NORTHWESTERN UNIVERSITY

# Real-Time Algorithms for Symbol-Based Automation

A THESIS

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

Anastasia Mavrommati

EVANSTON, ILLINOIS

September 2017

—-

# ABSTRACT

Real-Time Algorithms for Symbol-Based Automation

Anastasia Mavrommati

Intelligent behavior in humans is largely associated with encoding information to discrete symbols. However, symbolic behavior in robotic systems is not widespread, mainly due to lack of tools that constitute symbol-based automation implementable in real time. This thesis proposes real-time algorithms that facilitate and promote symbol-based *action* and *sensing*. In particular, we show that time-efficient control with symbolic actions is possible, using mode scheduling and, subsequently, finite state machines. Serving as the link between action and sensing is a model predictive control algorithm that performs adaptive *exploration*, driven by an information distribution. This exploration strategy is the key that enables sensing—i.e., learning and tracking—symbols in a variety of settings, reinforcing the importance of information equivalence. The proposed methodologies are validated through simulation examples, reflecting real-world situations, as well as experimentation that verifies real-time execution and implementability. In the final example of this thesis, we demonstrate—in experiment, using a mobile robot performing tactile exploration—how learned symbols can be subsequently employed as self-localization landmarks through their information signature.

## Acknowledgements

First and foremost I would like to thank my advisor, Professor Todd Murphey, for being immensely understanding, encouraging and supportive these past five years. I can say with certainty now that I wouldn't have made it this far if it was not for his inspiring enthusiasm and expert perspective guiding me through a difficult period full of classes and research, so thank you!

I would also like to thank my committee, Professor Brenna Argall and Professor Ed Colgate for taking time to read the manuscript and helping me advance my research career with their valuable advice.

Special thanks to my lab mates for their help and for being who they are—social, enthusiastic, open-minded and professional all at once. They all make our lab an enjoyable place to work in with lots of positive energy.

Thanks to my family and friends back in Greece, for always being there for me, even from so many miles away. Finally, to Emmanouil, my partner in life, the greatest of thanks for his calm, inspiring soul that supports me unconditionally. I've been blessed to have you by my side, thank you.

This work would not be possible without my funding resources: in particular, my thesis was supported by Army Research Office grant W911NF-14-1-0461 and by the National Science Foundation under awards CMMI-1200321 and IIS-1426961.

# Table of Contents

# List of Figures

CHAPTER 1

# Introduction

Humans sense signals and perform actions at a continuous level of granularity. For example, throwing a ball is the result of continuous signals generated by efferent neurons. Similarly, sensing an uneven surface through touch relies on an overwhelming amount of continuous signals generated by mechanoreceptors on the fingertip. However, intelligent behavior is largely associated with breaking down these continuous signals to discrete *symbols* [**14**]. For instance, if the sensed “uneven terrain” is the interior of a handbag, incoming information is encoded to a series of symbols: keys, wallet, notebook, tablet and so on. Drawing inspiration from the human world, this thesis aims to provide a complete, self-contained insight into symbol-based *action* (Part 1), *exploration* (Part 2), and *sensing and creation* (Part 3) for automated systems. Throughout, *real-time* capacity lies in the center as the irreplaceable key to implementability in real-world situations.

## 1.1. Part 1: ACTION

Traditionally, the need for symbolic action is seen in hybrid control theory—where continuous and discrete quantities coexist. Consequently, this thesis starts by exploring the options and proposing solutions for real-time control with symbolic actions and with minimal computational cost. The ability to select control symbols fast and on-the-go based on continuous feedback—just like humans pick left or right when walking to avoid an incoming pedestrian—is essential for robotic systems, because it frees up significant amount of processing potential intended for more intelligent tasks. For example, if avoiding incoming obstacles is broken down to left- or right-step decisions encoded in the circuitry of a finite state machine, the humanoid robot can use most of its processing capacity searching for the mailbox to throw its owner's letter in.

Two real-time symbolic control approaches are proposed. First, Chapter 2 introduces a mode scheduling algorithm that is rendered fast enough for real-time implementation [**15**, **16**]. The method is validated experimentally by regulating the swing angle of an overhead crane using symbolic actions i.e. accelerate left, accelerate right, and maintain current velocity. This experiment brought forth an interesting conclusion: crane regulation does, in fact, require only 2 bits of control information for real-time operation!

Relying on this outcome, this thesis goes further into investigating how symbolic control policies (also referred to as control alphabet policies) can be entirely encoded into finite state machines so that online control is computation-free. The solution to this problem will benefit automation systems in terms of computing power allocation and compactness by boosting their multi-tasking capacity (power allocation) and promoting miniaturization (compactness), in the fields of aviation [**17**], manufacturing [**18**], and robotic locomotion [**19**] among others. Chapter 3 proposes the solution of multi-resolution cell subdivision and validates it, most importantly, using the SLIP model for walking [**20**]. Section 4.1 quotes this result to justify the need for an information-driven exploration strategy in building control policies, leading to Part 2 of the thesis.

## 1.2. Part 2: EXPLORATION

In this thesis, a symbol $s$ is defined as a discrete action or sensed entity, uniquely identified by an information signature. Information signature is tied to the process of exploring (extracting and/or detecting) for a symbol. In sensed symbols, this is more or less straightforward; if I'm blindly searching for the keys (symbol) in my bag using the sense of touch (sensor), I'm exploring with respect to its information signature by tracking pointy edges, metallic texture etc. The information signature of keys, in this case, is engraved in my nervous system and can be different for each sensor used for exploration (visual, tactile, auditory, etc.). This internal representation of continuous neural signals as information signatures of discrete symbols (keys) is driven by the need to perform decision and control tasks in a manner that is independent of factors such as scaling, transformation and occlusions. This ability to use symbol-specific information equivalently for tracking different symbol configurations, will be referred to in the thesis as "*information equivalence*". The concept of information equivalence is valuable in humans—because we can search for, and identify, the keys in all sorts of different settings—but that's only the case because we have an intrinsic way of exploring with respect to this information. This relationship between information and exploration brings forth the need for an online exploration methodology that is information-driven in a distributed way and, most importantly, is fully automated.

Following this reasoning, Part 2 of my thesis is dedicated in proposing an information-driven exploration algorithm [**21**–**23**]—that meets the requirements that would allow a robot to sense symbols in real time—e.g., stability, adaptability, distributability and more (see Section 4.2). The concept of ergodicity plays a crucial role in achieving this. The goal is to build trajectories that improve ergodicity online, i.e. spend time in the field proportional to an information distribution. However, the ergodic metric is particularly hard to improve indefinitely—i.e., without a finite time horizon in mind. Consequently, the problem is framed as a hybrid control problem so that a single control action is computed at each time step to improve the ergodic cost functional. Chapter 6 verifies the algorithm performance in generating area coverage trajectories for a variety of different dynamic agents, including a model of a custom robotic fish built at the Michigan State University [**22**].

Part 2 is the link between symbolic action and symbolic sensing. Because, throughout the thesis, I distinguish between action symbols (i.e., discrete control actions) and sensed symbols (i.e., targets, object shapes etc.), the benefit of exploration for symbolic action might not be clear at first. However, Chapter 7 will show that all symbols are, to some extent, sensed information entities, that require exploration. In particular, we will see how we can "sense" the boundaries of action symbols on the state space through abstract exploration. See more about the dependence of action symbols on exploration in Section 4.1.

## 1.3. Part 3: SENSING

Part 2 shows that distribution-driven exploration in real time is possible. But, how can we use this outcome as an asset for enabling robots to sense in a symbolic manner? Part 3 is dedicated in analyzing methods that apply this exploration strategy and use it to sense symbols through their information signature. Before presenting the methodologies, we first establish the connection between symbolic action and sensing in Chapter 7, by introducing a preliminary idea for building control policies though abstract state-space exploration.

The primary contribution of Part 3 is two-fold: first, we show how to *track* known symbols (i.e. with known measurement models) and second, how to *learn* unknown symbols. Chapter 8 demonstrates symbol tracking in the context of target localization with bearing-only measurements. We start by showing how we can derive the information signature of a target (to be identified as symbol) using the Fisher Information on the target's measurement model. Subsequently, an agent can be controlled to track targets by being ergodic with respect to this expected information density. This is our first encounter with "information equivalence" in this thesis; real-time ergodic exploration allows us to track the symbols' information signature (as extracted through their measurement model in Section 8.1) regardless of their state or the agent's state and with no changes in the exploration process. We demonstrate the method in both simulation and in experiment, illustrating that it is independent of the number of symbols being tracked and can be run in real-time on computationally limited hardware platforms.

In target tracking, we assumed that the target's measurement model, and thus information signature, is known with some uncertainty. This however is hardly ever true. In most cases, we are searching for symbols that have not be identified as such yet and thus their information signature is unknown. For this reason, the thesis continues by using ergodic exploration to *learn* information signatures, in the context of tactile sensing of physical object shapes (identified as symbols). Subsequently, we show how the concept of information equivalence allows us to use the extracted information signatures to estimate symbol transformation so that the symbols act as stable natural landmarks for robot pose estimation.

# Part 1

# Acting with Symbols in Real Time

CHAPTER 2

# Real-Time Hybrid Control

This chapter is motivated by the problem of reliable and fast implementation of mode scheduling algorithms in real-time applications where control authority is discrete. Switched systems cover a range of real-world control platforms like antilock braking systems and other valve-operated settings. However, most current approaches to dynamic compensation rely on specialized ODE solvers to simulate the hybrid dynamics and thus exhibit high execution times and/or approximation errors. In our approach, we only require that a set of data is calculated offline and stored in memory so that online computational complexity is significantly reduced. Importantly, for memory efficiency, we show that approximation accuracy is independent of the number of stored samples i.e. the size of the stored dataset. Using ROS, we apply the proposed algorithm to regulate the swing angle of a mass suspended from a planar robotic system in real time with hybrid control signals. Our experimental results verify that our algorithm is fast and rejects disturbances online even using inexpensive hardware for sensing and actuation. This result presents an opportunity for real-time optimal control of automation platforms with finite set of control signals.

## 2.1. Introduction

It is common in the automation industry that control authority is not continuous, for example, due to the discontinuous characteristics of actuators operated by valves. Optimal mode scheduling—model-based control of both the sequence and timing of operating modes—is a natural and efficient way of approaching these discrete[1] problems, with an abundance of algorithms proposed by the research community (e.g. [**24**, **25**] and more—see below). However, despite the common use of model predictive control (MPC, in the sense of receding horizon control as defined in Section 2.3.2) for continuous control problems, mode scheduling algorithms are rarely used in MPC schemes for real-time control in discrete control setting—in fact, the discontinuous components of automation platforms are sometimes disregarded to allow for application of continuous dynamic solutions (e.g. [**26**]). The main reasons for this lack of applicability are prohibitive execution times and/or high approximation errors resulting from the use of specialized ODE solvers for numerical integration of hybrid differential equations. We aim to overcome this drawback, by considering the problem of real-time mode scheduling for an autonomous linear time-varying switched system to optimize a quadratic performance metric. In particular, the contributions of this thesis are: 1) The formulation of an open-loop mode scheduling algorithm (referred to as Single Integration Optimal Mode Scheduling—SIOMS) so that no differential equation needs to be solved for during optimization; 2) The formulation and experimental validation of receding-horizon SIOMS for real-time closed-loop mode scheduling so that a differential equation only needs to be integrated over a limited time interval—typically the time step of the receding-horizon window—rather than the full time horizon.

Switching control problems arise in a number of application domains in automation industry, such as robot locomotion [**27**], manufacturing production [**18**, **28**], power electronics [**29**], telecommunications [**30**] and air traffic management [**31**]. Several algorithmic methods have been proposed to deal with scheduling problems. Many approaches have relied on a bi-level hierarchical structure with only a subset of the design variables considered at each level [**28**, **32**].

[1]Here and for the rest of this thesis, "discrete" denotes a finite number of control settings and is not to be confused with discretization in time, unless otherwise noted.

Other proposed methods include: embedding methods [**25**], which relax, or embed, the integer constraint and find the optimal of the relaxed cost; relaxed dynamic programming [**33**, **34**] where complexity is reduced by relaxing optimality within pre-specified bounds; variants of gradient-descent methods [**35**, **36**]; and application-specific solutions [**37**].

The iterative projection-based approach as introduced in [**38**–**40**] forms the basis for the work in this thesis. The mode scheduling problem is formulated as an infinite-dimensional optimal control problem where the variables to be optimized are a set of functions of time constrained to the integers. For a projection-based method, the design variables are in an unconstrained space but the cost is computed on the projection of the design variables to the set of admissible switched system trajectories. In [**38**], an iterative optimization algorithm is synthesized that employs the Pontryagin Maximum Principle and a projection-based technique. We adapt this algorithm by taking advantage of the linearity of the dynamical system under concern. The specific case of linear switching control has been extensively investigated by others (see [**41**–**43**]). The approaches in [**44**, **45**] solve for a differential equation at each step of the iterative algorithm. Previous attempts to avoid the online integration of differential equations are limited to switching time optimization problems where the mode sequence is fixed [**44**–**46**].

In this thesis, we extend our work in [**16**] to present and experimentally evaluate a projection-based mode scheduling algorithm (SIOMS) where a single set of differential equations is solved offline, so that no additional simulation is required during the open-loop optimization routine. *These offline solutions to differential equations are independent of the mode sequence and switching times* in contrast to [**44**–**46**]. Moreover, no assumption about the time-variance of the modes is made. Therefore, SIOMS does not exclude many important linear systems, such as time-varying power systems [**47**], traffic models [**48**], and nonlinear systems linearized about a trajectory.

Here, our objective is to emphasize the importance of SIOMS as a tool for model-based mode scheduling in real-time applications (see [**28**,**49**], for example). Real-time implementation of switched system algorithms is often impractical due their dependence on numerical solutions of differential equations. Three main points are listed to support this argument. First, the use of ODE solvers often renders algorithm execution times prohibitive for real-time implementation [**50**, **51**]. Second, the solution approximation through discretization does not always guarantee consistency (see Definition 3.3.6 in [**52**]), and lastly, discontinuous differential equations require specialized event-based numerical techniques that are prone to approximation errors [**53**].

SIOMS overcomes the aforementioned issues by avoiding online simulations. As far as the first point is concerned, one of the strongest assets of the proposed algorithm is that its *timing behavior—i.e., the execution time of a single iteration of the optimization algorithm—is independent of the choice of ODE solver*; it only depends on the number of multiplications and inversions required for the calculation of the optimality condition. As a result, *SIOMS is fast* and intrinsically free of the common trade-off between execution time and approximation accuracy that normally dictates the selection of numerical integration technique. Furthermore, authors in [**51**] address the second issue by proposing a time discretization that guarantees consistency for nonlinear systems. In this work, by restricting our focus to linear time-varying systems, we introduce a method where *approximation accuracy and consistency are independent of the number of samples* used for approximation of the state and co-state trajectories. Lastly, to address the third point, SIOMS only requires offline integration of differential equations that are continuous and as smooth as each of the linear modes. Thus, our algorithm exhibits *robustness* *to numerical errors due to discontinuous vector fields*. All the above SIOMS advantages are verified through a method-comparison numerical study in Section 2.4.

Exploiting the aforementioned computational and timing advantages, we can use SIOMS for real-time control applications by means of a receding-horizon synthesis [**54**]. Importantly, in receding-horizon optimization with SIOMS, a simple update step removes the need for numerical integration over the full time horizon in between consecutive algorithm runs—only integration over a few time steps is required. Although stability analysis of closed-loop SIOMS is not provided in this thesis, we include a short discussion on stability requirements based on established results in Section 2.3.2.

Finally, we choose to experimentally validate SIOMS real-time implementability by regulating the swing angle of a mobile robot and suspended mass system, online, using a finite number of control actions (i.e. switched system modes). For additional complexity, the string holding the mass exhibits a pre-defined time-varying length. Although not intrinsically a switched system, our example resembles many systems that admit hybrid input by construction and thus are difficult to control (e.g. antilock braking systems (ABS) [**55**], tanks [**56**] and other valve-operated systems). Moreover, despite the fact that conventional real-time control of variants of this system has been extensively studied [**57–59**]), we are interested in showing how a limited number of actions may suffice for control, even with time-varying parameters; a result that opens a discussion for alternative, inexpensive actuation and sensing solutions in seemingly complex control platforms. Our experimental work—based on the Robot Operating System (ROS)—demonstrates that closed-loop SIOMS regulates the example system reliably in real time, while rejecting disturbances.

## 2.2. Review

### 2.2.1. Switched Systems

Switched systems are a class of hybrid systems [**60**,**61**] that evolve according to one of $N$ vector fields (modes) $f_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, ..., N\}$ at any time over the finite time interval $[T_0, T_M]$, where $T_0$ is the initial time and $T_M > 0$ is the final time. We consider two representations of the switched system, namely mode schedule and switching control. As a unique mapping exists between each representation [**38**], the two will be used interchangeably throughout the chapter.

**Definition 1** The *mode schedule* is defined as the pair $\{\Sigma, \mathcal{T}\}$ where $\Sigma = \{\sigma_1, ..., \sigma_M\}$ is the sequence of active modes $\sigma_i \in \{1, ..., N\}$ and $\mathcal{T} = \{T_1, ..., T_{M-1}\}$ is the set of the switching times $T_i \in [T_0, T_M]$. The total number of modes in the mode sequence—which may vary across optimization iterations—is $M \in \mathbb{Z}^+$.

**Definition 2** A *switching control* corresponds to a list of curves $u = [u_1, ..., u_N]^T$ composed of $N$ piecewise constant functions of time, one for each different mode $f_i$. For all $t \in [T_0, T_M], \sum_{i=1}^N u_i(t) = 1$, and for all $i \in \{1, ..., N\}, u_i(t) \in \{0, 1\}$. This dictates that the state evolves according to only one mode for all time. We represent the set of all admissible switching controls as $\Omega$.

We will refer to the mode schedule corresponding to the switching control $u$ as $\{\Sigma(u), \mathcal{T}(u)\}$.

For a system with $n$ states $x = [x_1, ..., x_n]^T$ and $N$ different modes, the state equations are given by

$$\dot{x}(t) = F(t, x(t), u(t)) := \sum_{i=1}^{N} u_i(t) f_i(x(t), t) \tag{2.1}$$

subject to the initial condition $x(T_0) = x_0$. For this chapter, we restrict our focus to linear time-varying systems so that

$$F(t, x(t), u(t)) := \sum_{i=1}^{N} u_i(t)A_i(t)x(t). \tag{2.2}$$

Alternatively, we may express the system dynamics with respect to the current mode schedule as follows:

$$F(t, x(t), \Sigma, \mathcal{T}) := \overline{A}(t, \Sigma, \mathcal{T})x(t) \tag{2.3}$$

where $\overline{A}(t, \Sigma, \mathcal{T}) = A_{\sigma_i}(t)$ for $T_{i-1} \leq t < T_i$.

### 2.2.2. Problem Statement

Our objective is the minimization of a quadratic cost function

$$J(x, u) = \int_{T_0}^{T_M} \frac{1}{2}x(\tau)^T Q(\tau)x(\tau)d\tau + \frac{1}{2}x(T_M)^T P_1 x(T_M) \tag{2.4}$$

where $x$ is the state, $u$ the switching control and the pair $(x, u)$ satisfy (2.1). Here, $Q$ and $P_1$ are the running and terminal cost respectively, and are both symmetric positive semi-definite. Note that this cost functional can also be adapted to include reference trajectory, in which case the objective would be to minimize the error between the state and the reference ( [**46**]).

### 2.2.3. Projection-based Optimization

From Definition 2 of an admissible switching control $u$, it follows that our optimization problem is subject to an integer constraint [**38**]. Let $\mathcal{S}$ represent the set of all pairs of admissible state and switching control trajectories $(x, u)$, i.e. all pairs that satisfy the constraint (2.1) and are consistent with Definition 2 so that $u \in \Omega$. In [**39**], the authors propose a projection-based technique for handling these constraints set by $\mathcal{S}$. In particular, an equivalent problem is considered where the design variables $(\alpha, \mu)$ belong to an unconstrained set $(\mathcal{X}, \mathcal{U})$ and the cost $J$ is evaluated on the projection of these variables to the set $\mathcal{S}$. Now, the problem is reformulated as

$$\min_{(\alpha,\mu)} J(\mathcal{P}(\alpha, \mu)) \tag{2.5}$$

where $\mathcal{P}$ is a projection—with $\mathcal{P}(\mathcal{P}(\alpha, \mu)) = (\mathcal{P}(\alpha, \mu))$—that maps curves from the unconstrained set $(\mathcal{X}, \mathcal{U})$ to the set of admissible switched systems $\mathcal{S}$. As the cost is calculated on the admissible projected trajectories, this problem is equivalent to the original problem described in 2.2.2 and (2.4).

