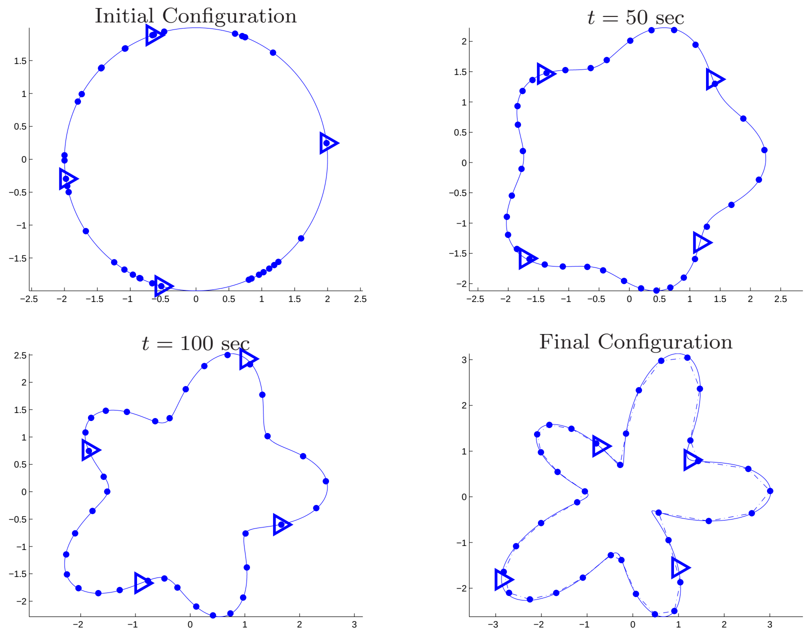


Figure 6.10 The Approximate Estimate and Balancing Law for a time-varying boundary: the configuration of the agents (drawn as triangles) and interpolation points at time instants 0, 50, 100, and 200 seconds. The last figure also shows the approximating polygon.

proximate a closed, convex body is a classical one; see the survey by Gruber (1983). In particular, the asymptotic formula in Lemma 6.5 was extended in (Gruber, 1983) to higher dimensions in terms of the Gauss curvature.

Boundary estimation and tracking is useful is numerous applications such as the detection of harmful algal blooms (Marthaler and Bertozzi, 2003; Bertozzi et al., 2004), oil spills (Clark and Fierro, 2007), and fire containment (Casbeer et al., 2005, 2006). Marthaler and Bertozzi (2003) adopt the so-called “snake algorithm” (from the computer vision literature) to detect and track the boundary of harmful algal bloom. Each agent is equipped with a chemical sensor that is able to measure the concentration gradient and with a communication system that is able to exchange information with a data fusion center. Bertozzi et al. (2004) suggest an algorithm that requires only a concentration sensor: the agents repeatedly cross the region boundary using a bang–bang angular velocity controller. Clark and Fierro (2007) use a random coverage controller, a collision avoidance controller, and a bang–bang angular velocity controller to detect and surround an oil spill. Casbeer et al. (2006) describe an algorithm that allows Low Altitude Short Endurance Unmanned Vehicles (LASEUVs) to closely monitor the boundary of a fire. Each of the LASEUVs has an infrared camera and a short-range communication device to exchange information with other agents, and to download the information collected to the base station. In Zhang and Leonard (2005), a formation of four robots tracks at unitary speed, the level sets of a field. Their relative positions change, so that they can optimally measure the gradient and estimate the curvature of the field in the center of the formation. In Zhang and Leonard (2007), a controller is proposed to steer a group of constant-speed robots onto an equally spaced configuration along a close curve.

6.6 PROOFS

This section presents the main result of the chapter on the correctness of Theorem 6.12. For the sake of completeness, we review first some notations and main concepts for Input-to-State-Stability (ISS) of discrete-time systems, as introduced in Angeli (1999), Jiang and Wang (2001), and Angeli (1999); Sontag (2008). We then make use of these to prove the result of Theorem 6.12.

6.6.1 Review of ISS concepts

A function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a *$\mathcal{K}$-function* if it is continuous, strictly increasing, and $\gamma(0) = 0$. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a *$\mathcal{KL}$-function* if, for each $t \in \mathbb{R}_{\geq 0}$, the function $s \mapsto \beta(s,t)$ is a $\mathcal{K}$-function, and for each $s \in \mathbb{R}_{\geq 0}$, $t \mapsto \beta(s,t)$ is decreasing and $\beta(s,t) \to 0$ as $t \to +\infty$.