The optimal mode scheduling algorithm developed in [**38**] utilizes the max-projection operator. The max-projection operator $\mathcal{P} : \mathcal{X} \times \mathcal{U} \to \mathcal{S}$ at time $t \in [T_0, T_M]$ is defined as

$$\mathcal{P}(\alpha(t), \mu(t)) := \begin{cases} \dot{x}(t) = F(t, x(t), u(t)), \qquad x(T_0) = x_0 \\ u(t) = \mathcal{Q}(\mu(t)) \end{cases} \tag{2.6}$$

where $\mathcal{Q}$ is a mapping from a list of $N$ real-valued control trajectories, $\mu(\cdot) = [\mu_1(\cdot), ..., \mu_N(\cdot)]^T$ $\in \mathbb{R}^N$ to a list of $N$ feasible switching controls, $u \in \Omega$. We define $\mathcal{Q}$ as

$$\mathcal{Q}(\mu(t)) = \begin{bmatrix} \mathcal{Q}_1(\mu(t)) \\ \vdots \\ \mathcal{Q}_N(\mu(t)) \end{bmatrix} \quad \text{with} \quad \mathcal{Q}_i(\mu(t)) := \prod_{j \neq i}^{N} \mathbf{1}(\mu_i(t) - \mu_j(t)) \tag{2.7}$$

where $\mathbf{1} : \mathbb{R} \to \{0, 1\}$ is the step function given by

$$\mathbf{1}(t) = \begin{cases} 1, & t \geq 0 \\ 0, & else. \end{cases} \tag{2.8}$$

Notice that the max-projection operator does not depend on the unconstrained state trajectories $\alpha(\cdot)$. The unconstrained state $\alpha$ is included in the left hand side of the definition in order for $\mathcal{P}$ to be a projection.

### 2.2.4. Mode Insertion Gradient

The *mode insertion gradient* appears in previous studies [**24**, **62**, **63**]. Here, it is defined as the list of functions $d = [d_1(t), ..., d_N(t)] \in \mathbb{R}^N$ that calculate the sensitivity of cost $J$ to inserting one of the $N$ modes at some time $t$ for an infinitesimal interval (i.e. $\frac{dJ}{d\lambda^+}$ as $\lambda^+ \to 0$). Each element of $d$ is given by:

$$d_i(t) := \rho(t)^T (f_i(x(t), t) - f_{\sigma(t)}(x(t), t)), \quad i = 1, ..., N \tag{2.9}$$

where $x \in \mathbb{R}^n$ is the solution to the state equations (2.1) for all $t \in [T_0, T_M]$ and $\rho \in \mathbb{R}^n$, the co-state, is the solution to the adjoint equation[2]

$$\dot{\rho}(t) = -D_x F(t, x(t), u(t))^T \rho(t) - Q(t)x(t), \tag{2.10}$$

for all $t \in [T_0, T_M]$ subject to $\rho(T_M) = P_1 x(T_M)$. (In (2.9), $\sigma(t) : [T_0, T_M] \to \{1, ..., N\}$ is the function that returns the active mode at any time $t$.)

[2]$D_x f(\cdot)$ denotes the partial derivative $\frac{\partial f(\cdot)}{\partial x}$.

It has been shown in [**64**], that when a quadratic cost is optimized subject to a linear time-varying switched system, a linear mapping between state $x$ and co-state $\rho$ exists. Thus, we may express the co-state as

$$\rho(t) = P(t)x(t) \tag{2.11}$$

where $P(t) \in \mathbb{R}^{n\times n}$ is calculated by the following differential equation:

$$\dot{P}(t) = -\overline{A}(t,\Sigma,\mathcal{T})^T P(t) - P(t)\overline{A}(t,\Sigma,\mathcal{T}) - Q(t) \tag{2.12}$$

subject to $P(T_M) = P_1$. Note that this is the linear switched system analog to the Riccati equation from the LQR problem in classical control theory [**65**]. Using (2.2) and (2.11), the mode insertion gradient element can be written as

$$d_i(t) := x(t)^T P(t)^T [A_i(t) - A_{\sigma(t)}(t)]x(t). \tag{2.13}$$

### 2.2.5. Iterative Optimization

To calculate the switching control $u(t)$ that optimizes the quadratic performance metric (2.4), we follow an iterative approach. Iterative optimization computes a new estimate of the optimum by taking a step from the current estimate in a search direction so that a sufficient decrease in cost is achieved [**35, 38, 63, 66**]. A single iteration is commonly structured in the following scheme: Given a current iterate, i) Calculate a descent direction; ii) Calculate a step size; iii) Update the current iterate by taking a step in the descent direction. The procedure is repeated until a terminating condition is satisfied.

In the following section, we formulate an iterative projection-based algorithm for quadratic optimization of linear time-varying switched systems that requires no online simulations.

```
Algorithm 1 SIOMS

                                   Offline:
  ■ Solve for the STM $\Phi^j(t, T_0)$ and ATM $\Psi^j(t, T_M)$ $\forall j \in \{1, ..., N\}$ and $t \in [T_0, T_M]$.
  ■ Choose initial $u^0 \rightarrow \{\Sigma(u^0), \mathcal{T}(u^0)\}$.
  ■ Set $x(T_0) = x_0$ and $P(T_M) = P_1$.

                           Online iterative process:
                             Set $k = 0$, $u^k = u^0$.
  (1) Evaluate $x^k(t) := \chi(t, \Sigma(u^k), \mathcal{T}(u^k))$ as in Eq. (2.15).
  (2) Evaluate $P^k(t) := \varrho(t, \Sigma(u^k), \mathcal{T}(u^k))$ as in Eq. (2.21).
  (3) Evaluate the descent direction $-d^k(t)$ as in Eq. (2.28).
  (4) Calculate step size $\gamma^k$ by backtracking.
  (5) Update: $u^{k+1}(t) = Q(u^k(t) - \gamma^k d^k(t))$.
  (6) If $u^{k+1}$ satisfies a terminating condition, then exit, else, increment $k$ and repeat from step 1.
```

## 2.3. Single Integration Optimal Mode Scheduling

### 2.3.1. Open Loop Control over Finite Time Horizon

The problem of optimizing an arbitrary cost functional $J(x, u)$ subject to the switching control $u(t)$ and switching system state $x(t)$ is considered in [**38**]. Here, we increase the computational performance of [**38**] in the special case of linear time-varying systems with quadratic performance metric. In particular, we reformulate this problem so that no differential equations are solved during the iterative optimization routine. Algorithm 1 provides a summary of SIOMS.

Consider the optimization problem constrained by the system dynamics (2.3), as described in Section 2.2. The dynamic constraint dictates that a system simulation should be performed at each iteration in Algorithm 1 as soon as the next switching control has been calculated. In particular, the calculation of the mode insertion gradient (2.9) involves the solution of the state and adjoint equations, (2.3) and (2.10), while the max-projection operator also includes the state equation (2.3).

We follow a similar approach to the switching time optimization approach in [**46**], extending it to the situation where the mode sequence $\Sigma$ is unknown. Building on the existence of a linear relationship between the state and co-state as described in Section 2.2.4, we utilize operators to formulate algebraic expressions for the calculation of the state $x(t)$ and the relation $P(t)$ at any time $t \in [T_0, T_M]$. The operators are available prior to optimization through offline solutions to differential equations. Moreover, they are independent of the mode sequence and switching times.

In the switching time optimization case [**46**]—where the mode sequence is constant and the problem is finite-dimensional—a single optimization iteration involves only a finite number of state and co-state evaluations; these occur at the (finite) switching times for that particular iteration. However, mode scheduling is an infinite-dimensional optimal control problem and requires the time evolution of the state and co-state trajectories at each iteration.

Therefore, in order for the proposed algorithm to be feasible, an explicit mapping from time $t$ to $x$ and $P$ is needed at each iteration, depending on the current mode schedule $\{\Sigma, \mathcal{T}\}$. The mapping, below in (2.15) and (2.21), only includes algebraic expressions dependent on solutions to pre-computed differential equations. The exact number of multiplications executed in each iteration depends on how many time instances the state and co-state must be evaluated.

For the rest of the chapter, a variable with the superscript $k$ implies that the variable depends directly on $u^k$, the switching control at the $k^{\text{th}}$ algorithm iteration.

Evaluating $x(t)$. The operators for evaluating $x(t)$ are the state-transition matrices (STM) of the $N$ modes. Let $\Phi^j(\cdot, T_0) : \mathbb{R} \to \mathbb{R}^{n \times n}$ denote the STM for the linear mode $j \in \{1, ..., N\}$ with

$A_j(t)$. The STM are the solutions to the $N$ matrix differential equations

$$\frac{d}{dt}\Phi^j(t, T_0) = A_j(t) \cdot \Phi^j(t, T_0), \quad j = 1, ..., N \tag{2.14}$$

subject to the initial condition $\Phi^j(T_0, T_0) = I_n$. The following two STM properties are useful for computing the state $x(t)$ given a mode schedule $\{\Sigma, \mathcal{T}\}$. For an arbitrary STM, $\Phi$, characterized by $A(t)$, we have [**67**] :

(1) $x(t) = \Phi(t, \tau)x(\tau)$

(2) $\Phi(t_1, t_3) = \Phi(t_1, t_2)\Phi(t_2, t_3) = \Phi(t_1, t_2)\Phi(t_3, t_2)^{-1}$.

We emphasize the importance of Property 2 in that it allows us to use a single operator for the evaluation of the state as explained in the following.

**Proposition 1.** *The state $x(t)$ at all $t \in [T_0, T_M]$ depends on the mode schedule $\{\Sigma, \mathcal{T}\}$ and the STM $\Phi^j(\cdot, T_0)$ and is given by $x(t) := \chi(t, \Sigma, \mathcal{T})$ where*

$$\begin{aligned}\chi(t, \Sigma, \mathcal{T}) = \sum_{i=1}^{M} \Big\{ \Big[ \mathbf{1}(t - T_{i-1}) - \mathbf{1}(t - T_i) \Big] \\ \Phi^{\sigma_i}(t, T_0)\Phi^{\sigma_i}(T_{i-1}, T_0)^{-1} x(T_{i-1}) \Big\} \\ \textit{subject to } x(T_0) = x_0,\end{aligned} \tag{2.15}$$

$\mathbf{1}(\cdot)$ *is the step function defined in* (2.8) *and $T_i$ , $\sigma_i$ are the $i^{th}$ switching time and corresponding active mode as defined in Section 2.2.1.*

**Proof.** Using the STM properties 1 and 2, the state $x$ at the $i$th switching time is

$$x(T_i) = \overline{\Phi}(T_i, T_0)x_0 = \left[ \prod_{j=i}^{1} \Phi^{\sigma_j}(T_j, T_{j-1}) \right] x_0 \tag{2.16}$$

where $\overline{\Phi}(T_i, T_0)$ is the state-transition matrix corresponding to $\overline{A}(t, \Sigma, \mathcal{T})$ as defined in (2.3). Hence, the state evolution is defined as a piecewise function of time, each piece corresponding to a time interval between consecutive switching times $\{T_i, T_{i+1}\}$:

$$x(t) = \begin{cases} \Phi^{\sigma_1}(t, T_0)x(T_0), & T_0 \le t < T_1 \\ \Phi^{\sigma_2}(t, T_1)\Phi^{\sigma_1}(T_1, T_0)x(T_0), & T_1 \le t < T_2 \\ \vdots & \vdots \\ \Phi^{\sigma_M}(t, T_{M-1})[\prod_{j=M-1}^{1} \Phi^{\sigma_j}(T_j, T_{j-1})]x(T_0) & T_{M-1} \le t \le T_M \end{cases} \tag{2.17}$$

For a more compact representation of the state, we employ unit step functions and (2.16) to get

$$x(t) = \sum_{i=1}^{M} \left\{[\mathbf{1}(t - T_{i-1}) - \mathbf{1}(t - T_i)]\Phi^{\sigma_i}(t, T_{i-1})x(T_{i-1})\right\} \tag{2.18}$$

where, from STM property 2,

$$\Phi^{\sigma_i}(t, T_{i-1}) = \Phi^{\sigma_i}(t, T_0)\Phi^{\sigma_i}(T_{i-1}, T_0)^{-1}. \tag{2.19}$$

This concludes the proof. □

Prior to the iterative optimization, the STM operators $\Phi^j(t, T_0)$ are solved offline for $t \in [T_0, T_M]$ and for all different modes $j = 1, ..., N$. Thus, given a mode schedule, the calculation of state $x(t)$ via (2.15) requires no additional integrations beyond the offline calculations used for $\Phi^j(t, T_0)$.

Evaluating $P(t)$. As proven in [**46**], an analogous operator to the STM exists for the evaluation of the relation $P(t)$ appearing in (2.11). As in [**46**], we will refer to the operator as the adjoint-transition matrix (ATM) and use $\Psi^j(\cdot, T_M) : \mathbb{R} \to \mathbb{R}^{n\times n}$ to denote the ATM corresponding to each mode $j \in \{1, ..., N\}$. The ATM are defined to be the solutions to the following $N$ matrix differential equations:

$$\frac{d}{dt}\Psi^j(t, T_M) = -A_j(t)^T\Psi^j(t, T_M) - \Psi^j(t, T_M)A_j(t) - Q(t) \tag{2.20}$$

subject to the initial condition $\Psi^j(T_M, T_M) = 0_{n\times n}$.

The following two ATM properties will be useful for evaluating $P(t)$ given a mode schedule $\{\Sigma, \mathcal{T}\}$. For an arbitrary ATM, $\Psi$, characterized by $A(t)$ and associated STM $\Phi$, and cost function defined by $Q(t)$, we have [**46**]:

(1) $P(t) = \Psi(t, \tau) \circ P(\tau) := \Psi(t, \tau) + \Phi(\tau, t)^T P(\tau)\Phi(\tau, t)$
(2) $\Psi(t_1, t_3) = \Psi(t_1, t_2) \circ \Psi(t_2, t_3) := \Psi(t_1, t_2) + \Phi(t_2, t_1)^T\Psi(t_2, t_3)\Phi(t_2, t_1)$.

Notice that Property 2 of ATM is equivalent to Property 2 of STM and similarly allows us to evaluate the co-state.

**Proposition 2.** *The relation $P(t)$ at all $t \in [T_0, T_M]$ depends on the current mode schedule $\{\Sigma, \mathcal{T}\}$, the STM $\Phi^j(\cdot, T_0)$ and the ATM $\Psi^j(\cdot, T_M)$ and is given by $P(t) := \varrho(t, \Sigma, \mathcal{T})$ where*

$$\begin{aligned}\varrho(t, \Sigma, \mathcal{T}) = \sum_{i=1}^{M} \Big\{ \Big[&\mathbf{1}(t - T_{i-1}) - \mathbf{1}(t - T_i)\Big]\cdot \\ \Big[&\Psi^{\sigma_i}(t, T_M) + \Phi^{\sigma_i}(t, T_0)^{-T}\Phi^{\sigma_i}(T_i, T_0)^T[P(T_i) \\ &-\Psi^{\sigma_i}(T_i, T_M)]\Phi^{\sigma_i}(T_i, T_0)\Phi^{\sigma_i}(t, T_0)^{-1}\Big]\Big\}\end{aligned} \tag{2.21}$$

$$\textit{subject to } P(T_M) = P_1,$$

$\mathbf{1}(\cdot)$ *is the step function defined in* (2.8) *and* $T_i$ , $\sigma_i$ *are the* $i^{th}$ *switching time and corresponding active mode as defined in Section 2.2.1.*

**Proof.** From the ATM properties 1 and 2, $P(t)$ at the $i^{\text{th}}$ switching time is

$$\begin{aligned} P(T_i) &= \overline{\Psi}(T_i, T_M) \circ P(T_M) \\ &= \overline{\Psi}(T_i, T_M) + \overline{\Phi}(T_M, T_i)^T P(T_M) \overline{\Phi}(T_M, T_i) \end{aligned} \tag{2.22}$$

where $\overline{\Psi}(T_i, T_M)$ is the adjoint-transition matrix corresponding to $\overline{A}(t, \Sigma, \mathcal{T})$ as defined above. From ATM property 2, this is equal to

$$\begin{aligned} \overline{\Psi}(T_i, T_M) &= \Psi^{\sigma_{i+1}}(T_i, T_{i+1}) \circ \cdots \circ \Psi^{\sigma_M}(T_{M-1}, T_M) \\ &= \sum_{m=i+1}^{M} \overline{\Phi}(T_{m-1}, T_i)^T \Psi^{\sigma_m}(T_{m-1}, T_m) \overline{\Phi}(T_{m-1}, T_i). \end{aligned} \tag{2.23}$$

As in the previous case, we aim to derive an expression for the evaluation of $P(t)$ at arbitrary time instances. Again, we will represent $P(t)$ as a piecewise function of time:

$$P(t) = \begin{cases} \Psi^{\sigma_M}(t, T_M) \circ P(T_M), & T_{M-1} \le t < T_M \\ \Psi^{\sigma_{M-1}}(t, T_{M-1}) \circ P(T_{M-1}), & T_{M-2} \le t < T_{M-1} \\ \quad\vdots & \quad\vdots \\ \Psi^{\sigma_1}(t, T_1) \circ P(T_1), & T_0 \le t < T_1 \end{cases} \tag{2.24}$$

For a more compact representation of $P(t)$, we employ unit step functions to get

$$P(t) = \sum_{i=1}^{M} \left\{ [\mathbf{1}(t - T_{i-1}) - \mathbf{1}(t - T_i)][\Psi^{\sigma_i}(t, T_i) \circ P(T_i)] \right\} \tag{2.25}$$

where, from ATM property 2,

$$\Psi^{\sigma_i}(t, T_i) = \Psi^{\sigma_i}(t, T_M) + \Phi^{\sigma_i}(T_i, t)^T \Psi^{\sigma_i}(T_i, T_M)\Phi^{\sigma_i}(T_i, t). \tag{2.26}$$

Combining ATM property 1 with (2.21) and (2.26), we end up with the expression

$$\begin{aligned} P(t) = \sum_{i=1}^{M} \Big\{ &[\mathbf{1}(t - T_{i-1}) - \mathbf{1}(t - T_i)] \cdot \\ &[\Psi^{\sigma_i}(t, T_M) + \Phi^{\sigma_i}(T_i, t)^T [P(T_i) - \Psi^{\sigma_i}(T_i, T_M)]\Phi^{\sigma_i}(T_i, t)] \Big\} \end{aligned} \tag{2.27}$$

with $\Phi^{\sigma_i}(T_i, t) = \Phi^{\sigma_i}(T_i, T_0)\Phi^{\sigma_i}(t, T_0)^{-1}$. This completes the proof. □

Prior to the iterative optimization, the ATM operators $\Psi^j(t, T_M)$ are solved offline for all $t \in [T_0, T_M]$ and for all different modes $j = 1, ..., N$. Thus, given a mode schedule, the calculation of $P(t)$ via (2.21) requires no additional integrations.

Calculating the descent direction using the mode insertion gradient. An iterative optimization method computes a new estimate of the optimum by taking a step in a search direction from the current estimate of the optimum so that a sufficient decrease in cost is achieved. The mode insertion gradient $d(t)$ defined above, has a similar role in the mode scheduling optimization as the gradient does for finite-dimensional optimization. It has been shown in [**38**, **62**, **63**] that $-d^k(t)$ is a descent direction.

**Proposition 3.** *An element of $d^k(t)$ is given by*

$$\begin{aligned} d_i^k(t) := &\chi(t, \Sigma(u^k), \mathcal{T}(u^k))^T \varrho(t, \Sigma(u^k), \mathcal{T}(u^k)) \\ &[A_i(t) - A_{\sigma^k(t)}(t)]\chi(t, \Sigma(u^k), \mathcal{T}(u^k)) \end{aligned} \tag{2.28}$$

*where $i = \{1, ..., N\}$.*

**Proof.** After the definition for the state and co-state, an equivalent expression for the mode insertion gradient may be obtained from (2.15),(2.21) and (2.9). □

Update rule. A new estimate of the optimal switching control $u^{k+1}$ is obtained by varying from the current iterate $u^k$ in the descent direction and projecting the result to the set of admissible switching control trajectories. For this purpose, we employ the max-projection operator (2.6) and get a new estimate of the optimum,

$$\begin{aligned} u^{k+1}(t) &= Q(u^k(t) - \gamma^k d^k(t)) \\ x^{k+1}(t) &:= \chi(t, \Sigma(u^{k+1}), \mathcal{T}(u^{k+1})) \end{aligned} \tag{2.29}$$

where $Q$ is given by (2.7). For choosing a sufficient step size $\gamma^k$, we may utilize a projection-based backtracking process as described in [**40**].

The reader is referred to [**38**, **39**] for a more detailed description of these algorithm steps, along with the associated proofs for convergence.

Calculating the optimality condition. The optimality function $\theta^k \in \mathbb{R}$ is [**38**]

$$\theta^k := d^k_{i_0}(t_0) \tag{2.30}$$

where

$$(i_0, t_0) = \min_{i\in\{1,\ldots,N\}, t\in[T_0,T_M]} d_i(t). \tag{2.31}$$

The limit of the sequence of optimality functions is proven to go to zero as a function of iteration $k$ in [**38**]. This allows us to utilize $\theta^k$ also as a terminating condition for the iterative algorithm.

### 2.3.2. A Receding-Horizon Approach

Section 2.3.1 provides an offline approach for computing an open-loop optimizer for the problem in Section 2.2.2. Here, we follow a receding-horizon approach in order to achieve closed-loop optimization over an infinite time horizon.