Consider the discrete-time nonlinear system

$$x(\ell+1) = f(x(\ell), u(\ell)), \tag{6.6.1}$$

where ℓ takes values in $\mathbb{Z}_{\geq 0}$, x takes values in $\mathbb{R}^n$, and u takes values in $\mathbb{R}^m$. We assume that $f : \mathbb{Z}_{\geq 0} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is continuous. In what follows, we let $\|u\|_{2,\infty} = \sup\{\|u(\ell)\|_2 \mid \ell \in \mathbb{Z}_{\geq 0}\} \leq +\infty$.

Definition 6.14 (Input-to-state stability). The system (6.6.1) is *input-to-state stable (ISS)* if there exist a $\mathcal{KL}$-function β and a $\mathcal{K}$-function γ such that, for each initial condition $x_0 \in \mathbb{R}^n$ at time $\ell_0 \in \mathbb{Z}_{\geq 0}$ and for each bounded input $u : \mathbb{Z}_{\geq 0} \to \mathbb{R}^m$, the system evolution x satisfies, for each $\ell \geq \ell_0$,

$$\|x(\ell)\|_2 \leq \beta(\|x_0\|_2, \ell - \ell_0) + \gamma(\|u\|_{2,\infty}). \qquad \bullet$$

Definition 6.15 (ISS-Lyapunov function). A function $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is an *ISS-Lyapunov function* for system (6.6.1) if:

(i) it is continuously differentiable;

(ii) there exist $\mathcal{K}_\infty$-functions, α_1, α_2, such that $\alpha_1(\|x\|_2) \leq V(x) \leq \alpha_2(\|x\|_2)$; and

(iii) there exist a $\mathcal{K}_\infty$-function α_3 and a $\mathcal{K}$-function σ such that

$$V(f(x,u)) - V(x) \leq -\alpha_3(\|x\|_2) + \sigma(\|u\|_2). \qquad \bullet$$

We refer to Jiang and Wang (2001) for a proof of the following result.

Theorem 6.16 (ISS and Lyapunov functions). *System (6.6.1) is ISS if and only if it admits an ISS-Lyapunov function.*

6.6.2 Proof of Theorem 6.12

Proof. In the interest of brevity, we prove only the statements that pertain to the boundary estimation task $\mathcal{T}_{\varepsilon\text{-bndry}}$, that is, to the task

$$\left| \mathcal{D}_\lambda(q^{[i]}_{\alpha-1}, q^{[i]}_{\alpha}) - \mathcal{D}_\lambda(q^{[i]}_{\alpha}, q^{[i]}_{\alpha+1}) \right| < \varepsilon,$$

for all $\alpha \in \{1, \dots, n_{\text{ip}}\}$ and $i \in I$. We refer to Susca et al. (2008) for the proof of the statements regarding the agent equidistance task.

We begin our analysis with the case of a single robot, that is, we consider the ESTIMATE UPDATE AND BALANCING LAW algorithm, and we first consider the case of a time-invariant boundary with exact `path` representation and with no approximations in any computation. Define the shorthand $\mathcal{D}_\alpha = \mathcal{D}_\lambda(q_\alpha, q_{\alpha+1})$, for $\alpha \in \{1, \dots, n_{\text{ip}}\}$, and the positive vector $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_{n_{\text{ip}}}) \in \mathbb{R}^{n_{\text{ip}}}_{>0}$. We now characterize how the vector $\mathcal{D}$ changes after one application of the state-transition function with counter `nxt` in the SINGLE-ROBOT ESTIMATE UPDATE LAW. We refer to an application of the state-transition function as a projection-and-placement event. Because the boundary is time-invariant, the projection operation (performed by the function `perp-proj`) leaves the interpolation point $q_{\texttt{nxt}}$ unchanged. Furthermore, as discussed in Remark 6.7, the placement operation (performed by the function `cyclic-balance`) modifies the interpolation point $q_{\texttt{nxt}-1}$ so that

$$\begin{bmatrix} \mathcal{D}_{\texttt{nxt}-2} \\ \mathcal{D}_{\texttt{nxt}-1} \end{bmatrix}^+ = \frac{1}{4} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \mathcal{D}_{\texttt{nxt}-2} \\ \mathcal{D}_{\texttt{nxt}-1} \end{bmatrix},$$

where we adopt the shorthand $\mathcal{D}^+_\alpha = \mathcal{D}_\lambda(q^+_\alpha, q^+_{\alpha+1})$, for $\alpha \in \{1, \dots, n_{\text{ip}}\}$. For

$\texttt{nxt} \in \{1, \dots, n_{\text{ip}}\}$, define $A_{\texttt{nxt}} \in \mathbb{R}^{n_{\text{ip}} \times n_{\text{ip}}}$ by

$$(A_{\texttt{nxt}})_{jk} = \begin{cases} 3/4, & \text{if } (j,k) \text{ equals } (\texttt{nxt}-1, \texttt{nxt}-1) \text{ or } (\texttt{nxt}-2, \texttt{nxt}-2), \\ 1/4, & \text{if } (j,k) \text{ or } (j,k) \text{ equals } (\texttt{nxt}-1, \texttt{nxt}-2), \\ \delta_{jk}, & \text{otherwise} \end{cases}$$

and define the undirected graph $G_{\texttt{nxt}}$ with vertices $\{1, \dots, n_{\text{ip}}\}$ and with the single edge $(\texttt{nxt}-1, \texttt{nxt}-2)$. In summary, the matrix $A_{\texttt{nxt}}$ determines the change of state $\mathcal{D}^+ = A_{\texttt{nxt}}\mathcal{D}$ when a projection-and-placement event takes place with counter nxt and its associated graph is $G_{\texttt{nxt}}$.

Because the boundary has finite length and the agent moves at lower-bounded speed, an infinite number of projection-and-placement events take place for each interpolation point. After a re-parametrization of time, let $\ell \in \mathbb{N}$ denote the times at which projection-and-placements events take place and let $\texttt{nxt}(\ell) \in \{1, \dots, n_{\text{ip}}\}$ denote the index corresponding to the event taking place at time ℓ. At each time $\ell \in \mathbb{N}$, we write

$$\mathcal{D}(\ell) = A_{\texttt{nxt}(\ell)}\mathcal{D}(\ell - 1). \tag{6.6.2}$$

Next, note that $A_{\texttt{nxt}(\ell)}$, for $\ell \in \mathbb{N}$, is a non-degenerate sequence of symmetric and doubly stochastic matrices. Additionally, note that the undirected graph $\cup_{\tau \geq \ell} G_{\texttt{nxt}(\tau)}$ is connected. Therefore, by Theorem 1.65 and Corollary 1.70, we know that, for all $\alpha \in \{1, \dots, n_{\text{ip}}\}$,

$$\lim_{\ell \to +\infty} \mathcal{D}_\alpha(\ell) = \frac{1}{n_{\text{ip}}} \sum_{\alpha=1}^{n_{\text{ip}}} \mathcal{D}_\alpha(0) = \frac{1}{n_{\text{ip}}}\big(\text{total pseudodistance length of } \partial Q\big).$$

This proves that the interpolation points become equally spaced along ∂Q with respect to pseudodistance and that the boundary estimation task is achieved in the time-invariant case. In other words, this concludes the proof of the boundary estimation part of statement (i) for a single robot in Theorem 6.12.

Next, we consider the COOPERATIVE ESTIMATE UPDATE LAW for networks of $n \geq 2$ agents. Each agent has maintains a local copy of the pseudodistance vector and of the interpolation points (which always take value in ∂Q because the boundary is time-invariant). Specifically, for $i \in \{1, \dots, n\}$, agent i maintains vector $\mathcal{D}^{[i]}$ and interpolation points $q_\alpha^{[i]}$, for $\alpha \in \{1, \dots, n_{\text{ip}}\}$. We define the *aggregate pseudodistance vector* $\mathcal{D}$ as follows: we let $\mathcal{D}_\alpha$ equal the most recently updated element of the vector $\{\mathcal{D}_\alpha^{[1]}, \dots, \mathcal{D}_\alpha^{[n]}\}$, that is, the pseudodistance between the most recently updated interpolation points $q_\alpha^{[i]}$ and $q_{\alpha+1}^{[i]}$, for $i \in \{1, \dots, n\}$. As before, after a re-parametrization of time, let $\ell \in \mathbb{N}$ denote the times at which projection-and-placements events take place (independently of which agent i executes the event) and let $\texttt{nxt}(\ell) \in \{1, \dots, n_{\text{ip}}\}$ denote the index corresponding to the event taking place at time ℓ (independently of which agent i executes the event). Note that when agent i updates the interpolation points $q_{\mathtt{nxt}(\ell)-1}$ and $q_{\mathtt{nxt}(\ell)-2}$, agent i then transmits the updated points to its immediately following agent $i-1$, so that the processor state of agent $i-1$ contains the correct updated information. Also note that since the boundary is time-invariant, the updated interpolation points belong to the trajectory $\mathtt{path}^{[i-1]}$ that agent $i-1$ maintains in its memory: this fact guarantees that agent $i-i$ can properly perform the `cyclic-balance` operation. In summary, equation (6.6.2) is the correct model not only for the SINGLE-ROBOT ESTIMATE UPDATE LAW but also for the COOPERATIVE ESTIMATE UPDATE LAW. This concludes the proof of the boundary estimation part of statement (i) for $n \geq 2$ robots in Theorem 6.12.