Receding-horizon control strategies (often referred to as MPC strategies [**33**, **54**, **68**, **69**]) have become quite popular recently, partly due to their robustness to model uncertainties or to sensor measurement noise. This chapter's approach enables real-time closed-loop execution of finite-horizon optimal control algorithms. Based on our performance evaluation in the next section, the finite-horizon SIOMS is well-suited for receding-horizon linear switched-system control because it is fast and accurate.

A receding-horizon scheme for optimal mode scheduling can be implemented as follows. From the current time $t$ and measured state $x(t)$ as the initial condition in (2.1), use SIOMS to obtain an optimal switching control $u_t(\tau)$ for $\tau \in [t, t+T]$ where $T := (T_M - T_0)$ in Algorithm

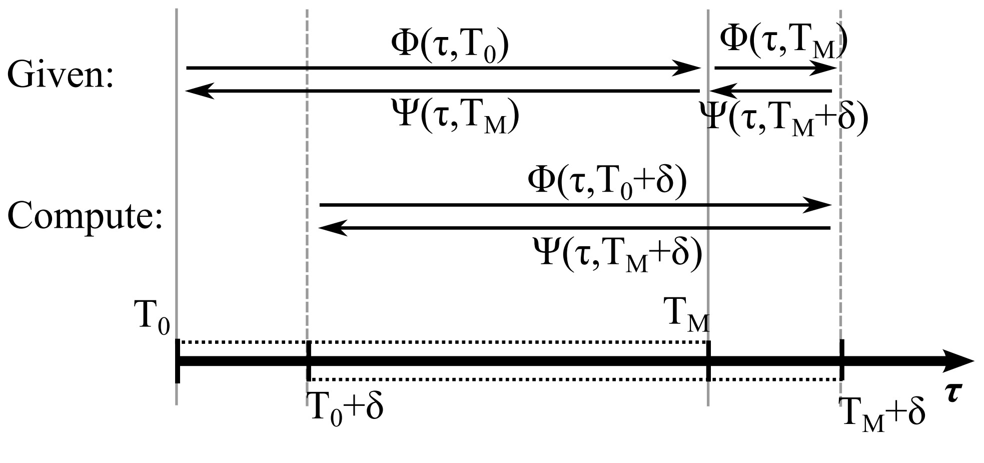


Figure 2.1. An illustration of the operators update step in a receding-horizon scheme. A differential equation needs to be integrated only over a limited time interval $\delta$ rather than the time horizon $(T_M - T_0) := T$.

1. Apply the calculated control for time duration $\delta$ with $0 < \delta \leq T$ to drive the system from $x(t)$ at time $t$ to $x(t+\delta)$. Set $t \leftarrow t + \delta$ and repeat. This scheme requires execution of the optimal mode scheduling algorithm every $\delta$ seconds.

Following Algorithm 1, SIOMS requires an offline calculation of operators before the online iterative process is executed. However, in order for SIOMS to be efficient in a receding-horizon approach, it is undesirable to recalculate each STM and ATM every $\delta$ seconds for the next $T$ seconds. Instead, each STM and ATM of the previous time interval $[T_0, T_M]$ are updated for the new information on $[T_M, T_M + \delta]$ only (Fig. 2.1). Such an approach is feasible because of the following lemma.

**Lemma 1.** *Suppose $\Phi(t, T_0)$ and $\Psi(t, T_M)$ are known for all $t \in [T_0, T_M]$. Assuming also that $\Phi(t, T_M)$ and $\Psi(t, T'_M)$ are known for all $t \in [T_M, T'_M]$, the STM and ATM for the time interval $t \in [T'_0, T'_M]$ with $T_0 < T'_0$ and $T_M < T'_M$ are given by*

$$\Phi(t, T'_0) = \begin{cases} \Phi(t, T_0)\Phi(T'_0, T_0)^{-1}, & T'_0 \leq t < T_M \\ \Phi(t, T_M)\Phi(T_M, T_0)\Phi(T'_0, T_0)^{-1}, & T_M \leq t \leq T'_M \end{cases} \tag{2.32}$$

*and*

$$\Psi(t, T'_M) = \begin{cases} \Psi(t, T_M) \circ \Psi(T_M, T'_M), & T'_0 \leq t < T_M \\ \Psi(t, T'_M), & T_M \leq t \leq T'_M. \end{cases} \tag{2.33}$$

**Proof.** The proof of Lemma 1 is a straightforward consequence of STM property 2 and ATM properties 1 and 2 in Section 2.3.1. □

Despite its simplicity, Lemma 1 is the key to efficient real-time execution of a receding-horizon hybrid control scheme. Using Lemma 1 with $T'_0 = T_0 + \delta$ and $T'_M = T_M + \delta$, we formulate Algorithm 2 that allows for real-time closed-loop SIOMS execution. The proposed formulation requires a numerical integration over the limited time interval $\delta$ rather than the full time horizon $T$ (step 3.1 in Algorithm 2). A graphical representation of the operators update every $\delta$ seconds (step 3 in Algorithm 2) is given in Fig. 2.1.

As mentioned in the introduction, this thesis does not address stability of the receding-horizon SIOMS algorithm. Many papers offer a detailed stability analysis for receding-horizon algorithms, e.g. [**54**,**70**,**71**], with only a few focusing on switched systems in particular [**72**,**73**]. The latter establish stability criteria that rely on hysteresis and dwell-time conditions. However, as is obvious from our main example in Section 2.5, we wish to give special consideration to applications where SIOMS is used for system control with hybrid inputs by expressing the problem in a switched system framework. The foundation of a stability proof for this SIOMS implementation is given in [**74**] where authors provide stability conditions on the design parameters of model predictive control algorithms with input discontinuities. In short, conditions are imposed on the time horizon $T$, the terminal and running cost in (2.4) and the terminal constraint set.

## 2.4. Open-Loop Implementation and Evaluation

In this section, SIOMS is implemented in a standard open-loop manner (see Algorithm 1) and its performance is evaluated in terms of i) execution time, ii) error of approximation and iii) computational complexity.

**Algorithm 2** Receding-Horizon SIOMS

```
■ Initialize current time $t$, finite horizon $T$ and control duration $\delta$.
■ Solve for $\Phi^j(\tau, t)$ and $\Psi^j(\tau, t + T)$ $\forall j \in \{1, ..., N\}$ and $\tau \in [t, t + T]$.
----------------------------------------------------------------------
Do every $\delta$ seconds while control $u_t(\tau)$ is applied:
1. Update $T_0 \leftarrow t$, $T_M \leftarrow t + T$ and set $x(T_0) = x(t)$.
2. Run online part of Algorithm 1 to get $u_t(\tau)$ for $\tau \in [t, t + \delta]$.
    3.1 Solve for $\Phi^j(\tau, T_M)$ and $\Psi^j(\tau, T_M + \delta)$ $\forall \tau \in [T_M, T_M + \delta]$. *
    3.2 Get $\Phi^j(\tau, T_0 + \delta)$ and $\Psi^j(\tau, T_M + \delta)$ $\forall j \in \{1, ..., N\}$ and $\tau \in [T_0 + \delta, T_M + \delta]$ from known $\Phi^j(\tau, T_0)$
        and $\Psi^j(\tau, T_M)$ using Lemma 1. *
    3.3 Update $\Phi(\tau, T_0) \leftarrow \Phi(\tau, T_0 + \delta)$ and $\Psi(\tau, T_M) \leftarrow \Psi(\tau, T_M + \delta)$. *
----------------------------------------------------------------------
* In a real-time application, step 3 can be executed at any time when processing requirements are low, without
increasing the amount of time needed for calculation of control (i.e. steps 1-2).
```

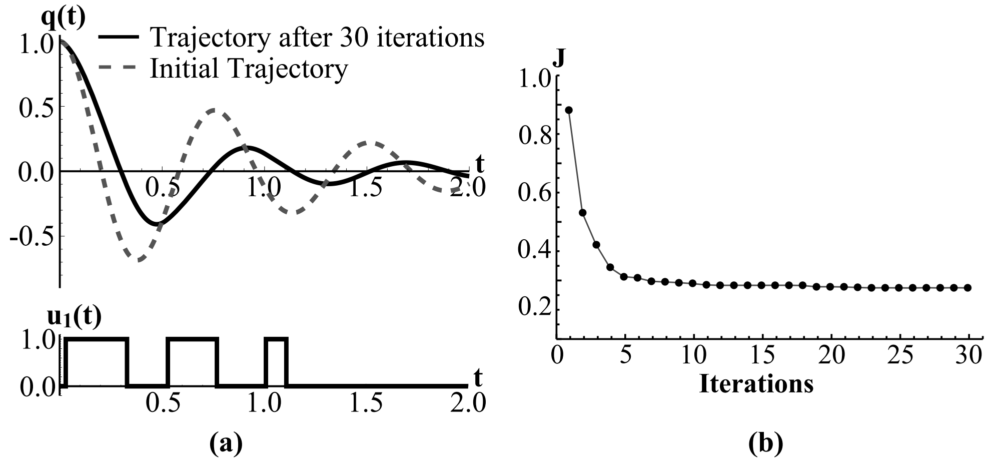


Figure 2.2. Spring-Mass-Damper vibration control: (a) Optimal trajectory and switching control and (b) the cost versus iteration count.

As a baseline example, we use SIOMS to apply switched stiffness vibration control on an unforced spring-mass-damper system. A linear time-invariant system is particularly suited for evaluation purposes as an analytical solution exists and can be compared with the computed numerical solution. Variants of this example system have been used extensively in literature for the evaluation of hybrid controllers [**75**, **76**]. Denoting by $k_i$ the variable spring stiffness and by

$m$ and $d$ the mass and damping coefficient, the system equations take the form in (2.2) with

$$A_i(t) = \begin{pmatrix} 0 & 1 \\ -\frac{k_i}{m} & -\frac{d}{m} \end{pmatrix} \tag{2.34}$$

and $N = 2$ i.e. two possible modes. The state vector is $x = [q(t), \dot{q}(t)]^T$, where $q(t)$ is the mass position. System parameters are defined as $m = 1$, $d = 2$, $k_1 = 30$, and $k_2 = 70$. Our objective is to find the mode schedule that minimizes the system vibration and is accordingly characterized by the quadratic cost functional (2.4) with $Q = diag[1, 0.1]$, $P_1 = 0_{2\times2}$ and $[T_0, T_M] = [0, 2]$. As an initial estimate $u^0(t)$, the system is in mode 2 with an initial condition $x_0 = [1, 0]^T$ and cost $J_0 \approx 0.98$.

Fig. 2.2a shows the optimal switching control and corresponding optimal $q(t)$ trajectory after 30 SIOMS iterations. The cost is reduced to $J \approx 0.38$ ( Fig. 2.2b).

### 2.4.1. Execution Time and Approximation Error

The execution time of iterative optimal control algorithms might be prohibitive for real-time applications [**51**]. It is often the case that appropriate numerical techniques for integrating the state and adjoint equations, (2.1) and (2.10), improve execution times. However, there is a trade-off to consider—a fast numerical ODE solver might be prone to approximation errors. In open-loop SIOMS (Algorithm 1), no differential equations need to be numerically solved as part of the online iterative process. Hence, we will show that both the online execution time and approximation error can be kept low at the same time.

Referring to Algorithm 1, a set of operator trajectories is pre-calculated and stored offline, covering the full time horizon $[T_0, T_M]$. In practice, the exact number of stored samples $\mathcal{N}$

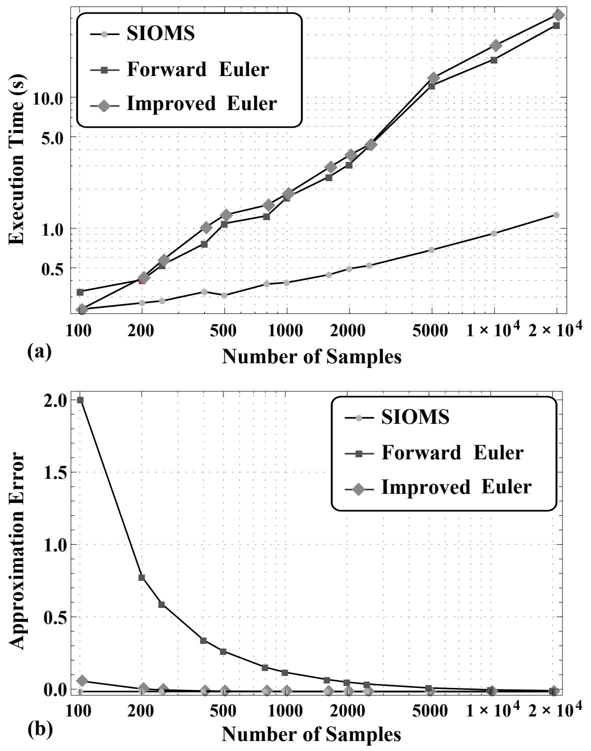


Figure 2.3. Variation of (a) online execution times and (b) approximation errors (2-norm of the root-mean-squared differences between the analytic and computed state values) with respect to the selected number of samples evaluated across 3 different optimization methods. SIOMS can achieve both objectives (i.e. fast execution and high approximation accuracy) for a wide range of sample sizes.

needs to be determined to reflect the processor's computational capacity and memory availability.[3] We use the mass-spring-damper to illustrate how the SIOMS execution time and error of approximation vary across different choices of sample sizes (Fig. 2.3).

For comparison purposes, we additionally evaluate the performance of the projection-based mode scheduling algorithm in [**38**] using the same example employing two different numerical techniques, namely the Forward and Improved Euler methods, for the integration of the state

[3]Interpolating methods may be used for intermediate time instances.

and adjoint equations. In contrast to SIOMS where expressions exist for the state and co-state evaluation ((2.15) and (2.21)), here, the solution to the state and co-state equations, (2.2) and (2.10), is approximated in every algorithm iteration. The Forward Euler method provides the following approximation to the state and co-state trajectories of a general linear time-varying switched system:

$$\begin{aligned} x(t_{h+1}) &= (\mathbb{I} + \Delta t \cdot \overline{A}(t_h, \Sigma, \mathcal{T}))x(t_h), \qquad x(t_0) = x_0 \\ \rho(t_h) &= (\mathbb{I} + \Delta t \cdot \overline{A}(t_{h+1}, \Sigma, \mathcal{T}))^T \rho(t_{h+1}) + \Delta t \cdot Qx(t_{h+1}), \\ &\qquad\qquad \rho(t_N) = P_1 x(t_N) \end{aligned} \tag{2.35}$$

where $\mathbb{I}$ is the $n \times n$ identity matrix, $t_{h+1} = t_h + \Delta t$ with $\Delta t$ the step size and $\overline{A}(t, \Sigma, \mathcal{T})$ is defined in (2.3). The Euler method is simple but can be unstable and inaccurate. On the other hand, Improved Euler (i.e. two-stage Runge Kutta) maintains simplicity but with reduced approximation errors. It applies the following approximation:

$$\begin{aligned} x(t_{h+1}) &= \Big[\mathbb{I} + \frac{\Delta t}{2} A(t_h) + \frac{\Delta t}{2} A(t_{h+1})(\mathbb{I} + \Delta t \cdot A(t_h))\Big] x(t_h), \qquad x(t_0) = x_0 \\ \rho(t_h) &= \Big[\mathbb{I} + \frac{\Delta t}{2} A(t_{h+1})^T + \frac{\Delta t}{2} A(t_h)^T (\mathbb{I} + \Delta t \cdot A(t_{h+1})^T)\Big] \rho(t_{h+1}) \\ &\quad + \Delta t(\mathbb{I} + \frac{\Delta t}{2} A(t_h)^T) Qx(t_{h+1}), \qquad \rho(t_N) = P_1 x(t_N) \end{aligned} \tag{2.36}$$

where $\mathbb{I}$ is the $n \times n$ identity matrix and $t_{h+1} = t_h + \Delta t$ with $\Delta t$ the step size. It is assumed for notational simplicity that $A(\cdot) := \overline{A}(\cdot, \Sigma, \mathcal{T})$ defined in (2.3). Notice that both approximation methods for the state and co-state, (2.35) and (2.36), depend on the step size $\Delta t$ as opposed to SIOMS state and co-state expressions (2.15) and (2.21) that are independent of a step size.

In the following example, the execution time and error of approximation are measured against the selected number of samples $\mathcal{N}$. The step size $\Delta t$ is constant so that the samples are evenly-spaced. For the Forward and Improved Euler methods, the number of samples $\mathcal{N}$ determines the fixed step size $\Delta t$ used for online integration of (2.1) and (2.10) resulting in the approximations (2.35) and (2.36). However, in SIOMS the number of samples $\mathcal{N}$ does not determine the step size used in the offline numerical integration—instead, the STM and ATM equations, (2.14) and (2.20), are numerically solved[4] and the resulting trajectories are sub-sampled with the desired sampling frequency $1/\Delta t$ to create the final stored data points. Note that we are only able to perform this additional sub-sampling interpolation because it does not affect the total execution time of the online algorithm portion. The fact that the sub-sampling process is applied on smooth trajectories produced by the continuous vector fields in (2.14) and (2.20)—along with the fact that the expressions for evaluating the state and co-state, (2.15) and (2.21), do not depend on any discretization step size—guarantees that approximation accuracy of each sample does not drop as the number of samples decreases.[5] Regardless of the particular choice of $\Delta t$, the role of $\Delta t$ has the same impact on all three representations of state and co-state evolution (SIOMS, Forward and Improved Euler)—in each case, $\Delta t$ determines the number of samples $\mathcal{N}$ (that can be) available (without interpolation) during each iteration. All methods were implemented in MATLAB, on a laptop with an Intel Core i7 chipset.

The results are summarized in Fig. 2.3. Figure 2.3a illustrates the variation of online execution time with respect to the selected number of samples. Execution time refers to the number of seconds required for 10 algorithm iterations—no significant change in cost is observed in subsequent iterations as seen in Fig. 2.2b. In all cases, the final optimal cost was found to be in

[4]For this example, equations (2.14) and (2.20) are solved by a fixed-step Improved Euler's method (i.e. two-stage Runge Kutta) with step size equal to $10^{-4}$.

[5]Choosing to use interpolating methods might be concerning with regard to approximation accuracy of the full state and co-state trajectories. Regardless, there are two ways to keep approximation errors low: i) by using higher-order interpolating methods and ii) by using a larger number of samples. With SIOMS, we can select a large number of samples without dramatically increasing the execution time (Fig. 2.3).

the range 0.45-0.5. Both Euler methods exhibit a similar rising trend with the execution time reaching a maximum of 13 seconds when 20,001 samples are used (i.e. step size of 0.0001 secs). With SIOMS, however, a significantly lower increase rate is observed with a maximum online execution time at only approximately 1.3 seconds. The reasoning for this observed difference is that with Euler methods, all samples of the state and co-state trajectories must be calculated in every iteration whereas in SIOMS one only needs to calculate the state and co-state values necessary for the procedures of the algorithm (e.g. computation of new switching times) using the expressions (2.15) and (2.26) respectively.

The variation of approximation error with respect to the number of samples is depicted in Fig. 2.3b. Here, by approximation error we refer to the 2-norm of the root-mean-squared (RMS) differences between the analytic and computed state values for all states at sample points. As explained earlier, the error with SIOMS remains approximately zero ($\approx$ 0.0002) regardless of the sample size. The trade-off between computation time and approximation error is particularly obvious with the Forward Euler's method, where the error only approaches zero when a maximum number of samples is employed by which time the corresponding execution time is prohibitive. Interestingly, Improved Euler's method starts with a lower error ($\approx$ 0.07) and drops to its minimum value of $\approx$ 0.0002 when 1600 samples and above are used. With the lowest approximation error ($\approx$ 0.0002), Improved Euler can achieve a minimum execution time of approximately 3 seconds compared to 0.2 seconds achieved by SIOMS. With low execution time ($\approx$ 0.2 with 100 samples used), Improved Euler can achieve a minimum error close to 0.1 compared to 0.0002 achieved by SIOMS.

### 2.4.2. Computational Complexity

In Section 2.3, we showed that all the state and co-state information needed in Algorithms 1 and 2, is encoded in the STM, $\Phi^j(t, T_0)$, and ATM, $\Psi^j(t, T_M)$, $\forall j \in \{1, ..., N\}$ which are solved for all $t \in [T_0, T_M]$ prior to the optimization routine. Therefore, the calculation of $x^k(t)$ and $P^k(t)$ and consequently the optimality condition $\theta^k$ relies simply on memory calls and matrix algebra. No additional differential equations need to be solved for during optimization.

The algorithm complexity can be discussed in terms of the number of matrix multiplications involved in each iteration. Recall that at each iteration, $x(t)$ is given by (2.15) and $P(t)$ by (2.21), but the total number of state and co-state evaluations depends on the number of time instances the descent direction (2.28) must be evaluated (e.g. for the calculation of $\theta^k$ in (2.31)). Taking this into consideration, we will look at the algebraic calculations required for the evaluation of the state, co-state and descent direction at a single time instance $t$.