Let us now relax the assumption on the boundary and consider a time-varying $t \mapsto \partial Q(t)$; as before, we first consider the case of a single robot. We assume that pseudodistances between interpolation points along the agent `path` curve are computed exactly. By assumption, the boundary ∂Q varies in a continuously differentiable way and slowly in time and, therefore, the projection of the interpolation points is well defined and unique. For the case of a time-varying boundary, the state trajectory in continuous time is a curve of the form $\mathcal{D} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_{\text{ip}}}$ defined as follows. Note that, in general, the interpolation points lie outside ∂Q at almost all times, and therefore it makes no sense to define $\mathcal{D}_\alpha$ as the pseudodistance from point q_α to $q_{\alpha+1}$ along ∂Q. Rather, we give the following definition: $\mathcal{D}$ is the vector of pseudodistances computed by the robot along the curve `path`. As a consequence, the state trajectory $\mathcal{D}$ is constant for almost all times and it changes only at projection-and-placement events. Specifically, let ℓ denote the time at which a projection-and-placements event takes place with corresponding index $\mathtt{nxt}(\ell) \in \{1, \dots, n_{\text{ip}}\}$. At time ℓ, the SINGLE-ROBOT ESTIMATE UPDATE LAW computes new values for the pseudodistances $\mathcal{D}_{\mathtt{nxt}-2}$ and $\mathcal{D}_{\mathtt{nxt}-1}$ based on the processor state of the agent (i.e., based on the interpolation points and the `path` variable); these values are the new values of the state $\mathcal{D}$. Because the boundary has upper-bounded length uniformly in time and because the agent moves at constant speed, an infinite number of projection-and-placement events takes place for each interpolation point and the duration of time between two consecutive events is uniformly upper bounded. Given this fact, we may let $\ell \in \mathbb{N}$ serve as index for all projection-and-placement times. Clearly, if the boundary does not vary with time, then the transition $\mathcal{D}(\ell) = A_{\mathtt{nxt}(\ell)}\mathcal{D}(\ell-1)$ describes the projection-and-placement event at instant ℓ. Because, instead, the boundary is time-varying, we model the change in $\mathcal{D}$ due to the boundary motion by

$$\mathcal{D}(\ell) = A_{\mathtt{nxt}(\ell)}\big(\mathcal{D}(\ell-1) + \mathcal{U}(\ell)\big), \tag{6.6.3}$$

where $\mathcal{U}(\ell) \in \mathbb{R}^{n_{\text{ip}}}$ is a disturbance. By design, $\mathcal{U}(\ell)$ is nonzero only on components $(\mathtt{nxt}(\ell)-1)$ and $(\mathtt{nxt}(\ell)-2)$ of $\mathcal{D}$. By the assumptions that the boundary varies in a continuously differentiable way and slowly with time, and that its length is upper bounded, we know that $\mathcal{U}$ is vanishing in the rate of change of the boundary. Finally, define the disagreement vector $\ell \to \mathbf{d}(\ell) \in \operatorname{span}\{\mathbf{1}_{n_{\mathrm{ip}}}\}^{\perp}$ by

$$\mathbf{d}(\ell) = \mathcal{D}(\ell) - \frac{\mathbf{1}_{n_{\mathrm{ip}}}^T \mathcal{D}(\ell)}{n_{\mathrm{ip}}} \mathbf{1}_{n_{\mathrm{ip}}}. \tag{6.6.4}$$

From equation (6.6.3) and from the fact that $A_{\mathtt{nxt}(\ell)}$ is doubly stochastic, the update law for $\mathbf{d}$ is

$$\mathbf{d}(\ell) = A_{\mathtt{nxt}(\ell)} \mathbf{d}(\ell-1) + \mathbf{u}(\ell), \quad \ell \in \mathbb{N}, \tag{6.6.5}$$

where $\mathbf{u}(\ell) = \mathcal{U}(\ell) - \frac{1}{n_{\mathrm{ip}}} \mathbf{1}_{n_{\mathrm{ip}}}^T \mathcal{U}(\ell) \mathbf{1}_{n_{\mathrm{ip}}}$.