First, for executional efficiency, one may calculate all the state and co-state values at the switching times, $x(T_i)$ and $P(T_i)$, given the current mode schedule $(\Sigma(u^k), \mathcal{T}(u^k))$ at the beginning of each iteration. To compute the state, begin with $x(T_0) = x_0$ and then recursively calculate

$$x(T_i) = \Phi^{\sigma_i}(T_i, T_{i-1})x(T_{i-1}) \forall i \in \{1, ..., M-1\}. \tag{2.37}$$

Using STM property 2 and following a similar approach as in the derivation of (2.15), this computation comes down to $2(M-1)$ matrix multiplications, assuming that all $\Phi^j(t, T_0)^{-1}$ for all $j \in \{1, ..., N\}$ have also been stored in memory. Similarly, begin with $P(T_M) = P_1$ and then recursively calculate

$$
\begin{aligned}
P(T_i) =& \Psi^{\sigma_{i+1}}(T_i, T_M) + \Phi^{\sigma_{i+1}}(T_{i+1}, T_i)^T \\
& [P(T_{i+1}) - \Psi^{\sigma_{i+1}}(T_{i+1}, T_M)]\Phi^{\sigma_{i+1}}(T_{i+1}, T_i)
\end{aligned} \tag{2.38}
$$

for all $i \in \{1, ..., M - 1\}$. Note that the derivation of the above expression is identical to the derivation of (2.21). Knowing that all $\Phi^{\sigma_{i+1}}(T_{i+1}, T_i)$ have already been calculated in (2.37), another $2(M - 1)$ multiplications are required for the calculation of $P(t)$. To summarize, the standard computational cost of the algorithm comes down to a total of $4(M - 1)$ multiplications per iteration.

Now, to evaluate equation (2.15) and (2.21) at any random time $t$ during the optimization process, we only need 6 additional multiplications, 2 for the state $x(t)$ and 4 for the relation $P(t)$. Therefore, to evaluate the descent direction at any random time, 9 multiplications are required in total, including the algebra involved in (2.28).

Finally, each iteration of Algorithm 1 involves $4(M - 1)$ multiplications for the calculation of $x(T_i)$ and $P(T_i)$, and $9\lambda$ additional multiplications where $\lambda$ is the number of evaluations of the expression (2.28) for the descent direction.

## 2.5. Closed-Loop Simulation and Experimental Implementation

In this section, SIOMS is implemented in a closed-loop manner (see Algorithm 2) and is tested on a cart and suspended mass system in simulation and on an experimental setup.

The system model under concern is linear time-varying with two configuration variables, $q(t) = [y(t), \zeta(t)]^T$, where $y(t)$ is the horizontal displacement of the cart and $\zeta(t)$ is the rotational angle of the string as seen in Fig. 2.4. The length of the string varies with time. Denoting by $h(t)$

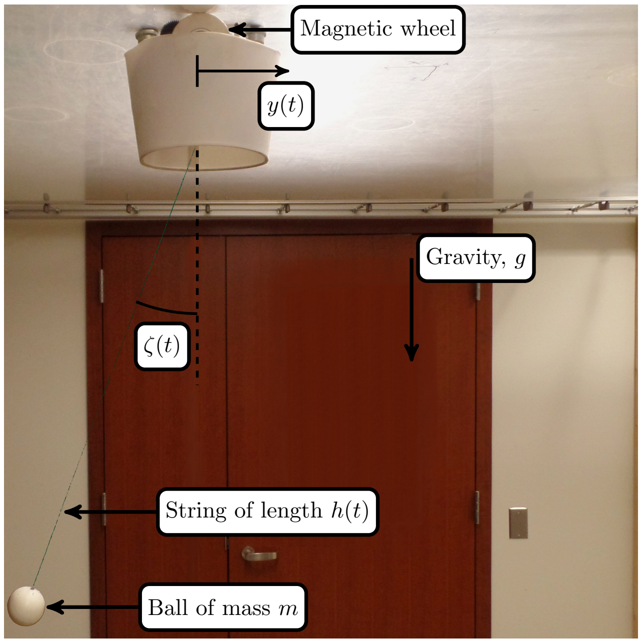


Figure 2.4. The experimental setup consists of an one-dimensional differential drive mobile robot with magnetic wheels (i.e. cart) and a ball suspended by a string. The string changes length by means of an actuated reeling system attached on the robot. The system configuration is measured by a Microsoft Kinect at $\approx 30\ Hz$. The full state is estimated using an Extended Kalman Filter. The Robot Operating System (ROS) is used for collecting sensed data and transmitting control signals (i.e. robot acceleration values). See more in [**1**, **2**].

the time-varying string length, $g$ the gravity acceleration and by $m$ and $c$ the mass and damping coefficient, the linearized system equations around the equilibrium $x = \mathbf{0}$, take the form in (2.2) with

$$A_i(t) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\alpha_i \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{g}{h(t)} & -\frac{c}{mh(t)^2} & -\frac{1}{h(t)}\alpha_i \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.39}$$

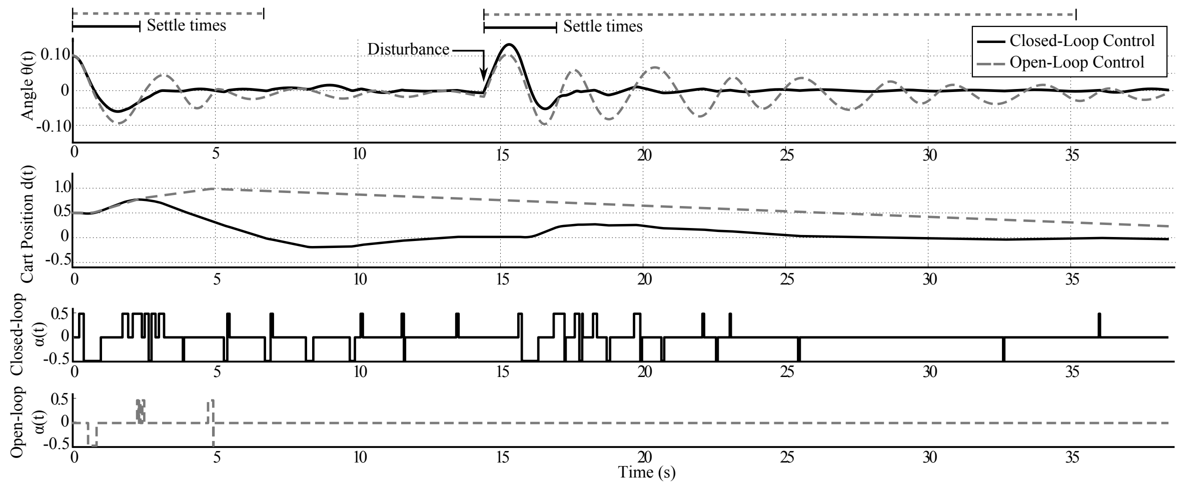


Figure 2.5. Open-loop SIOMS (Algorithm 1 with $T_M = 40\ s$) vs Closed-loop SIOMS (Algorithm 2 with $\delta = 0.5$ and $T = 3\ s$) in simulation.

$$h(t) = sin(t) + 2 \tag{2.40}$$

and $N = 3$ i.e. three possible modes. The cart's horizontal acceleration $\alpha$ is directly controlled and can switch between the values $\alpha_1 = 0$, $\alpha_2 = -0.5$ and $\alpha_3 = 0.5$. Notice we have augmented the state-space from $\mathbb{R}^4$ to $\mathbb{R}^5$ in order to transform the originally affine model to the linear form in (2.2). The augmented state vector is $x = [y, \dot{y}, \zeta, \dot{\zeta}, \tilde{u}]$ where $\tilde{u}$ is the auxiliary state variable. System parameters are defined as $m = 0.124\ kg$, $c = 0.05$ and $g = 9.8\ m/s^2$.

Our objective is to find the mode schedule that minimizes the angle oscillation while the cart remains in a neighborhood near the origin and is accordingly characterized by the quadratic cost functional (2.4) with $Q = diag[0, 0, 10, 1, 0]$ and $P_1 = diag[0.1, 0.01, 10, 1, 0]$. The system starts at an initial condition $x_0 = [0.5, 0, 0.1, 0, 1]^T$.

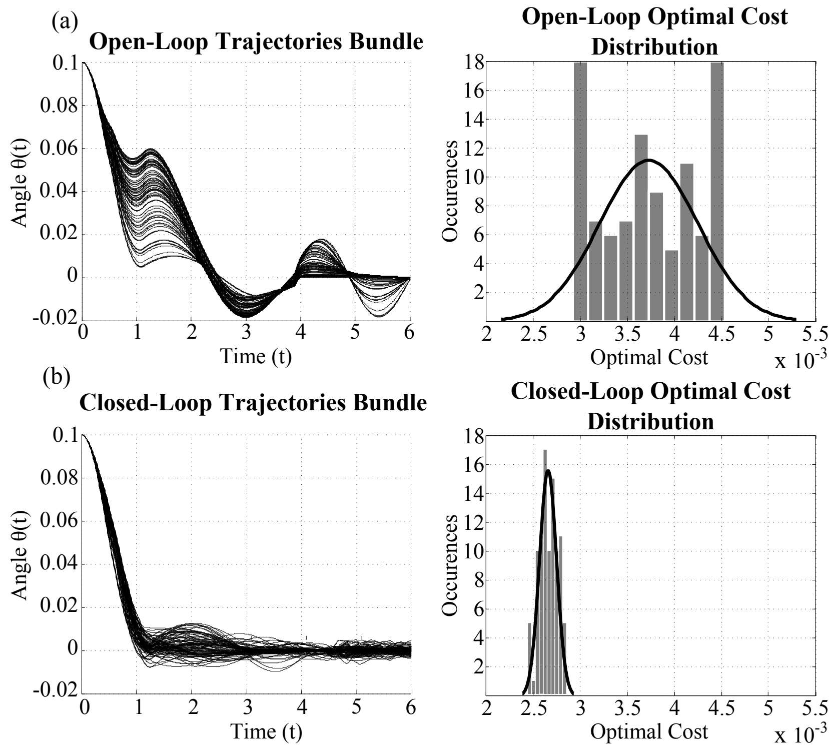


Figure 2.6. Robustness to uncertainty in the damping coefficient through Monte-Carlo analysis. Angle trajectories bundle and optimal cost distribution for (a) open-loop SIOMS (Algorithm 1 with $T_0 = 0$ and $T_M = 6$) and (b) closed-loop SIOMS (Algorithm 2 with $\delta = 0.2$ and $T = 3$).

### 2.5.1. Simulation Results

We apply Algorithm 2 to the optimal control problem stated previously and compare its performance with Algorithm 1 in terms of (1) disturbance rejection and (2) robustness to system parameter uncertainties. For real-time SIOMS execution, both algorithms were implemented in Python.

Disturbance rejection. We ran Algorithm 2 with parameters $\delta = 0.5$ and $T = 3\ s$ for a total of 40 seconds. A disturbance is applied at time $\approx 14\ s$. Each run of Algorithm 1 (i.e. 5 SIOMS iterations) lasted on average $0.04\ s$ of CPU time. Note that the algorithm was implemented in a real-time manner—starting the system simulation/integration from $t = 0\ s$, a new switching control is calculated and applied every $\delta = 0.5$ seconds using information about the current system state. For comparison, we additionally ran a one-time open-loop SIOMS (Algorithm 1) with $T_0 = 0\ s$ and $T_M = 40\ s$. The cost is reduced from $J_0 \approx 1.96$ to $J \approx 0.58$ after 15 iterations; the optimal switching control was pre-calculated and later applied to the system.

The results are illustrated in Fig. 2.5. Starting at an initial value of 0.1 *rad*, the angle has a settle time[6] of about 2.5 $s$ with closed-loop control compared to 5 $s$ when open-loop SIOMS is applied. As expected, the disturbance triggers a high angle oscillation with a settle time > 20 $s$, as the effect is not taken into account by the open-loop controller. The receding-horizon SIOMS, however, results in a much lower settle time of 2.5 $s$, providing an efficient real-time response to the random disturbance. The last 2 diagrams in Fig. 2.5 show the switched cart acceleration $\alpha$ with respect to time as calculated by each algorithm. In close-loop control where the most reliable performance is observed, a total of 65 switches occur with an average mode duration of $\approx 0.42\ s$ and a minimum mode duration (i.e. period during which the mode remain fixed) of $\approx 0.02\ s$.

Robustness to model uncertainties. In a subsequent comparison, we examine the robustness of Algorithm 2 to model uncertainties and compare its performance to Algorithm 1. In particular, we perform a Monte-Carlo analysis where both algorithms are run 100 times in the following scheme: the optimal switching control is calculated using the system model in (2.39) and is subsequently applied to an equivalent system with randomly added noise in the damping parameter, i.e. $c_{actual} = 0.05 + \omega$ where $\omega$ is a random real number in the range $[-0.05, 3.0]$ so that $c_{actual} \in [0, 3.05]$.

[6]Settle time is defined here as the time from the arrival of the disturbance until the angle reaches and stays within the settle boundary from −0.025 *rad* to 0.025 *rad* surrounding the origin.

We demonstrate the results in Fig. 2.6. The diagrams on the left show the resulting angle trajectories for $t \in [0, 6]$ of all algorithm runs. It can be observed that open-loop SIOMS is more sensitive to changes in the damping coefficient compared to closed-loop SIOMS that exhibits a more robust performance. The distribution of optimal costs across all runs is given in the remaining diagrams of Fig. 2.6. For both open-loop and closed-loop SIOMS, the optimal cost is calculated as in (2.4) over the resulting trajectories $x(t)$ for all $t \in [0, 6]$. The mean optimal cost in open-loop SIOMS is $\approx$ 0.0037 compared to $\approx$ 0.0027 in the closed-loop implementation. In addition, with receding-horizon SIOMS (Algorithm 2) the standard deviation is $0.08 \cdot 10^{-3}$ which is significantly lower than the standard deviation $0.51 \cdot 10^{-3}$ observed in open-loop SIOMS (Algorithm 1).

### 2.5.2. Experimental Results

In this section, the performance of the closed-loop hybrid controller (Algorithm 2) is evaluated experimentally on a real cart and suspended mass system (Fig. 2.4). More information about this experimental platform can be found in [**1**, **2**]. Due to geometric constraints and model discrepancies, a few changes in the parameters were made as follows: $h(t) = 0.4sin(t) + 1$ in (2.40), $c = 0.001$ in (2.39), $\delta = 0.4$ and $T = 5\ s$ in Algorithm 2. The same objective as in simulation was pursued i.e. real-time angle regulation with the robot position maintained close to the origin. The weight matrices in (2.4) were set as $Q = diag[0, 0, 1000, 0, 0]$ and $P_1 = diag[1, 0, 100, 0, 0]$. We ran 2 sets of experiments to illustrate the features of the hybrid controller based on Algorithm 2.

In Experiment 1, the SIOMS controller is initially inactive and we perturb the string angle by setting a predefined oscillatory trajectory to the cart/robot[7]. After approximately 6.6 seconds, the controller is activated to optimally drive the angle to zero using Algorithm 2. One example trial of Experiment 1 is illustrated in Fig. 8.6. During the perturbation, the angle exhibits an oscillatory response with peak amplitude at 0.25 *rad*. Once receding-horizon SIOMS is applied, the string angle starts approaching the origin with a settle time of 4.8 seconds and the robot moves slightly to the left before returning to the origin. For comparison purposes, Fig. 8.6 also

[7]A video of the experiment is available in *https://vimeo.com/nxrlab/sioms1*.

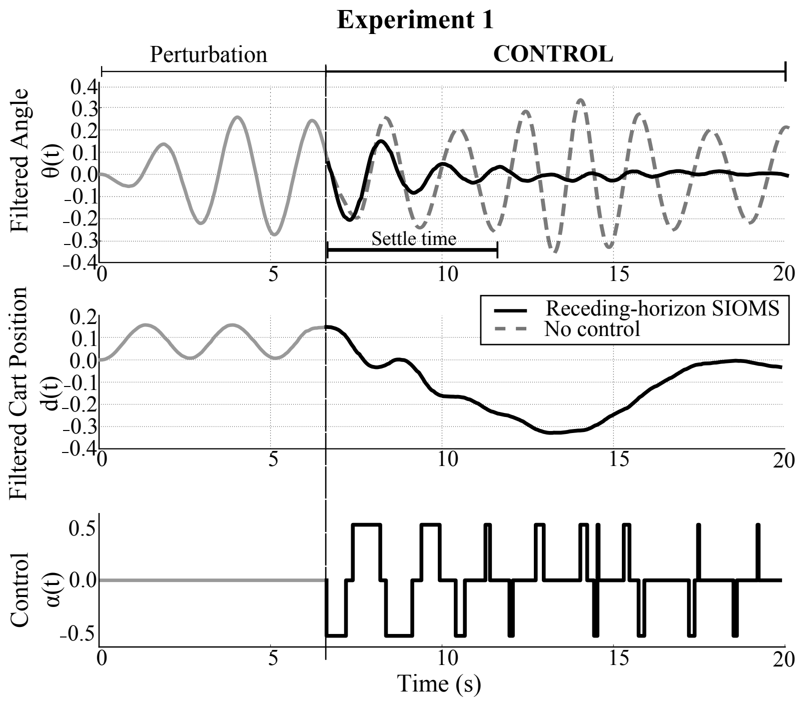


Figure 2.7. An example trial of Experiment 1. First, the robot follows a sinusoidal trajectory perturbing the string angle. Approximately 6.6 seconds later, receding-horizon SIOMS is applied in real time and drives the angle back to the origin in approximately 4.8 seconds. Without control, the angle exhibits high oscillations with minimal decay.

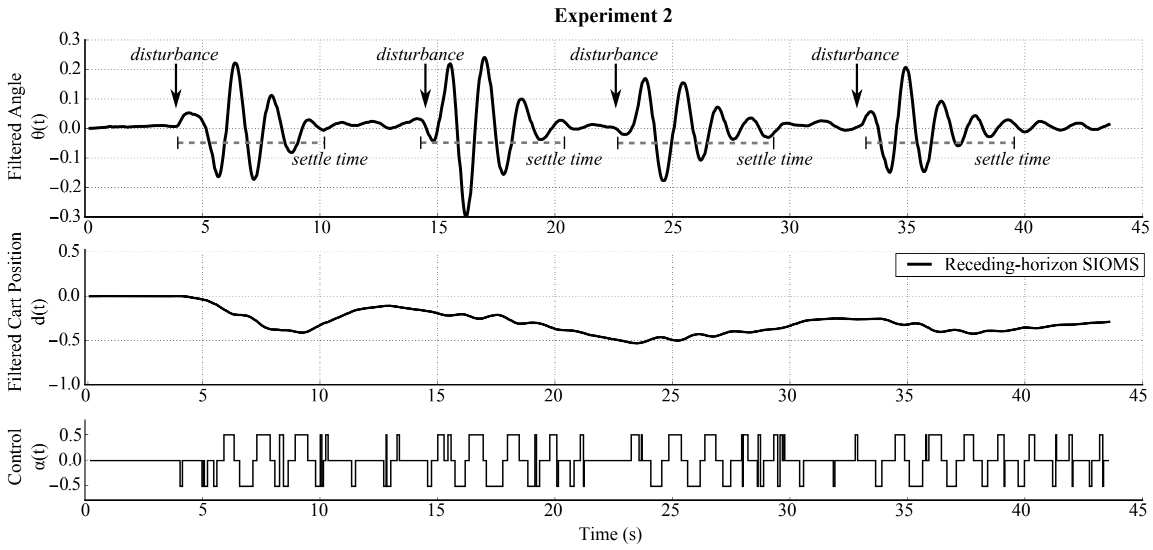


Figure 2.8. An example trial of Experiment 2. SIOMS controller is always active while a person applies random disturbances by pushing the suspended ball four times sequentially. The controller reacts in real time to regulate the angle. Approximate settle time is at 6 seconds.

Table 2.1. We ran 12 trials of Experiment 1 with 4 different perturbation levels.

| | **Perturbation 1** | | | **Perturbation 2** | | | **Perturbation 3** | | | **Perturbation 4** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | peak angle = $0.18rad$ | | | peak angle = $0.25rad$ | | | peak angle = $0.33rad$ | | | peak angle = $0.4rad$ | | |
| *Trial* | *1* | *2* | *3* | *1* | *2* | *3* | *1* | *2* | *3* | *1* | *2* | *3* |
| **switches / second** | 2.80 | 3.20 | 2.60 | 2.36 | 2.61 | 2.80 | 3.88 | 2.77 | 2.46 | 2.89 | 2.59 | 2.66 |
| **average mode duration (s)** | 0.34 | 0.27 | 0.32 | 0.31 | 0.35 | 0.32 | 0.38 | 0.33 | 0.35 | 0.31 | 0.32 | 0.34 |
| **settle time (s)** | 2.9 | 3.6 | 4.2 | 3.4 | 4.2 | 4.8 | 7.2 | 7.5 | 6.9 | 9.4 | 8.6 | 9.1 |

shows the angle trajectory for the case when no control was applied following the perturbation (i.e. $\alpha(t) = 0$). One may observe that the uncontrolled system is highly underdamped with no settle time achieved in a time horizon of $\approx$ 14 seconds. Note that the sinusoidal change in peak amplitude and frequency is a result of the time-varying string length.

We repeated Experiment 1 for four different perturbation levels, each characterized by the peak angle amplitude achieved. Three trials per perturbation were run i.e. twelve trials in total. As performance metrics, we used a) the number of switches per second, b) the average mode duration and c) the settle time for the string angle. Our goal was to verify the reliability and efficacy of the controller in noisy conditions induced by sensor and model deficiencies. The results are given in Table 2.1. Throughout the trials, the number of switches per second ranges from 2.3 to 3.8. The average mode duration also exhibits low variation among different trials with a range from 0.27 to 0.38 seconds. As expected, settle times increase with higher perturbation levels but remain fairly close among trials of the same perturbation.

In Experiment 2, we sought to evaluate the performance of the hybrid controller when random disturbances occur in real time. To achieve this, the experiment is initialized at $y = 0$, $\zeta = 0$, $\alpha = 0$ and zero velocities. With the receding-horizon SIOMS controller activated, a person pushes the suspended mass to create real-time disturbances. The controller responds to the disturbance to regulate the angle and drive it back to zero[8]. An example trial of Experiment 2 is presented in Fig. 8.7 where four consecutive disturbances of varied amplitudes are applied. One may observe that the controller regulates the angle with settle times of approximately 6 seconds in all four cases. Furthermore, as a result of the terminal cost applied at $y(t)$, the robot does not deviate significantly from the origin.

[8]A video of the experiment is available in *https://vimeo.com/nxrlab/sioms2*.

CHAPTER 3

# Control Alphabet Policies

This chapter presents a method for synthesis of control alphabet policies, given continuum descriptions of physical systems and tasks. First, we describe a model predictive control scheme, called switched sequential action control (sSAC), that generates global state-feedback control policies with low computational cost. During synthesis, sSAC alphabet policies are directly encoded into finite state machines using a cell subdivision approach. As opposed to existing automata synthesis methods, controller synthesis is based entirely on the original nonlinear system dynamics and thus does not rely on but rather results in a lower-complexity symbolic representation. The method is validated for the cart-pendulum inversion problem, the double-tank system and the SLIP model. The approach presents an opportunity for real-time task-oriented control of complex robotic platforms using exclusively sensor data with no online computation involved.

## 3.1. Introduction

This chapter constructs control alphabet policies (CAP) that achieve desired performance objectives, given a (nonlinear) system and finite set of constant control symbols. Symbolic control has been popular in the areas of robot control and motion planning [**77**] as a means to provide solutions for control on embedded systems with limited computational power [**4**].

Common symbolic control approaches include linear temporal logic (LTL) [**78**] and motion description languages (MDLs) [**79**]. Here, as in MDLs, we synthesize symbolic policies considering discretization at the controls level, i.e., symbols denote control modes forming a control alphabet. For example, in a helicopter-like vehicle, tasks like “land”, “ascend”, and “hover” might correspond to alphabet policies composed of constant control modes.

Our objective is to synthesize simple control alphabet policies that can be stored in finite state machines and realized with digital computer tools (e.g. [**80**]) inexpensively, i.e. without requiring online control calculation. The solution to the problem of CAP synthesis for a wide range of systems and control objectives will benefit automation systems in terms of computing power allocation and compactness by boosting their multi-tasking capacity (power allocation) and promoting miniaturization (compactness), in the fields of aviation [**17**], manufacturing [**18**], and robotic locomotion [**19**] among others.

The proposed control alphabet policies (abbreviated as CAP) are structured as finite state machines during the synthesis process. Figure 3.1 shows how our approach (in red) compares to common methodologies for controller automata sunthesis (in blue). The latter [**3–6**] generate lower-complexity symbolic representations of continuous systems (i.e. bisimulation [**6**] or symbolic model [**4**]) and compute the policy on the system abstraction. On the contrary, this work proposes that CAP synthesis (illustrated in blue) is entirely based on the original continuous nonlinear system dynamics instead of their lower-complexity abstractions. As a result, our objective is to provide solutions even for cases where a discrete system approximation is hard or impossible to obtain to aid in the controller synthesis. Subsequently, state-space abstractions are extracted based on the numerical global control policy. However, note that these state partitions do not aim to be a bisimulation of the actual system (as is formally defined in [**6**]) but a method for inexpensive representation of state-feedback controls and fast policy execution.

In order to achieve this alternative scheme, it is imperative that we formulate a method for state-feedback global control synthesis that is computationally inexpensive and can be encoded in finite states machines. Common numerical approaches in hybrid control [**16**,**24**,**25**,**32**] output open-loop time-dependent control trajectories that do not favor the synthesis of global state-feedback control alphabet policies[1] while they exhibit high execution times. To overcome these issues, we propose a numerical algorithm for global symbolic control which we call switched sequential action control (sSAC). The algorithm is based on sequential action control (SAC), a recent model-based optimal control scheme described in detail in [**81–84**] and relies on hybrid systems theory to select the next symbol that optimally *improves* the task objective instead of optimizing it.

The final step of the proposed synthesis process generates state-space partitions based on an sSAC control policy, using a cell subdivision approach, typically used for computation of invariant sets [**85**]. As opposed to [**4**], the approach employs a multi-resolution grid, so that the distribution of state space partitions is non-uniform. As a result, the number of discrete state partitions doesn't grow exponentially with respect to the dimension of the state space. Furthermore, since the non-uniform partitions are hyper-rectangles that intrinsically satisfy the control policy (see Definition 2 and 3), the guard equations of the finite state machines can be directly represented as nested state inequalities for fast controller execution.

[1]In other words, common mode scheduling algorithms do not naturally assign a symbolic action $u$ at any state $x$.

For method validation, we construct control alphabet policies for cart-pendulum inversion, double-tank fluid levels control and forward hopping using the spring-loaded inverted pendulum (SLIP) model. When used for cart-pendulum inversion, we show that this automated numerical process—with sSAC and two symbols—generates structurally identical results to the bang-bang analytical control law published in [**86**]. The last two examples illustrate CAP synthesis in systems with hybrid dynamics. CAP synthesis for the SLIP model indicates alternative uses of the CAP policies as embedded "background" controllers around which online controllers (e.g. sSAC or SAC) work to achieve high-level objectives. For example, a CAP finite state machine embedded in a robotic biped can be used to inexpensively coordinate walking, while high-level controllers complete more complex tasks (similar to the hypothesis that spinal cord circuitry coordinates locomotion in humans and other vertebrates [**87**]).

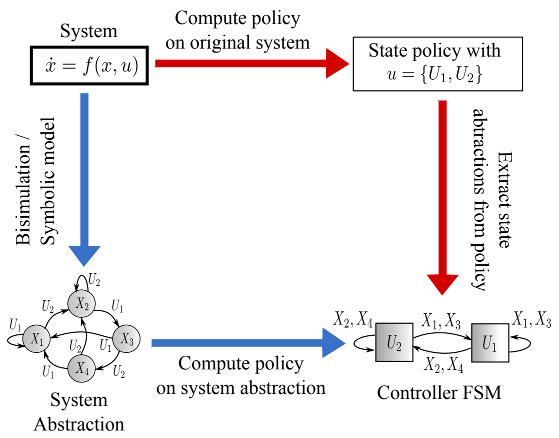

System
$\dot{x} = f(x,u)$
Compute policy on original system
State policy with $u = \{U_1, U_2\}$
Bisimulation / Symbolic model
Extract state abtractions from policy
$X_1$ $X_2$ $X_3$ $X_4$ $U_1$ $U_2$
System Abstraction
Compute policy on system abstraction
$X_2, X_4$
$X_1, X_3$
$U_2$ $U_1$
Controller FSM

Figure 3.1. A diagram showing how our approach (in red) compares to common automata synthesis methodologies [**3–6**] (in blue).

## 3.2. Symbolic Control Calculation

For the rest of this chapter, we consider continuous-time nonlinear systems with $n$ states $x : \mathbb{R} \to \mathcal{X} \subseteq \mathbb{R}^n$ and $m$ inputs $u : \mathbb{R} \to U \subset \mathbb{R}^m$ following equations of the general form

$$\dot{x} = f(x, u). \tag{3.1}$$

At any time $t$, input $u(t)$ can be one of the $N$ constant-value control vectors from the set $U = \{u_1, ..., u_N\} \subset \mathbb{R}^m$. The state is sometimes denoted as $t \mapsto x(t; t_0, x_0, u)$ when we want to make explicit the dependence on the initial time, initial state, and corresponding control signal.

**Assumption 1.** *The elements of dynamics vector* (3.1) *are real, bounded, continuously differentiable in x, and continuous in t and u.*

**Definition 1.** *A symbol is a constant-value control vector $u_i \in \mathbb{R}^m$ that belongs in the system's alphabet, i.e. control set $U$.*

To calculate state-dependent control symbols, we introduce switched sequential action control (sSAC), a variation of sequential action control (SAC) in [**81**], that performs global closed-loop symbolic control using the symbols in $U$. The algorithm follows a receding-horizon approach; controls are obtained by repeatedly solving online an open-loop symbolic control problem $\mathcal{P}$ every $t_s$ seconds (with sampling frequency $\frac{1}{t_s}$), every time using the current measure of the system state $x_{curr}$. However, it differs from common receding-horizon schemes in two major points: a) $\mathcal{P}$ does not search for a control trajectory over the full time horizon $T$ but rather selects a single symbolic control action $u_{k^*} \in U$ to be applied for a short amount of time $\lambda$, and b) open-loop solution optimally improves the performance objective (3.2) instead of optimizing it. $\mathcal{P}$ improves general tracking objectives of the form

$$J(x(\cdot)) = \int_{t_0}^{t_0+T} l(x(t))\, dt + \bar{m}(x(t_0 + T)) \;, \tag{3.2}$$

with incremental cost $l(x(t))$, terminal cost $\bar{m}(x(t_0 + T))$, initial time $t_0$ and time horizon $T$. The open-loop problem $\mathcal{P}$ is

$$\begin{aligned} &\mathcal{P}(t_0, x_0, T, J) : \\ &\quad \text{Find } k^* \in \{1, 2, ..., N\} \text{ and } \tau, \lambda \in \mathbb{R} \text{ such that} \\ &\qquad J(x(t; t_0, x_0, u_{sSAC})) < J(x(t; t_0, x_0, u_{default})) \\ &\qquad \text{with } u_{sSAC}(t) = \begin{cases} u_{k^*} & \tau \le t \le \tau + \lambda \\ u_{default} & \text{else} \end{cases} \\ &\qquad \text{subject to (3.1) with } t \in [t_0, t_0 + T] \text{ and } x(t_0) = x_0. \end{aligned} \tag{3.3}$$

The term $u_{default}$ refers to a default (nominal) control symbol. It is often $u_{default} = \mathbf{0}$ so that problem $\mathcal{P}$ outputs the optimal symbolic action relative to doing nothing (allowing the system to drift for a horizon into the future). Alternatively, $u_{default}$ may be an optimized feedforward controller providing a nominal trajectory around which sSAC would provide feedback.

The solution $u_{sSAC}(t)$ of problem $\mathcal{P}$ generates a switch of duration $\lambda$ in the dynamics (3.1) from $f(x, u_{default})$ to $f(x, u_{k^*})$. The triplet $(k^*, \tau, \lambda)$—i.e. a single symbol $u_{k^*}$, $k^* \in \{1, 2, ..., N\}$ along with its associated application time $\tau$ and duration $\lambda$—defines a symbolic sSAC control *action*. As the receding horizon strategy progresses, $\mathcal{P}(t_0, x_0, T, J)$ is solved for the current time $t_0$ using the measured state $x_0$, and the output control $u_{sSAC}(t)$ is applied for $t_s$ seconds with $0 < t_s \leq T$. The process is then repeated at the next sampling instance, i.e. $t_0 \leftarrow t_0 + t_s$. This closed-loop receding horizon strategy, results in a sequence of symbolic actions, forming a piecewise constant control signal $\overline{u}_{cl}(t)$ with state response $\overline{x}_{cl}(t)$. With regard to CAP synthesis, note that in each cycle iteration and with $u_{default}$ determined, sSAC only takes as input the current state $x_0$ and outputs a single control symbol $u_{k^*}$ or $u_{default}$ to be applied for a finite duration at $t_0$. We take advantage of this natural state-dependence of sSAC controls in order to achieve synthesis of control policies in Section 3.3.

### 3.2.1. Calculating a Control Actions Schedule

To make explicit the dependence on action duration $\lambda$, application time $\tau$ and symbol $k^*$, we write inputs $u : \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R} \times \{1, 2, ..., N\} \to U$ of the form of $u_{sSAC}(t)$ in (5.8) as

$$u(t; \lambda, \tau, k^*) = \begin{cases} u_{k^*} & \tau \leq t \leq \tau + \lambda \\ u_{default} & \text{else.} \end{cases}$$

When $\lambda = 0$, it is $u(t; 0, \cdot, \cdot) \equiv u_{default}$, i.e. no action is applied. Accordingly, we define $\bar{J}(\lambda, \tau, k^*) := J(x(t; t_0, x_0, u(t; \lambda, \tau, k^*)))$ so that the performance cost depends directly on the application parameters of a sSAC action. Using this notation, the open-loop problem $\mathcal{P}$ searches for the triplet $(k^*, \tau, \lambda)$ such that $\bar{J}(\lambda, \tau, k^*) < \bar{J}(0, \cdot, \cdot)$. There exists an open, non-zero neighborhood, $V = \mathcal{N}(\lambda \to 0)$, where the change in cost $\Delta J := \bar{J}(\lambda, \tau, k^*) - \bar{J}(0, \cdot, \cdot)$ is locally modeled by Taylor expansion as

$$\Delta J \approx \frac{d\bar{J}(\cdot, \tau, k^*)}{d\lambda^+}\lambda \tag{3.4}$$

for finite durations $\lambda \in V$. The quantity $\frac{d\bar{J}(\cdot,\tau,k^*)}{d\lambda^+}$—called mode insertion gradient and written $\frac{dJ}{d\lambda^+}\Big|_{\tau,k^*}$ for brevity—measures the first-order sensitivity of cost function (3.2) to application of symbol $u_{k^*}$ for infinitesimal duration $\lambda \to 0^+$ at time $\tau$. It is calculated as ( [**62**, **88**])

$$\frac{dJ}{d\lambda^+}\bigg|_{t,k} = \rho(t)^T(f(x(t),u_k) - f(x(t),u_{default})) \quad \forall t \in [t_0, t_0+T]. \tag{3.5}$$

The adjoint variable $\rho : \mathbb{R} \to \mathbb{R}^n$ provides the sensitivity of (3.2) to state variations along a predicted trajectory $x(t)$ $\forall$ $t \in [t_0, t_0+T]$. The adjoint satisfies[2]

$$\begin{aligned} \dot{\rho} &= -D_x l(x)^T - D_x f(x, u_{default})^T \rho \\ \text{subject to } \; \rho(t_0+T) &= D_x \bar{m}(x(t_0 \; + \; T))^T. \end{aligned} \tag{3.6}$$

Expression (3.4) indicates that the difference $\Delta J$ depends on the value of the mode insertion gradient $\frac{d\bar{J}(\cdot,\tau,k^*)}{d\lambda^+}$ in (3.5) and is parameterized by the application time $\tau$ and duration $\lambda$.

Using the above, we can compute a schedule, $u^* : \{t \,|\, t \in [t_0, t_0+T]\} \to U$, corresponding to the symbols that would optimally improve performance if applied for some duration at an arbitrary time $t \in [t_0, t_0+T]$. To achieve optimal cost improvement, i.e. $\Delta J < 0$ in (3.4) or equivalently $J(x(t;t_0,x_0,u_{sSAC})) < J(x(t;t_0,x_0,u_{default}))$ in $\mathcal{P}$, the schedule selects at each time $t$ the symbol number $k$ that drives (3.5) the closest to a specified negative value, $\alpha_d \in \mathbb{R}^-$. Therefore, based on the simulation of (3.1) and (3.6), the control schedule is calculated as

$$u^*(t) = \arg\min_{u_k, k=1,\dots,N} \left\{ \left[ \frac{dJ}{d\lambda^+}\bigg|_{t,k} - \alpha_d \right]^2 + \|u_k\|_R^2 \right\}, \quad t \in [t_0, t_0+T]. \tag{3.7}$$

[2]$D_x f(\cdot)$ denotes the partial derivative $\frac{\partial f(\cdot)}{\partial x}$.

The matrix $R > \mathbf{0} \in \mathbb{R}^{m \times m}$ provides an optional[3] metric on control effort. Parameter $\alpha_d$ determines how smooth or aggressive the sSAC response will be, which is particularly useful when we have a large number of symbols and the controller is used in human-in-the-loop applications (e.g. see [**82**]). In any other situation, $\alpha_d$ can be eliminated with no change on the controller's effect.

## 3.3. Control Alphabet Policies

The objective of this chapter is to generate control policies for inexpensive symbolic control. The resulting control laws are essentially hybrid automata, known to describe (discrete) switching conditions across a finite number of (continuous) dynamic modes [**89**]. Here, to stress the importance of the control alphabet policies (CAP) as standalone symbolic controllers relying on sensor data only, we use the following definition.

**Definition 2.** *A control alphabet policy (CAP) is a 3-tuple $\langle U, L, \mathcal{T} \rangle$ where*[4]*:*

*— U is the alphabet i.e a set of N symbols;*

*— L is a finite set of state space partitions that satisfy the sSAC control policy, i.e.* $L = \{L_i \subset \mathcal{X},\ i = \{1, 2, ...\} : u_{sSAC}(0)\big|_{\mathcal{P}(\cdot, x_m, \cdot, \cdot)} = u_{sSAC}(0)\big|_{\mathcal{P}(\cdot, x_n, \cdot, \cdot)}\ \ \forall x_m, x_n \in L_i\}$*;*

*— $\mathcal{T} : U \times L \to U$ is a (deterministic) transition function encoding a state-feedback control policy. This can be illustrated as a finite directed multigraph $(U, L)$, with elements in U being the vertices and elements in L the edges.*

[3]The weight on $u_i$ is optional because there is only a finite number of control symbols.

[4]From automata literature, control alphabet policies are similar in structure to labeled transition systems (LTS).

In this section, we describe a cell subdivision method that utilizes a non-uniform grid to extract state abstractions based on the symbolic control calculation methodology in Section 3.2[5].

**Algorithm 3** Compute Control Alphabet Policy

Initialize layer $k = 0$, desired maximum level $k_{max}$, time horizon $T$, number of test points $p \in \mathbb{N}$, set of state space partitions $L = \emptyset$, and set of compact sets to be divided $\Sigma^{(0)} = \{\Omega\}$ with $\Omega \subseteq \mathcal{X} \subseteq \mathbb{R}^n$ and $\Sigma^{(k)} = \emptyset\ \forall\ k > 0$.

```
(1) For all compact sets $\Sigma^{(k,j)} \subset \mathbb{R}^n$, $j = 1, ...$ in $\Sigma^{(k)}$:
    (a) Define a hyper-grid $\mathbf{C}^{(k,j)}$ on $\Sigma^{(k,j)}$ with $P^{(k,j)} \in \mathbb{N}$ cells and initialize labels $\ell(C_i^{(k,j)}) = Null$, $i = 1, ..., P^{(k,j)}$ (unlabeled grid cells).
    (b) For every grid cell $C_i^{(k,j)} \subset \Omega$, $i = 1, ..., P^{(k,j)}$:
        ■ Select $p$ test points $x_*$ from the cell interior or boundary.
        ■ For each point $x_*^{(s)} \in \mathbb{R}^n$, $s = 1, ..., p$, solve $\mathcal{P}(0, x_*^{(s)}, T, J)$ in (5.8). Then, $u^{(s)} = u_{sSAC}(0) \in U$.
        ■ If $u^{(s^1)} = u^{(s^2)}\ \forall\ s^1, s^2 \in \{1, 2, ..., p\}$ or $k = k_{max}$,
              label grid cell $C_i^{(k,j)}$ such that $\ell(C_i^{(k,j)}) = mode(\{u^{(s)} \in U : s = 1, ..., p\})$;
              add $C_i^{(k,j)}$ to set $L$;
          else
              add $C_i^{(k,j)}$ to set $\Sigma^{(k+1)}$.
(2) $k \leftarrow k + 1$
(3) Repeat from (1) until $\Sigma^{(k)} = \emptyset$ and $\bigcup_{L_i \in L} L_i = \Omega$.
```

### 3.3.1. State abstractions using cell subdivision

**Definition 3.** *Consider a compact set $\Omega \subset \mathbb{R}^n$ to a be a subset on the state space $\mathcal{X} \subseteq R^n$. A hyper-grid $\mathbf{C}$ on $\Omega$ divides $\Omega$ in a family of equally-sized n-orthotopes or hyper-rectangles*[6] *$C_i,\ i = 1, ..., P,\ P \in \mathbb{N}$ called cells, such that $\bigcup_{C_i \in \mathbf{C}} C_i = \Omega$ and $C_i \cap C_j = \emptyset\ \ \forall C_i, C_j \in \mathbf{C},\ \ i \neq j$. Each hyper-grid $\mathbf{C}$ is fully defined by n sets of values $\bar{x}_i = \{x_i^{min}, x_i^{min} + \delta_i, x_i^{min} + 2\delta_i, ..., x_i^{max}\}$ that in turn define the partitioning of each state with $\delta_i$ the size of the $i^{th}$ cell dimension, so that each*

[5]Note that the approach of this section can be applied without modification using any other global policy that outputs state-dependent controls based on a finite control alphabet.

[6]A *n*-orthotope or hyper-rectangle is the generalization of a rectangle for *n* dimensions.