Equation (6.6.5) is the correct update equation even in the case of $n \geq 2$ robots moving along a time-varying boundary. This fact is a consequence of the two-hop separation assumption (see Definition 6.10). Indeed, as explained in Remarks 6.11, the inequality $\mathtt{nxt}^{[i-1]} \leq \mathtt{nxt}^{[i]} - 2$ guarantees that each robot can correctly perform each projection-and-placement event. Given the sequence $\ell \in \mathbb{N}$, define a new sequence $\ell_k \in \mathbb{N}$, for $k \in \mathbb{N}$, as follows: set $\ell_1 = 1$, assume without loss of generality that agent 1 is the agent executing the first projection-and-placement event with index $\mathtt{nxt}(1)$, and let $\ell_k \geq 2$ be the k-th time when agent 1 performs the projection-and-placement event with same index $\mathtt{nxt}(1)$. Reasoning about the possible positions of all agents at time ℓ_{k-1} and ℓ_k, one can see that $\ell_k - \ell_{k-1} \leq 2n \cdot n_{\mathrm{ip}}$. Define sequence $\mathcal{A}_{\ell_k} \in \mathbb{R}^{n_{\mathrm{ip}} \times n_{\mathrm{ip}}}$, for $k \in \mathbb{N}$, by $\mathcal{A}(1) = A_{\mathtt{nxt}(1)}$ and

$$\mathcal{A}(\ell_k) = A_{\mathtt{nxt}(\ell_k)} \cdots A_{\mathtt{nxt}(\ell_{k-1}+2)} A_{\mathtt{nxt}(\ell_{k-1}+1)}, \quad \text{for } k \geq 2.$$

By Exercise E1.17, each matrix $\mathcal{A}(\ell_k)$ is doubly stochastic and irreducible, because it is the product of doubly stochastic matrices and because the union of the undirected graphs associated with the matrices defining $\mathcal{A}(\ell_k)$ is connected. By definition, equation (6.6.5) becomes, for $k \in \mathbb{N}$,

$$\begin{aligned} \mathbf{d}(\ell_k) &= \mathcal{A}(\ell_k)\mathbf{d}(\ell_{k-1}) + \sum_{\ell=\ell_{k-1}+1}^{\ell_k} A_{\mathtt{nxt}(\ell_k)} \cdots A_{\mathtt{nxt}(\ell+1)} \mathbf{u}(\ell) \\ &= \mathcal{A}(\ell_k)\mathbf{d}(\ell_{k-1}) + \mathcal{B}(\ell_k)\mathbf{u}_{\mathrm{stacked}}(\ell_k), \end{aligned} \tag{6.6.6}$$

where the vector $\mathbf{u}_{\mathrm{stacked}}(\ell_k)$ contains all vectors $\mathbf{u}(\ell_{k-1}+1), \dots, \mathbf{u}(\ell_k)$, and the matrix $\mathcal{B}(\ell_k)$ is defined in the trivial corresponding way.

Define $V : \mathbb{R}^{n_{\mathrm{ip}}} \to \mathbb{R}_{\geq 0}$ by $V(x) = x^T x$ and adopt this function as a candidate ISS-Lyapunov function for system (6.6.6). We compute

$$V(\mathbf{d}(\ell_{k+1})) - V(\mathbf{d}(\ell_k)) = -\mathbf{d}(\ell_k)^T R(\ell_k)\mathbf{d}(\ell_k) \\ + \mathbf{u}_{\text{stacked}}^T(\ell_k)\mathbf{u}_{\text{stacked}}(\ell_k) + 2\mathbf{u}_{\text{stacked}}^T(\ell_k)\mathcal{A}(\ell_k)\mathbf{d}(\ell_k),$$

where $R(\ell_k) = I_{n_{\text{ip}}} - \mathcal{A}(\ell_k)^T\mathcal{A}(\ell_k)$. Because $\mathcal{A}(\ell_k)$ is doubly stochastic and irreducible, we know (from Exercise E1.5) that $R(\ell_k)$ is positive semidefinite and that its simple eigenvalue 0 is associated with the eigenvector $\mathbf{1}_{n_{\text{ip}}}$. This fact implies that the quantity $-x^T R(\ell_k)x$ is strictly negative for all $x \notin \text{span}\{\mathbf{1}_{n_{\text{ip}}}\}^{\perp}$. To upper bound this quantity by a negative number, we let $\mathcal{A}_s$ be a generic element of the set of all the possible matrices $\mathcal{A}(\ell_k)$; such matrices are the iterated products of at most $2n \cdot n_{\text{ip}}$ matrices of the form $A_{\texttt{nxt}}$, where each matrix $A_{\texttt{nxt}}$, $\texttt{nxt} \in \{1, \dots, n_{\text{ip}}\}$, appears at least once. Define the set of nonzero eigenvalues of $\mathcal{A}_s$ by

$$S_s = \{\lambda \in \mathbb{R} \mid \det\Big(\lambda I_{n_{\text{ip}}} - (\mathcal{A}_s^T\mathcal{A}_s - I_{n_{\text{ip}}})\Big) = 0\} \setminus \{0\}$$

and define the eigenvalue with smallest magnitude among all matrices by $\overline{r} = \min_s \min\{|\lambda| \mid \lambda \in S_s\}$. Note that $\overline{r} > 0$, because we are considering a finite set of matrices. We can then write

$$V(\mathbf{d}(\ell_k + 1)) - V(\mathbf{d}(\ell_k)) \le -\alpha_3(\|\mathbf{d}(\ell_k)\|) + \sigma(\|\mathbf{u}_{\text{stacked}}(\ell_k)\|),$$

where $\alpha_3(\|\mathbf{d}\|) = \frac{1}{2}\overline{r}\|\mathbf{d}\|^2$ and $\sigma(\|\mathbf{u}_{\text{stacked}}\|) = (\frac{2}{\overline{r}} + 1)\|\mathbf{u}_{\text{stacked}}\|^2$. By Definition 6.15, the system described by (6.6.6) is input-to-state stable. The input-to-state stability implies the existence of a positive ε, as in the boundary estimation part of statement (ii) for $n \ge 2$ robots in Theorem 6.12. ■

6.7 EXERCISES

E6.1 **(Alternative expression of the symmetric difference).** Show that the symmetric difference δ^S between two compact bodies C, $B \subseteq \mathbb{R}^3$ can be alternatively expressed as

$$\delta^S(C, B) = \mu(C \setminus B) + \mu(B \setminus C).$$

Hint: *Use the expressions* $C = (C \setminus B) \cup (C \cap B)$ *and* $B = (B \setminus C) \cup (C \cap B)$.

E6.2 **(Characterization of critical inscribed polygons for the symmetric difference).** Prove Lemma 6.4. Also, show that not all critical configurations are optimal. Specifically, consider the convex body and the gray inscribed triangle depicted in Figure E6.1(a). Show that the gray triangle is a saddle configuration for δ^S by establishing that modifications of the triangle as in Figure E6.1(b) decrease its area (and hence increase δ^S), whereas modifications of the gray triangle as in Figure E6.1(c) increase its area (and hence decrease δ^S).

E6.3 **(The "n-bugs problem" and cyclic interactions: cont'd).** Consider n robots at counterclockwise-ordered positions $\theta_1, \dots, \theta_n$ following the cyclic bal-

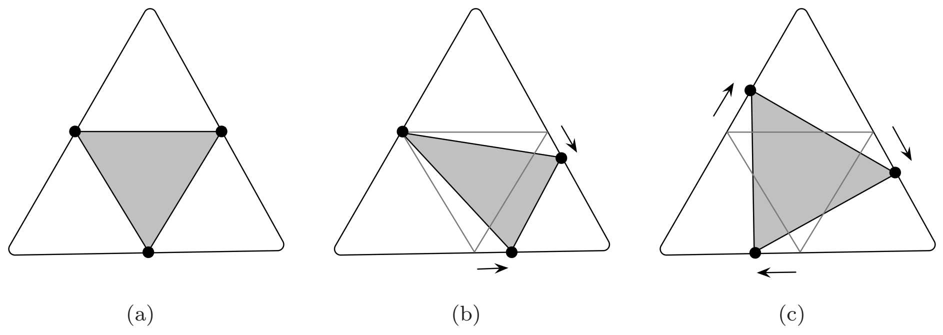


Figure E6.1 An illustration of the fact that polygons satisfying (6.2.1) may not be optimal for δ^S.