*state $i$ has $q_i = \frac{x_i^{max} - x_i^{min}}{\delta_i}$ partitions. It is then $P = \{q_i\}^n$. To each cell $C_i$, associate a label $\ell(C_i) \in \mathbb{R}^m$ which can take values in $U$ or be Null (i.e. cell is unlabeled).*

Using the above definition, the complete process is given in Algorithm 3. The algorithm[7] takes as input a subset of the state space $\Omega \subseteq \mathcal{X}$ and outputs a partition $L$ of $\Omega$, i.e. $\bigcup_{L_i \in L} L_i = \Omega$ and $L_i \cap L_j = \emptyset \quad \forall L_i, L_j \in L, \quad i \neq j$. It is structured in layers starting from layer 0, with each new cell subdivision corresponding to a new layer. Initially, a standard—coarse—grid is applied on a state-space subset (layer 0). For each cell in the grid, if the labeling criterion is not satisfied, the cell is further subdivided to a finer grid of cells (e.g. layer 1) and the process repeats with finer layers until all cells are labeled whereafter partition $L$ has been fully specified. The $p$ test points in Step b can be the vertices, middle point and edge middle points of the cell as well as random interior points drawn from a distribution. A lower limit at the number of test points is $p_{min} = 2^n + 1$, i.e. the vertices and center point of the $n$-orthotope. Selection of number of test points $p$ is determined by the trade-off between desired accuracy level i.e. resolution ($p \uparrow$) and computational cost i.e. control algorithm runs ($p \downarrow$).

The algorithm terminates when a maximum layer $k = k_{max}$ is reached, wherein the set of compact sets to be divided (unlabeled cells) is empty i.e., $\Sigma^{(k)} = \emptyset$. The algorithm must be terminated when $k = k_{max} < \infty$, so that all cells lying on controller's switching manifolds are labeled and the boundaries of state partitions are resolved. The selected maximum level $k_{max}$ determines the lowest possible cell size of the resulting multi-resolution grid, which denotes the policy precision. The following proposition provides a method for selecting $k_{max}$ when the desired precision is known.

[7] In Algorithm 1, step b, $mode(A)$ denotes the value that appears most often in the set $A$.

**Proposition 4.** *In Algorithm 1, let the initial* 0*-level hyper-grid* $\mathbf{C}^{(0)}$ *comprise* $P^{(0)} = P_{init}$ *cells with* $q_i^{init} \geq 2$ *partitions for each state i. In addition, let* $\delta_i^{(k_{max})}$ *denote the desired* $i^{th}$ *cell dimension at maximum level* $k_{max}$ *for all* $i \in \{1, ..., n\}$*. Then, the maximum layer* $k_{max}$ *can be calculated as*

$$k_{max} = \max_{i \in \{1,...,n\}} \left\{ \min_j \left\{ k_j \in \mathbb{N} : k_j \geq \log_{q_i^{inter}} \frac{x_i^{max} - x_i^{min}}{q_i^{init} \delta_i^{(k_{max})}} \right\} \right\}. \tag{3.8}$$

**Proof.** At a specified level $k$, the size of the $i^{th}$ cell dimension is computed as $\delta_i = \frac{x_i^{max} - x_i^{min}}{q_i^{init} (q_i^{inter})^{k_{max}}}$ (this result follows directly from the practice of subdivision in Algorithm 1). Solving for $k$ gives us a lower limit for the level $k$ with respect to desired precision $\delta_i$ as in (3.8). Then for each state $i$, we select the minimum natural number $k_j$ that satisfies the lower limit. Finally, $k_{max}$ is the maximum $k_j$ across all states $i = 1, ..., n$.

□

The computational cost of the algorithm primarily depends on the number of required control algorithm runs, i.e. how many times we need to solve the open loop problem $\mathcal{P}(\cdot, \cdot, \cdot, \cdot)$ in Section 3.2. If a fine uniform grid was used for state space discretization, the number of algorithm runs would be equal to the total number of grid points. However, here we use a multi-resolution grid, that is constructed starting from a coarse uniform grid at level 0 with each cell further subdivided to smaller cells until labeled. This can significantly reduce the number of grid points that require symbolic control calculation. The more cells are labeled at level $k < k_{max}$, the less runs are executed. In this case, the exact number of algorithm runs is not known a priori and depends on how many cells will be labeled at each level (according to the labeling criterion). As a reference point, the following proposition provides an upper limit for the number of executed control algorithm runs that is only reached in the worst case scenario when all cells are labeled at the maximum level $k_{max}$.

**Proposition 5.** *In Algorithm 1, let the initial* 0*-level hyper-grid* $\mathbf{C}^{(0)}$ *comprise* $P^{(0)} = P_{init}$ *cells with* $q_i^{init} \geq 2$ *partitions for each state i, and each lower-level hyper-grid* $\mathbf{C}^{(k,j)}$ *comprise* $P^{(k,j)} = P_{inter}$ *cells with* $q_i^{inter} \geq 2$ *partitions for each state, for all* $j \in \mathbb{N}$ *and* $k > 0$*. Then, if* $p = 2^n + r$ *test points (i.e.* $2^n$ *n-orthotope vertices and r additional points from the cell interior) are selected for each cell* $C_i^{(k,j)}$ $\forall$ *k, j, i, then the maximum possible number of control algorithm runs in Section 3.2 (i.e. worst-case scenario if all cells are labeled at level* $k_{max}$*) is*

$$runs = \prod_{i=1}^{n} \left[ q_i^{init} (q_i^{inter})^{k_{max}} + 1 \right] + (1 + (P_{inter})^{k_{max}}) P_{init} r. \tag{3.9}$$

Proof. The first term computes the number of vertices of the hyper-grid $\mathbf{C}^{(k_{max})}$ with $q_i = q_i^{init}(q_i^{inter})^{k_{max}}$ partitions for each state $i$ at maximum level $k = k_{max}$. As $\mathbf{C}^{(k_{max})}$ consists of $P^{(k_{max})} = (1 + (P_{inter})^{k_{max}})P_{init}$ cells, the second term quantifies the additional number of runs due to interior test points $r$. □

So if the test points are $p = 2^n + 1$, (i.e. the vertices and center point of each $n$-orthotope cell), and the number of inner cells is $P_{inter} = 2^n$ with $q_i^{inter} = 2$ partitions for each state $i$ at each level $k$, the maximum number of algorithm runs is equal to $\prod_{i=1}^{n} \left[ q_i^{init} 2^{k_{max}} + 1 \right] + (1 + 2^{n k_{max}}) P_{init} r$. This number increases exponentially with both the number of levels $k_{max}$ and the number of dimensions $n$. However, since we are using a multiple-level non-uniform grid, the *actual number of control algorithm runs* is expected to be significantly lower as more cells are labeled at levels $k < k_{max}$. Whether the methodology is scalable to higher dimensions largely depends on

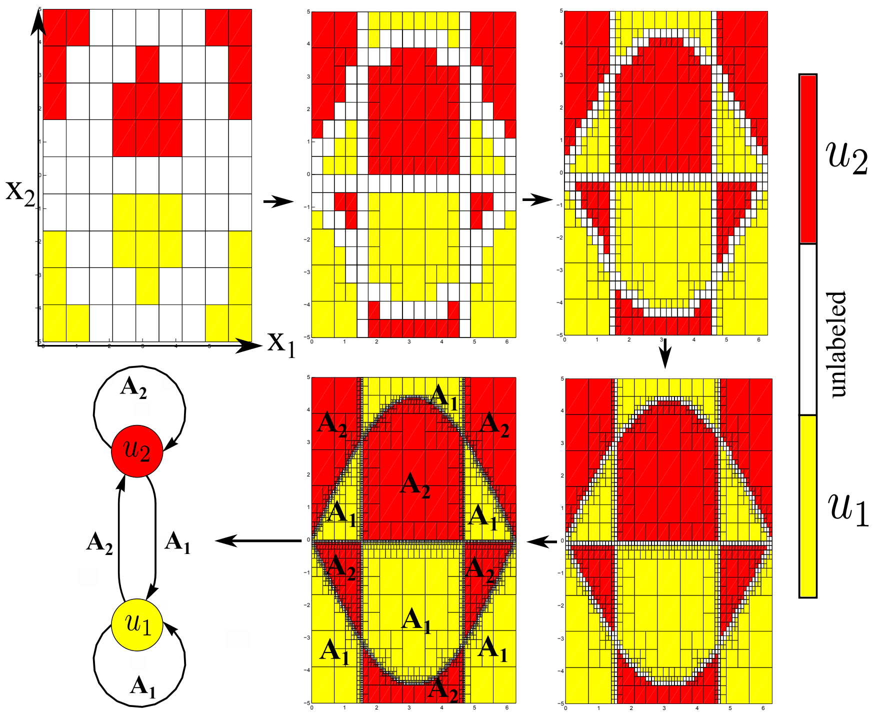


Figure 3.2. Control alphabet policies for cart-pendulum inversion. Phase plane plots showing consecutive layers of the cell subdivision strategy described in Algorithm 1 (layer 0 to layer 4) for synthesis of a two-symbol CAP with $U = \{u_1, u_2\} = \{5, -5\}$. The CAP finite state machine is also visualized. Vertices are the control symbols in $U$ and each edge label $A_i$ corresponds to the union of grid cells labeled with the symbol $u_i$.

the sparsity of controls over the system's state space and differs among choice of systems and symbols.

## 3.4. Example: Cart-Pendulum Inversion

### 3.4.1. Energy Tracking

This section demonstrates how to numerically synthesize control alphabet policies that globally lead to pendulum swing-up using full state feedback and a specified finite number of symbols. The reduced cart-pendulum system has $n = 2$ state variables, the angle between the vertical and

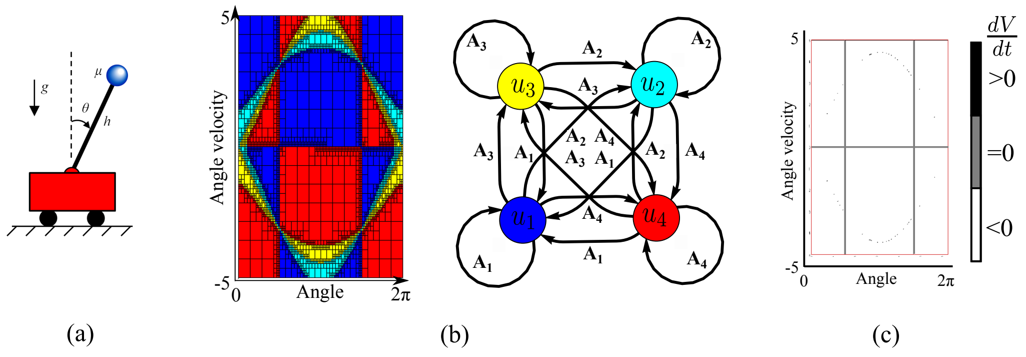


Figure 3.3. Control alphabet policies for cart-pendulum inversion using energy tracking cost. (a) Cart-pendulum system. (b) Phase plane plot and finite state machine showing the four-symbol CAP with $U = \{u_1, u_2, u_3, u_4\} = \{-5, -2, 2, 5\}$. (c) Numerical evaluation of the Lyapunov derivative on the control policies. Derivative is negative (stable) except for the gray lines $\dot{\theta} = 0$ and $cos\theta = 0$, where it is zero. These lines correspond to system singularities and are not concerning with regard to stability, as it happens that the vector field always drives the system out of these regions (non-invariant sets).

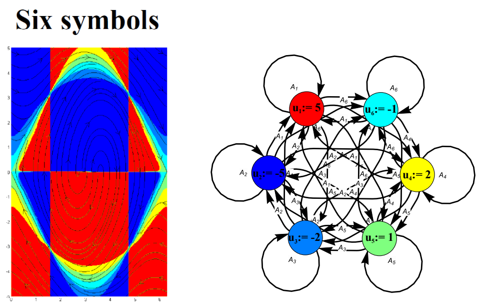


Figure 3.4. Control alphabet policies for cart-pendulum inversion using energy tracking cost: A six-symbol policy with $U = \{u_1, u_2, u_3, u_4, u_5, u_6\} = \{-5, -2, -1, 1, 2, 5\}$.

the pendulum $\theta$ and the rate of change of the angle $\dot{\theta}$. Denoting by $h$ the pendulum length, $g$ the gravity acceleration and $\mu$ the mass, the system equations take the form in (3.1) with

$$f(x,u) = \begin{pmatrix} \dot{\theta} \\ \frac{g}{h} sin\theta + \frac{u}{h} cos\theta \end{pmatrix}, \quad x = [\theta, \dot{\theta}] \tag{3.10}$$

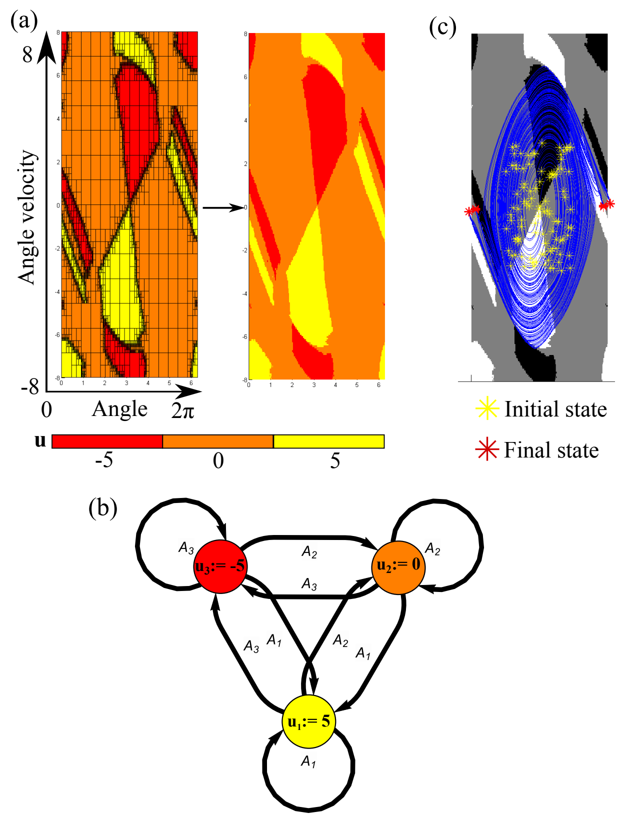


Figure 3.5. Control alphabet policy for cart-pendulum inversion using state tracking cost. (a) Phase plane plots showing the sSAC-generated 2-symbol control policy, along with the cell subdivision result from Algorithm 1. (b) Illustration of the control automaton. Vertices are control symbols and each edge label $A_i$ corresponds to the union of grid cells labeled with the symbol $u_i$ (see Algorithm 1). (c) Monte Carlo test: A bundle of 100 closed-loop trajectories (blue curves) with marked initial and final states.

where $u \in \mathbb{R}$ is the acceleration of the cart (i.e. $m = 1$). The pendulum is inverted when $(\theta, \dot{\theta}) = (0, 0)$. The system parameters take values $h = 2$, $\mu = 1$ and $g = 9.81$. Using this model, we synthesize control alphabet policies for symbolic cart-pendulum inversion by tracking the energy of the pendulum at the upright position. The energy of the uncontrolled pendulum $(u = 0)$ is $E(\theta, \dot{\theta}) = \frac{1}{2}\mu h^2 \dot{\theta}^2 + \mu g h(cos\theta - 1)$ so that $E_0 = 0$ at the upright unstable equilibrium.

For sSAC controls computation, we use the cost function (3.2) with $t_0 = 0$,

$$l(x(t)) = 0 \quad \text{and} \quad \bar{m}(x(t_f)) = \frac{1}{2}(E(\theta(t_f), \dot{\theta}(t_f) - E_0)^2. \tag{3.11}$$

In addition, for the open-loop problem $\mathcal{P}$, we used the parameters: $T = 1.2$, $u_{default} = 0$, $\alpha_d = -5J$, $R = 0.3$.

Figures 3.2, 3.3 and 3.4 illustrate the resulting sSAC state-dependent controls data for energy tracking using two symbols $U = \{-5, 5\}$, four symbols $U = \{-5, -2, 2, 5\}$ and six symbols $U = \{-5, -2, -1, 1, 2, 5\}$. It is noteworthy that the two-symbol control policy is structurally identical to the analytical bang-bang solution provided in[8] [**86**], albeit a result of a completely automated numerical procedure.

The computation was done on the compact set $\Omega = [0, 2\pi] \times [-5, 5]$ with a starting grid (layer 0) consisting of $P^{(0)} = 81$ grid cells and $q_i^{(0)} = 9$ partitions in each state $i$. In finer layers, cells were subdivided in grids of $P^{(k,j)} = 4 \ \forall \ k > 0, j \in \mathbb{N}$ cells. With $p = 5$ test points per cell and maximum level $k_{max} = 4$, a total of 3428 sSAC runs for the two-symbol policy and 8981 runs for the four-symbol policy were performed, compared to 41842 runs that would be required if all cells were labeled at maximum level 4 (computed using the expression in (3.9)). The resulting CAP are verified for Lyapunov stability next.

Lyapunov stability. For stability verification of the control laws, we use the Lyapunov function from [**86**], $V = |E - E_0|$ with $\frac{dV}{dt} = sign(E)\dot{E}$ and $\dot{E} = \mu \cdot h \cdot u \cdot cos\theta \cdot \dot{\theta}$. We numerically verified the value of the Lyapunov derivative for all states $x \in [0, 2\pi][-5, 5]$ (using a state-space discretization of grid size 0.01), and for both control policies $u$. The result (i.e. $sign(\frac{dV}{dt})\forall x$) is identical for both policies and is illustrated in Fig. 3.3c. The Lyapunov function decreases (i.e.

[8]The bang-bang control law in [**86**] is $u = |u_i| sign((E - E_0)\dot{\theta} cos\theta)$.

$\frac{dV}{dt} < 0$) as long as $\dot{\theta} \neq 0$ and $cos\theta \neq 0$ (gray lines). Implementation-wise, these lines (where $\frac{dV}{dt} = 0$) are not concerning with regard to stability, as it happens that the vector field always drives the system out of these regions (non-invariant sets). Scattered black points ($\frac{dV}{dt} > 0$) on the switching manifold are due to numerical noise generated by the grid discretization and numerical integration in sSAC.

### 3.4.2. State tracking

In this section, we generate automata for symbolic cart-pendulum inversion using a state tracking cost function. In particular, for sSAC controls computation, we use the cost function (3.2) with $t_0 = 0$,

$$l(x(t)) = \|x(t) - \bar{x}\|_Q^2 \quad \text{and} \quad \bar{m}(x(t_f)) = \|x(t_f) - \bar{x}\|_P^2 \tag{3.12}$$

and $\bar{x} = [0, 0]$ for the upright equilibrium. The weight matrices are $Q = Diag(\{1000, 10\})$ and $P = \mathbf{0}^{2\times 2}$.

Figure 3.5a, illustrates the resulting sSAC state-dependent controls data for state tracking using three symbols $U = \{-5, 0, 5\}$. The corresponding automaton is shown in Fig. 3.5b. Note that employing a larger number of symbols does not generate significantly different control policies in this state-tracking case[9]. This can be viewed as an indication that a maximum of three symbols are needed for cart-pendulum inversion. Since a Lyapunov function is not easy to analytically obtain for state tracking, we perform a Monte Carlo analysis to verify the automaton.

**3.4.2.1. Monte Carlo test.** To verify the state-tracking automaton in inverting the pendulum, we ran a Monte Carlo simulation with 1000 trials. For each trial, the system (3.10) starts from

[9]This is mainly because the symbols are restricted to be constant control vectors.

a random initial state $x_0$ and $t = 0$ and is integrated forward using Euler integration with fixed time step $\Delta t$. For every step, control $u$ is generated by the automaton using the current state $x_{curr}$ and the automaton transition function $\mathcal{T}$ (see Definition 1). The trial terminates when the system enters a region of attraction around the upright equilibrium[10] i.e. $|x_{curr} - \bar{x}| < \epsilon$, or when a time limit is exceeded i.e. $t > t_{max}$. A trial is marked as unsuccessful if the system does not enter the region of attraction before $t_{max}$. The rate of success was 100% in 1000 trials. Fig. 3.5c shows a bundle of 100 out of 1000 automaton-generated system trajectories (blue curves) with the initial and final states of each trial marked in yellow and red respectively. One can observe that although the initial states are scattered, all final states are close to the upright equilibrium, i.e. $x_{fin} = [0, 0]$ or $x_{fin} = [2\pi, 0]$.

## 3.5. Example: Two-Tank System

This section synthesizes control alphabet policies that track desired fluid levels at a double-tank system. Figure 3.6a shows the configuration of the system that consists of two tanks $T_1$ and $T_2$, with $T_1$ elevated at a height $h$ with respect to $T_2$. This tank configuration is a common laboratory setting and variants of it have been extensively used for evaluation of control methodologies [**24**, **56**]. The inflow and outflow rates to the tanks are controlled by the valves $V_1$, $V_2$, and $V_3$, that can only be open with flow rate $V_i = 1$ or closed so that $V_i = 0$. The states $x_1$ and $x_2$ are the fluid levels of tanks $T_1$ and $T_2$ respectively. According to Toricelli's law, a

[10]Note that once the region of attraction has been reached, the system can switch to a linearized controller (e.g. Linear Quadratic Regulator - LQR) for stabilization. This option will be addressed in the future, when more complex symbols are considered.