ancing system described in Exercise E1.30, with parameter $k = 1/4$:

$$\theta_i(\ell+1) = \frac{1}{4}\theta_{i+1}(\ell) + \frac{1}{2}\theta_i(\ell) + \frac{1}{4}\theta_{i-1}(\ell), \quad \ell \in \mathbb{Z}_{\geq 0}.$$

Show that

$$\mathrm{dist}_{\mathrm{cc}}\big(\theta_{i-1}(\ell), \theta_i(\ell+1)\big) = \frac{3}{4}\mathrm{dist}_{\mathrm{cc}}\big(\theta_{i-1}(\ell), \theta_i(\ell)\big) + \frac{1}{4}\mathrm{dist}_{\mathrm{cc}}\big(\theta_i(\ell), \theta_{i+1}(\ell)\big).$$

E6.4 **(ISS properties of averaging algorithms with inputs and outputs).** This is a guided exercise to prove some of the ISS properties of averaging algorithms with inputs and outputs. An *averaging algorithm with inputs* associated to a sequence of stochastic matrices $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\} \subseteq \mathbb{R}^{n\times n}$, a sequence of input gains $\{D(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\} \subseteq \mathbb{R}^{n\times k}$, and a sequence of disturbances $u : \mathbb{Z}_{\geq 0} \to \mathbb{R}^k$ is the discrete-time dynamical system

$$x(\ell+1) = F(\ell)x(\ell) + D(\ell)u(\ell), \quad \ell \in \mathbb{Z}_{\geq 0}. \tag{E6.1}$$

A natural question to ask is how the evolution of the trajectory x is affected by the noise u. Let us address this in the following. Define the matrix

$$P = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}.$$

Note that, with the notation of Exercise E1.7, one can write

$$T = \begin{bmatrix} P \\ \frac{1}{n}\mathbf{1}_n^T \end{bmatrix}.$$

Define the following output for the dynamical system (E6.1):

$$y_{\mathrm{err}} = Px = \begin{pmatrix} x_1 - x_2 \\ \vdots \\ x_{n-1} - x_n \end{pmatrix} \in \mathbb{R}^{n-1}.$$

This output can be thought of as an error signal that quantifies the disagreement among the components of the state. Now, consider the change of variables

$$z = Tx = \begin{pmatrix} y_{\mathrm{err}} \\ x_{\mathrm{ave}} \end{pmatrix},$$

where $x_{\mathrm{ave}} \in \mathbb{R}$ is the average of the components of x.

Verify that system (E6.1) reads in the new variable z as

$$z(\ell+1) = TF(\ell)T^{-1}\, z(\ell) + TD(\ell)u(\ell).$$

The previous result is a formal statement of the following intuition. Because of the definition of z and of the special structure of $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$, the variable x_{ave} plays no role in the evolution of y_{err}. Accordingly, we define the *error system* by

$$y_{\mathrm{err}}(\ell+1) = F_{\mathrm{err}}(\ell)y_{\mathrm{err}}(\ell) + D_{\mathrm{err}}(\ell)u(\ell), \tag{E6.2}$$

and the *average system* by

$$x_{\mathrm{ave}}(\ell+1) = x_{\mathrm{ave}}(\ell) + c_{\mathrm{err}}(\ell)y_{\mathrm{err}}(\ell) + D_{\mathrm{ave}}(\ell)u(\ell), \tag{E6.3}$$

for $D_{\mathrm{err}}(\ell) = PD(\ell)$ and $D_{\mathrm{ave}}(\ell) = \frac{1}{n}1_n^T D(\ell)$.

Assume now that:

(a) The sequence $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ is a non-degenerate sequence of stochastic matrices.

(b) For $\ell \in \mathbb{Z}_{\geq 0}$, let $G(\ell)$ be the unweighted digraph associated with $F(\ell)$. There exists a duration $\delta \in \mathbb{N}$ such that, for all $\ell \in \mathbb{Z}_{\geq 0}$ the digraph $G(\ell+1) \cup \ldots \cup G(\ell+\delta)$ contains a globally reachable node.

(c) The induced norm of $\{D(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$, for $\ell \in \mathbb{Z}_{\geq 0}$, is uniformly bounded.

Prove that, under assumptions (a), (b) and (c) on the averaging system with inputs, the following equivalent statements hold:

(i) the system (E6.1) with output y_{err} is IOS; and

(ii) the error system (E6.2) is ISS.

Bibliography

Angeli, D. [1999] *Intrinsic robustness of global asymptotic stability*, Systems & Control Letters, **38**(4-5), 297–307.

Bertozzi, A. L., Kemp, M., and Marthaler, D. [2004] *Determining environmental boundaries: Asynchronous communication and physical scales*, in *Cooperative Control*, V. Kumar, N. E. Leonard, and A. S. Morse, editors, volume 309 of *Lecture Notes in Control and Information Sciences*, pages 25–42, Springer, ISBN 3540228616.

Casbeer, D. W., Kingston, D. B., Beard, R. W., Mclain, T. W., Li, S.-M., and Mehra, R. [2006] *Cooperative forest fire surveillance using a team of small unmanned air vehicles*, International Journal of Systems Sciences, **37**(6), 351–360.

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

Clark, J. and Fierro, R. [2007] *Mobile robotic sensors for perimeter detection and tracking*, ISA Transactions, **46**(1), 3–13.

Goebel, R., Hespanha, J. P., Teel, A. R., Cai, C., and Sanfelice, R. G. [2004] *Hybrid systems: generalized solutions and robust stability*, in *IFAC Symposium on Nonlinear Control Systems*, pages 1–12, Stuttgart, Germany.

Gruber, P. M. [1983] *Approximation of convex bodies*, in *Convexity and its Applications*, P. M. Gruber and J. M. Willis, editors, pages 131–162, Birkhäuser, ISBN 3764313846.

Jiang, Z.-P. and Wang, Y. [2001] *Input-to-state stability for discrete-time nonlinear systems*, Automatica, **37**(6), 857–869.

Johansson, K. J., Egerstedt, M., Lygeros, J., and Sastry, S. S. [1999] *On the regularization of Zeno hybrid automata*, Systems & Control Letters, **38**(3), 141–150.

Lygeros, J., Johansson, K. H., Simić, S. N., Zhang, J., and Sastry, S. S. [2003] *Dynamical properties of hybrid automata*, IEEE Transactions on Automatic Control, **48**(1), 2–17.

Marthaler, D. and Bertozzi, A. L. [2003] *Tracking environmental level sets with autonomous vehicles*, in *Recent Developments in Cooperative Control and Optimization*, S. Butenko, R. Murphey, and P. M. Pardalos, editors, pages 317–330, Kluwer Academic Publishers, ISBN 1402076444.

McLure, D. E. and Vitale, R. A. [1975] *Polygonal approximation of plane convex bodies*, Journal of Mathematical Analysis and Applications, **51**(2), 326–358.

Mehaute, A. L., Laurent, P. J., and Schumaker, L. L., (editors) [1993] *Curves and Surfaces in Geometric Design*, A. K. Peters, ISBN 1568810393.

Sanfelice, R. G., Goebel, R., and Teel, A. R. [2007] *Invariance principles for hybrid systems with connections to detectability and asymptotic stability*, IEEE Transactions on Automatic Control, **52**(12), 2282–2297.

Sontag, E. D. [2008] *Input to state stability: Basic concepts and results*, in *Nonlinear and Optimal Control Theory*, P. Nistri and G. Stefani, editors, pages 163–220, Lecture Notes in Mathematics, Springer, ISBN 3540776443.

Susca, S. [2007] *Distributed Boundary Estimation and Monitoring*, Ph.D. thesis, University of California at Santa Barbara, available at `http://ccdc.mee.ucsb.edu`.

Susca, S., Martínez, S., and Bullo, F. [2008] *Monitoring environmental boundaries with a robotic sensor network*, IEEE Transactions on Control Systems Technology, **16**(2), 288–296.

van der Schaft, A. J. and Schumacher, H. [2000] *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*, Springer, ISBN 1852332336.

Zhang, F. and Leonard, N. E. [2005] *Generating contour plots using multiple sensor platforms*, in *IEEE Swarm Intelligence Symposium*, pages 309–316, Pasadena, CA.

— [2007] *Coordinated patterns of unit speed particles on a closed curve*, Systems & Control Letters, **56**(6), 397–407.

Algorithm Index

Subject Index

Symbol Index