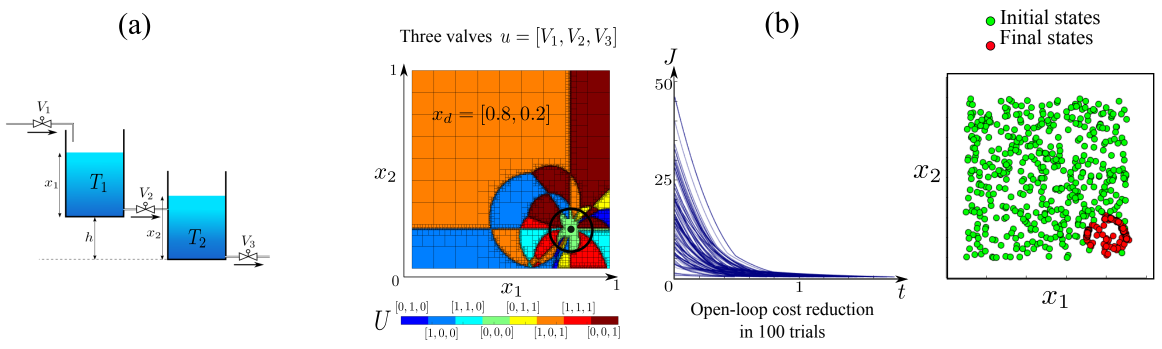


Figure 3.6. Control alphabet policies for tracking desired fluid levels in a double-tank system. (a) The two-tank configuration. (b) Case A: Phase plane plot showing the CAP policy with $u = [V_1, V_2, V_3]$ and $N = 8$ symbols. Desired state is $x_d = [0.8, 0.2]$. Figures on the right show Monte Carlo results with tolerance $\epsilon = 0.1$ (depicted as a circle). A 100% rate of success was achieved. Blue trajectories show reduction of open-loop cost $J$ in (3.2) over time for a sample of 100 trials. Note that $J$ was only calculated for verification purposes and was not part of the control calculation. Smaller figure shows a phase plane plot with the initial trial states in green and the final states in red for 500 trials.

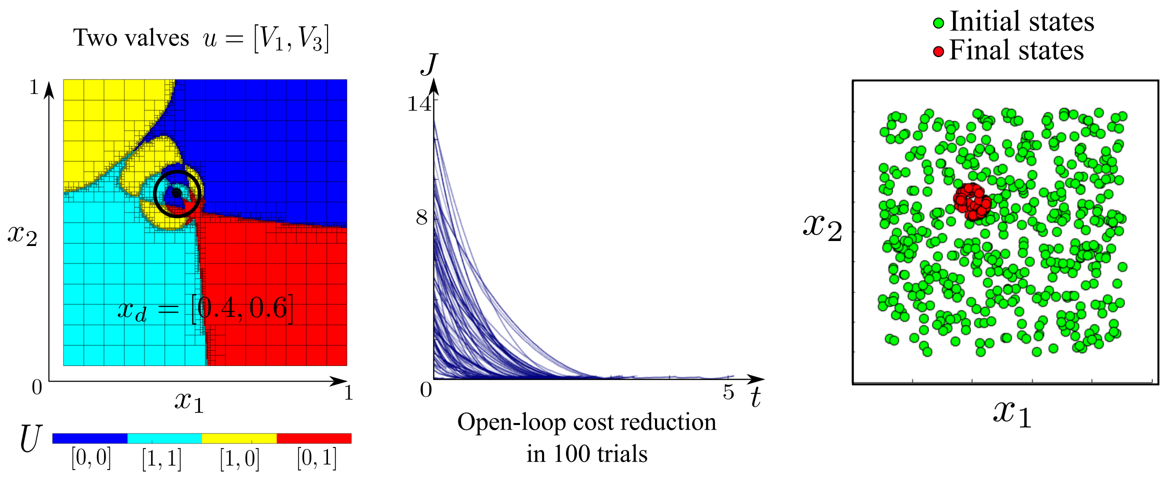


Figure 3.7. Control alphabet policies for tracking desired fluid levels in a double-tank system. Case B: Phase plane plot showing the CAP policy with $u = [V_1, V_3]$, $V_2 = 0.2$ and $N = 4$ symbols. Desired state is $x_d = [0.4, 0.6]$. Monte Carlo test was performed with tolerance $\epsilon = 0.05$. A 100% rate of success was achieved. In all trials, open-loop cost $J$ in (3.2) was decreased over time. Smaller figure shows a phase plane plot with the initial trial states in green and the final states in red for 500 trials.

simplified model of the system consists of the nonlinear vector field

$$f(x,u) = \begin{cases} \begin{bmatrix} V_1 - V_2\sqrt{x_1 - x_2 + h} \\ V_2\sqrt{x_1 - x_2 + h} - V_3\sqrt{x_2} \end{bmatrix}, & \text{if } x_2 > h \\ \begin{bmatrix} V_1 - V_2\sqrt{x_1} \\ V_2\sqrt{x_1} - V_3\sqrt{x_2} \end{bmatrix}, & \text{else.} \end{cases} \tag{3.13}$$

There is a (continuous) switch at the dynamics when $\Phi = x_2 - h$ crosses zero. The height takes value $h = 0.5$. For sSAC controls computation, we use the cost function (3.2) with $t_0 = 0$,

$$l(x(t)) = \|x(t) - x_d\|_Q^2 \quad \text{and} \quad \bar{m}(x(t_f)) = \|x(t_f) - x_d\|_P^2 \tag{3.14}$$

where $x_d$ is the desired state. The weight matrices are $Q = Diag(\{1,1\})$ and $P = Diag(\{100,100\})$. In addition, for the open-loop problem $\mathcal{P}$, we used the parameters: $T = 0.5$, $u_{default} = 0^{m\times 1}$, $\alpha_d = -5J$.

To explore the potential of CAP synthesis using sSAC, we synthesized policies for two different cases of control authority: A. with $u = [V_1, V_2, V_3]$, so that all valves are controlled and B. with $u = [V_1, V_3]$, so that only two valves are controlled and the middle valve has a constant flow rate $V_2 = 0.2$. The resulting policies and more details on each example case are given in Fig. 3.6 and Fig. 3.7.

For both control scenarios, we ran Monte Carlo tests with 1000 trials simulating system dynamics (3.13) from random initial states in the set $[0.1, 0.9] \times [0.1, 0.9]$. During simulation, control $u$ is generated by the CAP at 1000 Hz using the current state $x_{curr}$ and the CAP transition function $\mathcal{T}$ (see Definition 2). A trial terminates when the system is sufficiently close to the desired state as specified by tolerance $\epsilon$ i.e. $|x_{curr} - x_d| < \epsilon$ (successful trial), or when a time limit is exceeded i.e. $t > t_{max}$ (failed trial). Specifying a tolerance is consistent with previous results about quantized systems [**90**], stating that one can only implement feedback strategies that bring closed-loop trajectories arbitrarily close to the desired state. For both cases A and B, the rate of success was 100% in 1000 trials.

Computation of both policies (Algorithm 1) was done on the compact set $\Omega = [0, 1] \times [0, 1]$ with a starting grid (layer 0) consisting of $P^{(0)} = 81$ grid cells and $q_i^{(0)} = 8$ partitions in each state $i$. In finer layers, cells were subdivided in grids of $P^{(k,j)} = 4 \ \forall \ k > 0, \ j \in \mathbb{N}$ cells. With $p = 5$ test points per cell and maximum level $k_{max} = 4$, a total of $7,621$ sSAC runs for case A and $3,965$ runs for case B were performed, compared to $41,842$ runs that would be required if all cells were labeled at maximum level 4 (computed using the expression (3.9)). Interestingly, we repeated the calculation for $k_{max} = 5$, wherein a total of $17,125$ sSAC runs for case A and $9,474$ runs for case B were performed, compared to $166,546$ runs that would be required if all cells were labeled at maximum level 5. Therefore, although the worst-case scenario sSAC calculations increased exponentially, the actual number of calculations only almost doubled compared to the case with $k_{max} = 4$.

## 3.6. Example: Planar SLIP Hopper

This final section synthesizes control alphabet policies for hopping on vertical ground using a 2-D spring-loaded inverted pendulum (SLIP) model (fig. 3.8); a model that is common in analysis and control synthesis of dynamic hopping and running [**84**, **91**].

The state, $x = [\dot{x}_m, z_m, \dot{z}_m, y]$, consists of the horizontal velocity, the vertical position and vertical velocity of the center of mass followed by the horizontal displacement of "toe" (i.e. spring

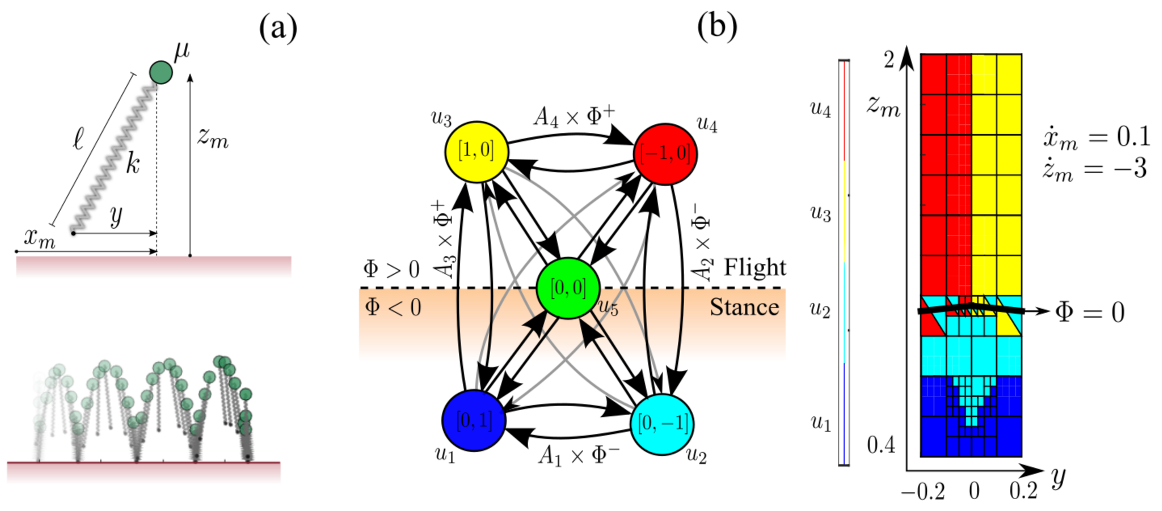


Figure 3.8. Control alphabet policy for hopping forward using the SLIP model. (a) The SLIP configuration and an illustration of SLIP hopping during a successful Monte Carlo trial. (b) On the left is the finite state machine with $N = 5$ symbols. The indicator function $\Phi$ is included at the finite state machine edges. On the right is a phase plane plot of $z_m$ with respect to $y$, showing the policy for $\dot{x}_m = 0.1$ and $\dot{z}_m = -3$. The switching manifold $\Phi = 0$ only depends on $z_m$ and $y$ and is plotted too. Note the cells on the manifold are assigned two labels (i.e colors) one for $\Phi > 0$ and one for $\Phi < 0$.

endpoint) with respect to the mass position. Horizontal position $x_m$ is not included because we assume a uniform vertical ground.

The dynamics of the SLIP are hybrid and include two phases: 1) a flight phase where the toe endpoint is in the air and 2) a stance phase where the toe is in rolling contact with the ground.

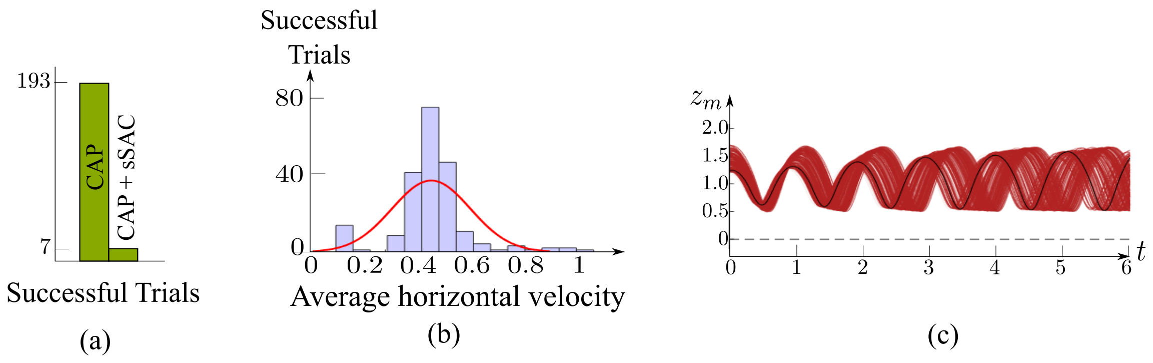


Figure 3.9. Control alphabet policy for hopping forward using the SLIP model. (a) We ran a Monte Carlo test with 200 trials, 193 of which were successful by only using the CAP for control. The other 7 trials relied on sSAC to account for states out of CAP range, i.e. for $x \notin \Omega$. (b) Histogram of the average hopper velocities across trials. The mean of the Gaussian fit is 0.39 with 0.16 standard deviation. (c) The bundle of $z_m$ trajectories for all trials.

The system's vector field is:

$$f(x,u) = \begin{cases} \begin{bmatrix} 0 \\ \dot{z}_m \\ -g \\ -u_f \end{bmatrix}, & \Phi(x) > 0 \\ \begin{pmatrix} (k(\ell_0 - \ell) + u_s)y\frac{1}{\mu\ell} \\ \dot{z}_m \\ (k(\ell_0 - \ell) + u_s)z_m\frac{1}{\mu\ell} - g \\ \dot{x}_m \end{pmatrix}, & \Phi(x) \leq 0 \end{cases} \tag{3.15}$$

where $g = 9.81$ is the acceleration due to gravity, $\mu = 1$ is the mass of the hopper, and $k = 90$ is the spring constant. The control vector is $u = [u_f, u_s]$ where $u_f$ controls the relative velocity of the toe $\dot{y}$ along the $x$ axis in flight mode and $u_s$ applies force along the spring axis in stance mode. Both inputs can take the values $\{-1, 0, 1\}$ so that there are $N = 5$ symbols in total (see Fig. 3.8b). The length, $\ell$, of the SLIP model matches the resting length of the spring, $\ell = \ell_0 = 1$, in flight and $\ell = \sqrt{y^2 + z_m^2}$ in the stance phase. The transition between flight and stance is state-based and determined by zero crossings of an indicator function, $\Phi(x) = z_m - \frac{\ell_0 z_m}{\ell}$. To avoid approximating $\Phi$ during CAP synthesis with multi-resolution $n$-orthotopes (which would be computationally expensive), we modify the CAP structure so that the transition function $\mathcal{T}$ in Definition 2 includes the indicator function as $\mathcal{T} : U \times L \times \Phi \rightarrow U$.

For sSAC controls calculation, we employ the cost function in (3.2) and (3.14), with weight matrices $Q = \mathbf{0}^{4\times4}$ and $P = Diag(\{120, 150, 0, 0\})$. Our objective is to synthesize a CAP that controls the SLIP model to hop forward with low velocity without falling. Therefore, we set $x_d = [0.5, 1.6, 0, 0]$. In addition, for the open-loop problem $\mathcal{P}$, we used the parameters: $T = 0.6$, $u_{default} = 0^{2\times1}$, $\alpha_d = -20000$. Because it is difficult to encode the SLIP behavior in a finite state machine for a wide range of states, we explore the efficacy of the CAP both as a standalone controller and as an inexpensive finite state machine that works in conjunction with sSAC to achieve a desired performance. We will see that even in the second case, computational cost of control is significantly reduced using the CAP, as sSAC is only activated for a small percentage of time. This example brings forth alternative applications of control alphabet policies where the latter are used as embedded "background" controllers around which online controllers (e.g. sSAC or SAC) work to achieve high-level objectives. For example, a CAP embedded in a biped can be used to inexpensively maintain an upright position, while high-level controllers complete more complex tasks.

The resulting finite state machine and a 2-dimensional section of the state partitions are shown in Fig. 3.8b. For CAP verification, we ran a Monte Carlo tests with 200 trials simulating system dynamics (3.15) from random initial states in the set $[0.01, 0.1]\times[1.2, 0.7]\times[-0.4, 0.4]\times[-0.1, 0.1]$ from time $t_0 = 0$ to $t_f = 6$. During simulation, control $u$ is generated by the CAP at 1000 Hz using the current state $x_{curr}$ and the CAP transition function $\mathcal{T} : U \times L \times \Phi \rightarrow U$ that takes into account the current value of the indicator function. A trial is marked as successful if the hopper does not fall (i.e. $z_m > 0$) with positive average velocity ($\dot{x}_m > 0$). 93.5% of the trials were successful using only the CAP for control calculation. For the rest 6.5%, sSAC was activated to compute controls for states out of the CAP range, i.e. for $x \notin \Omega$. The mean percentage of time during which sSAC was active per trial was 14.3%, i.e. 0.85 out of 6 time units on average, indicating that online sSAC runs did not significantly increase the computational cost.

CAP computation (Algorithm 1) was done on the compact set $\Omega = [-0.2, 0.8] \times [0.4, 2] \times [-4, 4]\times[-0.2, 0.2]$ with a starting grid (layer 0) consisting of $P^{(0)} = 800$ grid cells and $q^{(0)}_{1,3} = 5$, $q^{(0)}_2 = 16$, $q^{(0)}_4 = 2$ partitions in each state. In finer layers, cells were subdivided in grids of $P^{(k,j)} = 16 \; \forall \; k > 0, \;\; j \in \mathbb{N}$ cells. With $p = 17$ test points per cell and maximum level $k_{max} = 3$, a total of $3,100,288$ sSAC runs were performed, compared to $6,964,033$ runs that would be required if all cells were labeled at maximum level 3 (computed using the expression (3.9)).

# Part 2

# Real-Time Information-Driven Exploration for Symbols

CHAPTER 4

# Introduction

In this chapter, we introduce the idea of information-driven exploration for automated systems. We start by motivating the need for exploration based on symbolic automation. Furthermore, we list the requirements that an exploration strategy must meet so that it suitable for tracking and learning symbols in real time. Finally, a thorough investigation of existing exploration strategies is provided, along with current ergodic control approaches.

## 4.1. Why Exploration?

The reader might notice that symbols are used with varied meanings throughout this thesis (from constant control actions, to moving targets and shapes). However and according to the symbol definition provided in Section 1.2, it is always true that a quantity can be identified as a symbol as long as it can be associated with unique information. In accordance with this, it is imperative that we have a way of learning this information (when a symbol alphabet is unavailable) and tracking this information (once a symbol alphabet has been built). Both tasks can be achieved through *information-driven exploration*. To elaborate, consider the example case where an agent is searching for a circular-shaped object on a two-dimensional field. If the symbol information signature is simply the object shape, an agent must perform exploration proportionally to this spatial information density i.e. explore more around the areas where the object is expected to be based on its unknown shape.

In Part 1, we focused on acting with symbols, where symbols are defined as discrete control actions. But when building control policies for cart-pendulum inversion and SLIP walking (see Sections 3.4 to 3.6), we saw that symbols are in fact more than numbers; they are abstract shapes on the *n*-dimensional state space encompassing information about a state-dependent control policy. We used a multi-resolution grid to extract these symbols but consider this: what would we intuitively do if we were asked to sense bumps, engraved on a surface, using only the sense of touch on our single finger? If we followed the multi-resolution grid idea, we would point, briefly touch and then immediately remove our finger from pre-specified grid points on the surface. But this is far from what we would do in practice: we would probably start sweeping our finger along the surface focusing on areas where a bump is sensed. In other words, we would *explore* the surface driven by a dynamically-varying expected information density. The state space is not much different from the engraved surface and the bumps from discrete control symbol areas. Getting inspiration from the real world, we can abstractly explore the state space using information about potential boundary locations. An example of this process is described in Chapter 7.

To sum up, the ability to perform information-driven exploration is necessary for achieving symbol extraction and detection regardless of the nature of symbols.

## 4.2. Exploration Challenges and Contribution

This part of the thesis considers the problem of real-time motion planning for area search, coverage and target localization. Although the above operations are often considered separately, they essentially all share a common objective: tracking a specified distribution of information across the terrain. Our approach deviates from common solutions of space grid decomposition in area coverage [**92–96**] and/or information maximization in target localization [**8–11**,**97–100**] by employing the metric of *ergodicity* to plan trajectories with spatial statistics that match the terrain spatial distribution in a continuous manner. By following this approach, we can establish a unified framework that achieves simultaneous search and localization of multiple targets (e.g. localizing detected targets while searching for new targets when the number of total targets is unknown) without added complexity. Previous work [**101–103**] has suggested using ergodic control for the purpose of motion planning for search and localization (albeit separately). However, due to its roots to optimal control theory, ergodic control has been associated with high computational cost that makes it impractical for real-time operations with varying information distribution. The contribution of this work is a model predictive control (MPC) algorithm based on hybrid systems theory that exhibits low execution times even for high-dimensional systems with complex nonlinear dynamics while providing stability guarantees over both dynamic states and information evolution. To the best of our knowledge, this work includes the first application of an online ergodic control approach in real-time experimentation—here, using the `sphero` robot [**104**].

Ergodic theory relates the time-averaged behavior of a system to the set of all possible states of the system and is primarily used in the study of fluid mixing and communication. We use ergodicity to compare the statistics of a search trajectory to a terrain spatial distribution—the distribution may represent probability of detection for area search, regions of interest for area coverage and/or expected information density for target localization. The idea is that in an efficient exploration strategy, the trajectory followed by a robot should spend more time exploring regions of space with higher information, as encoded in the terrain distribution.

To formulate the ergodic control algorithm, we employ hybrid systems theory to analytically compute the next control action that optimally improves ergodicity, in a receding-horizon manner. The algorithm—successfully applied to autonomous visual rendering in [**23**]—is shown to be particularly efficient in terms of execution time and capable of handling high-dimensional systems. The overall contribution of this work combines the following features in a coherent manner.

*Real-time execution*: When planning motions for robotic agents performing search and localization, real-time optimal control in a receding-horizon format, allows for closed-loop model-based motion planning, while circumventing uncertainty on the vehicle dynamics [**105**]. In Chapter 6, we demonstrate in simulation real-time reactive control for a quadrotor model in SE(3) as well as a robotic fish-like vehicle. In Section 8.2.2, we show in experimentation how an ergodically controlled `sphero` robot can localize projected targets in real time using bearing-only measurements.

*Adaptive performance*: In real-world applications such as automated surveillance, search-and-rescue and target tracking, terrain spatial distributions may change dynamically to incorporate new information. For example, when tracking solar radiation [**106**] the distribution varies to reflect changing weather conditions; when tracking a target, it changes dynamically every time the belief on target's state is updated. The proposed algorithm reactively adapts to changing distribution of information across the workspace.

*Nonlinear agent dynamics*: Planning trajectories that satisfy the vehicle's dynamics ensures that the vehicle will not have to accomplish hard or infeasible maneuvers that will slow down the exploration procedure. In addition, it allows us to take advantage of the agent's dynamics in order to cover regions of interest more efficiently—for example, plan paths that use the natural surging and swaying motions of a robotic fish instead of canceling them. As opposed to geometric point-to-point approaches [**95**, **107**, **108**], the proposed algorithm controls agents with complex nonlinear dynamics, such as robotic fish [**109**], unmanned aerial vehicles (UAVs) [**110**, **111**], etc., taking advantage of their dynamical features to achieve efficient area exploration.

*Stability of information states*: We establish requirements for ergodic stability of the closed-loop system resulting from the receding-horizon strategy in Section 5.3.

*Multi-objective control capacity*: In real-world complex operations, area exploration might not be the only control objective that the robotic agent has to accomplish. For example, a UAV can localize a target while executing a perching motion [**112**, **113**], or a robot cleaner can cover a given area while identifying and avoiding obstacles [**114**]. To facilitate this, the proposed algorithm can work in a shared control scenario by wrapping around controllers that implement other objectives. This dual control problem solution is achieved by encoding the non-information related (secondary mission) control signal as the nominal control input $u_i^{nom}$ in the algorithm (see Algorithm 4). In the simulation examples, we show how this works by wrapping ergodic iSAC around a PD controller for height regulation in order to control a quadrotor to explore a terrain.

*Multi-agent distributability*: Formulating the multi-robot exploration task as an ergodic process is a promising approach for resolving the issue of task allocation across multiple controllers (e.g. swarm robots) [**115**]. Here, we show how ergodic control is distributable to $N > 1$ agents with no added computational complexity. Each agent computes control actions *locally* using their own processing unit but still shares information *globally* with the other agents after each algorithm cycle.

*Generalizability and robustness in multiple-targets localization*: It is common for the complexity of sensor motion planning strategies to scale with respect to the total number of targets, as a separate motion needs to be planned for each target [**12**, **116**]. As opposed to this, the proposed ergodic control approach controls the robotic agents to track a universal non-parameterized information distribution across the terrain instead of individual targets independently, thus being completely decoupled from the estimation process and independent of the number of targets. Through a series of simulation examples and Monte Carlo experimental trials, we show that ergodically controlled agents with limited sensor ranges can reliably detect and localize static and moving targets in challenging situations where a) the total number of targets is unknown; b) a model of prior target behavior is not available; c) agents acquire bearing-only measurements; d) a standard Extended Kalman Filter (EKF) is used for bearing-only estimation.

*Joint area coverage and target localization*: Planning routes that simultaneously optimize the conflicting objectives of search and tracking is particularly challenging. A common approach is to optimize an objective function that includes a weighted combination of the individual objectives [**117**, **118**]. However, performance is highly dependent on the choice of weights and in addition the optimization of such cost functions is generally time-consuming. To address this issue, we propose an ergodic control approach where the dual objective is encoded in the spatial information distribution that the statistics of the robotic agents trajectories must match.

## 4.3. Review of Exploration Strategies

### 4.3.1. Area Search and Coverage

An area search function is required by many operations, including search-and-rescue [**119**,**120**], hazard detection [**108**], agricultural spraying [**121**], solar radiation tracking [**106**] and monitoring of environmental phenomena, such as water quality in lakes [**122**]. In addition, complete area coverage navigation that requires the agent to pass through every region of the workspace [**123**] is an essential issue for cleaning robots [**114**, **124**], autonomous underwater covering vehicles [**125**,**126**], demining robots [**127**], automated harvesters [**128**], etc. Although slightly different in concept, both applications—search and coverage—involve motion planning for tracking a distribution of information across the terrain. For purposes of area search, this terrain spatial distribution indicates probability of detection and usually exhibits high variability in both space and time as it dynamically absorbs new information about the target's whereabouts. In area coverage applications, on the other hand, the terrain distribution shows regions of interest and is normally near-uniform with possible occlusions.

A number of contributions in the area of robotic search and coverage decompose the exploration space to reduce problem complexity. Grid division methods of various geometries, such as Voronoi divisions [**92–96**], are commonly employed to accomplish this. While these methods work well, their scalability to more complex and larger terrains where the number of discrete divisions increases, is a concern. In addition, existing methods plan paths that do not satisfy the robotic agents' dynamics and thus are not feasible system trajectories. This raises the need for an additional step where the path is treated as a series of waypoints and the agent is separately controlled to visit them all [**95**, **107**, **108**]. This double-step process—first, path planning and then, robot control— might result in hard-to-accomplish maneuvers for the robotic system, a setback that inhibits and slows down the operation. Finally, decomposition methods often do not respond well to dynamically changing environments—for example when probability of detection across the workspace varies according to incoming data—because grid updates can be computationally intensive. Therefore, most existing solutions only perform non-adaptive path planning for search and coverage offline i.e. when the distribution of information is known and constant. To overcome this issue when monitoring environments with changing distributions, an alternative solution is to control only the speed of the robotic agents over a predefined path [**129**].

The algorithm described in Chapter 5 is distinguished from the aforementioned methods as it provides a solution to the problem of area search and coverage without requiring any decomposition of the workspace, by representing probability of detection as a continuous function over the terrain. Furthermore, it performs online motion planning by reactively responding to changes in the terrain spatial distribution in real time, while taking into account the agent's dynamics.

*Multi-agent Coordinated Coverage:* The objective of multi-agent coverage control is to control the agents motion in order to collectively track a spatial probability-of-detection density function [**130**] across the terrain. Shared information is a necessary condition for coordination [**131**]. Several promising coverage control algorithms for mobile sensor networks have been proposed. In most cases, the objective is to control the agents to move to a static position that optimizes detection probability $\Phi$, i.e. to compute and track the final states $x_i(t_f)$ that maximize the sum of $\int \mathcal{F}(\|x - x_i(t_f)\|)\Phi(x)dx$ over all agents $i$ where $\mathcal{F}$ indicates sensor performance. Voronoi-based gradient descent [**132, 133**] is a popular approach but it can converge to local maxima. Other approaches employ cellular decomposition [**134**], simulated annealing [**135**] or game theory [**136**, **137**] to achieve distributed coverage. The main drawback is that existing algorithms do not consider time-dependent density functions $\Phi$, so they are not suitable for realistic applications where probability of detection varies.

Importantly, ergodic multi-agent coverage differs from the above coverage solutions in that it aims to control the agents so that the spatial statistics of their full trajectories—instead of solely their final states—optimize detection probability, i.e. the time-averaged integral $C(x) = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \delta[x - x_i(t)]$, where $\delta$ is the Dirac delta, matches the spatial distribution $\Phi(x)$ as $t_f \to \infty$. This means that if we capture a single snapshot of the agents ergodic motion, there is no guarantee that their current configuration will be maximizing the probability density. However, as time progresses the network of agents is bound to explore the terrain as extensively as possible. An advantage of the ergodic coverage control algorithm of this thesis is that it can be performed online in order to cover terrains with time-dependent (or sensed in real time) density functions. We show that multi-agent ergodic iSAC leads to a more time-efficient coverage than single-agent iSAC with minimum additional time and space complexity. In particular, the current implementation requires an established global communication network that allows all agents to exchange information; in the examples, we assume that ergodicity knowledge is communicated between vehicles using a wireless network.

### 4.3.2. Ergodic Control Algorithms

There are a few other algorithms that perform ergodic control in the sense that they optimize the ergodicity metric in (5.5). Mathew et al. in [**138**] derive analytic ergodic control formulas for simple linear dynamics (single and double integrator) by minimizing the Hamiltonian [**139**] in the limit as the receding time horizon goes to zero. Although closed-form, their solution is not generalizable to arbitrary nonlinear system dynamics and it also augments the state vector to include the coefficients difference so that the final system dimensionality is nominally infinite.

Miller et al. [**140**] propose an open-loop trajectory optimization technique using a finite-time horizon. This algorithm is ideal for generating optimal ergodic solutions with a prescribed time window. However, it exhibits relatively high computational cost that does not favor real-time algorithm application in a receding-horizon format. This approach has been used for offline receding-horizon exploration of unknown environments [**102**] and localization of a single static target [**101, 103**] in cases where real-time control is not imperative. De La Torre et al. [**141**] propose a stochastic differential dynamic programming algorithm for ergodic exploration in the presence of stochastic sensor dynamics.

CHAPTER 5

# Ergodic Exploration Algorithm

This chapter introduces a model predictive control algorithm, based on hybrid systems theory, that performs exploration driven by an information distribution. The algorithm is shown to meet all the requirements listed in the previous chapter. In particular, stability guarantees are provided in Section 5.3, while distributability to multiple agents is shown in Section 5.4.

## 5.1. Ergodicity

For area coverage and target localization using ergodic theory, the objective is to control an agent so that the amount of time spent in any given area of a specified search domain is proportional to the integral of a spatial distribution over that same domain. This section describes an ergodicity metric that satisfies this objective.

Consider a search domain that is a bounded $\nu$-dimensional workspace $\mathcal{X}_\nu \subset \mathbb{R}^\nu$ defined as $[0, L_1] \times [0, L_2] \times ... \times [0, L_\nu]$ with $\nu \leq n$. The spatial distribution over the search domain is denoted as $\Phi(x) : \mathcal{X}^\nu \to \mathbb{R}$, and it can represent probability of detection in search area coverage operations, such as search-and-rescue, surveillance, inspection etc. or expected information density in target localization tasks as in Chapter 8. The spatial statistics of a trajectory $x_\nu(t)$ are quantified by the percentage of time spent in each region of the workspace as

$$C(x) = \frac{1}{T} \int_{t_0}^{t_0+T} \delta[x - x_\nu(t)] dt \tag{5.1}$$

where $\delta$ is the Dirac delta. We use the distance from ergodicity between the spatial statistics of the time-averaged trajectory and the terrain spatial distribution as a metric. To drive the spatial statistics of a trajectory $x_v(t)$ to match those of the distribution $\Phi(x)$, we need to choose a norm on the difference between the distributions $\Phi(x)$ and $C(x)$. As in [**101**], we quantify the difference between the distributions, i.e., the *distance from ergodicity*, using the sum of the weighted squared distance between the Fourier coefficients $\phi_k$ of $\Phi(x)$, and the coefficients $c_k$ of the distribution $C(x)$ representing the time-averaged trajectory. In particular, the Fourier coefficients $\phi_k$ and $c_k$ are calculated respectively as

$$\phi_k = \int_{\mathcal{X}^v} \Phi(x_v) F_k(x_v) dx_v \tag{5.2}$$

and

$$c_k = \frac{1}{T} \int_{t_0}^{t_0+T} F_k(x_v(t)) dt \tag{5.3}$$

where $F_k$ is a Fourier basis function, as derived in [**138**]. Here, we use the following choice of basis function:

$$F_k(x) = \frac{1}{h_k} \prod_{i=1}^{v} \cos\left(\frac{k_i \pi}{L_i} x_i\right), \tag{5.4}$$

where $k \in \mathcal{K}$ is a set of $v$ coefficient indices $\{k_1, k_2, ..., k_v\}$ with $k_i \in \mathbb{N}$ so that $\mathcal{K} = \{k \in \mathbb{N}^v : 0 \leq k_i \leq K\}$, $K \in \mathbb{N}$ is the highest order of coefficients calculated along each of the $v$ dimensions, , and $h_k$ is a normalizing factor [**138**]. It should be noted, however, that any set of basis functions that is differentiable in the state and can be evaluated along the trajectory can be used in the derivation of the ergodic metric. Using the above, the ergodic metric on $x_v \in \mathcal{X}^v$ is defined as in [**101**, **138**, **140**]

$$\mathcal{E}(x_\nu(t)) = \sum_{k\in\mathcal{K}} \Lambda_k [c_k(x_\nu(t)) - \phi_k]^2 \tag{5.5}$$

with $\Lambda_k = \frac{1}{(1+||k||^2)^s}$ and $s = \frac{\nu+1}{2}$, which places larger weight on lower frequency information so that when $K \to \infty$ the series converges.

## 5.2. Algorithm Derivation

We shall consider nonlinear systems with input constraints such that

$$\dot{x} = f(t, x, u) = g(t, x) + h(t, x)\, u \quad \forall t \tag{5.6}$$

$$\text{with} \;\; u \in \mathcal{U} \;\; \text{and}$$

$$\mathcal{U} := \Big\{ u \in \mathbb{R}^m : u_{min} \le u \le u_{max},\; u_{min} < 0 < u_{max} \Big\},$$

i.e., systems that can be nonlinear with respect to the state vector, $x : \mathbb{R} \to \mathcal{X}$, but are assumed to be linear (or linearized) with respect to the control vector, $u : \mathbb{R} \to \mathcal{U}$. The state will sometimes be denoted as $t \mapsto x(t; x(t_i), u(\cdot))$ when we want to make explicit the dependence on the initial state (and time), and corresponding control signal. Using the metric (5.5), receding-horizon ergodic control must optimally improve the following cost at each time step $t_i$:

$$J_{\mathcal{E}} = Q \sum_{k\in\mathcal{K}} \Lambda_k \Big[ \underbrace{\frac{1}{t_i + T - t_0^{erg}} \int_{t_0^{erg}}^{t_i+T} F_k(x(t))dt}_{c_k^i} - \phi_k \Big]^2 \tag{5.7}$$

where $t_0^{erg}$ is the user-defined initial time of ergodic exploration, $x \in \mathcal{X}^\nu$ and $Q \in \mathbb{R}$ weights the ergodic cost against control effort weighted by $R$ in (5.21). Henceforth, for brevity we refer to the trajectory of the set of states to be ergodically explored as $x(t)$ instead of $x_v(t)$, although it should be clear that the ergodically explored states might or might not be all the states of the system dynamics (i.e., $v \leq n$).

To understand the challenges of optimizing (5.7), we distinguish between the dynamic states of the controlled system, $x \in \mathbb{R}^n$—e.g., the 12 states denoting position and heading in quadrotor dynamics—and the information states $c_k(x(\cdot))$ in (5.3), i.e., the parameterized time-averaged statistics of the followed trajectory over a finite time duration. The main difficulty in optimizing ergodicity is that the ergodic cost functional in (5.7) is non-quadratic and does not follow the Bolza form—consisting of a running and terminal cost [**142**]—with respect to the dynamic states. To address this, infinite-dimensional trajectory optimization methods that are independent of the cost functional form have been employed [**101**] to optimize ergodicity. However, the computational cost of such iterative methods is prohibitive for real-time control in a receding-horizon approach. Another method involves change of coordinates so that the cost functional is rendered quadratic with respect to the information states parameterized by Fourier coefficients. This allows the use of traditional optimal control approaches e.g., LQR, DDP, SQP etc. (see for example [**141**]). However, this approach entails optimization over an extended set of states (the number of parameterized information states is usually significantly larger than the dynamic states) which inhibits real-time execution. In addition, and perhaps more importantly, defining a running cost on the information states results in unnecessarily repetitive integration of the dynamic state trajectories. To avoid this, an option would be to optimize a terminal cost only, but this proves problematic in establishing stability of Model Predictive Control (MPC) algorithms (see [**74**]).

To overcome the aforementioned issues, we seek to formulate an algorithm that a) computes control actions that guarantee contraction of the ergodic cost at each time step b) naturally uses current sensor feedback to compute controls fast, in real time. For these reasons, we choose to frame the control problem as a hybrid control problem, similarly to Sequential Action Control (SAC) in [**81**, **84**]. By doing this, we are able to formulate an ergodic control algorithm that is rendered fast enough for real time operation—as opposed to traditional model predictive control algorithms that are usually computationally expensive [**54**]—for two main reasons: a) a single control action is calculated at every time step using a closed-form algebraic expression and b) this control action aims to optimally improve ergodicity (instead of optimizing it) by an amount that guarantees stability with respect to $x$ and $c_i$.

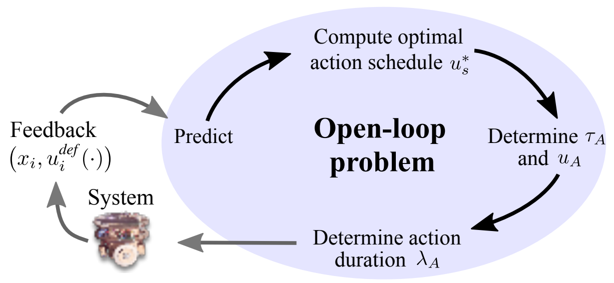
**Algorithm 4** Receding-horizon ergodic exploration (RHEE)

```
***Inputs:*** initial time $t_0$, initial state $x_0$, terrain spatial distribution $\Phi(x)$, ergodic initial time $t_0^{erg}$, final time $t_f$
***Output:*** closed-loop ergodic trajectory $\bar{x}_{cl} : [t_0, t_f] \rightarrow \mathcal{X}$
**Define** ergodic cost weight $Q$, highest order of coefficients $K$, control weight $R$, search domain bounds $\{L_1, ..., L_v\}$, sampling time $t_s$, desired rate of change $\alpha_d$, time horizon $T$.
**Initialize** nominal control $u^{nom}$, step $i = 0$.
■ Calculate $\phi_k$ using (5.2).
■ While $t_i < t_f$
    □ Solve open-loop problem $\mathcal{P}_{\mathcal{E}}(t_i,\ x_i, T, J_{\mathcal{E}})$ to get $u_i^*$:
        (1) Simulate system (5.6) for $t \in [t_i, t_i + T]$ under $u_i^{def}$ to get $x(t)$.
        (2) Simulate (5.12) for $t \in [t_i, t_i + T]$.
        (3) Compute $u_{sched}^*$ using (5.22).
        (4) Determine action application time $t_A$ and value $u_A$ by minimizing (5.23).
        (5) Determine action duration $\lambda_A$ using the line search process in Section 5.2.0.4 and the condition in (5.9) with $C_{\mathcal{E}}$ in (5.26).
    □ Apply $u_i^*$ to (5.6) for $t \in [t_i, t_i + t_s]$ to get $\bar{x}_{cl} \forall t \in [t_i, t_i + t_s]$.
    □ Define $t_{i+1} = t_i + t_s$, $x_{i+1} = \bar{x}_{cl}(t_{i+1})$.
    □ $i \leftarrow i + 1$
end while
```

An overview of the algorithm is given in Algorithm 4. Once the Fourier coefficients $\phi_k$ of the spatial distribution $\Phi(x)$ have been calculated, the algorithm follows a receding-horizon approach; controls are obtained by repeatedly solving online an open-loop ergodic control problem $P_{\mathcal{E}}$ every $t_s$ seconds (with sampling frequency $1/t_s$), every time using the current measure of the system dynamic state $x$. The following definitions are necessary before introducing the open-loop problem.

**Definition 4.** *An action $A$ is defined by the triplet consisting of a control's value, $u_A \in \mathcal{U}$, application duration, $\lambda_A \in \mathbb{R}^+$ and application time, $\tau_A \in \mathbb{R}$, such that $A := \{u_A, \lambda_A, \tau_A\}$.*

**Definition 5.** *Nominal control $u^{\text{nom}} : \mathbb{R} \to \mathcal{U}$, provides a nominal trajectory around which the algorithm provides feedback. When applying ergodic control as a standalone controller, $u^{\text{nom}}(\cdot)$ is either zero or constant. Alternatively, $u^{\text{nom}}(\cdot)$ may be an optimized feedforward or state-feedback controller.*



Compute optimal action schedule $u_s^*$
Determine $\tau_A$ and $u_A$
Determine action duration $\lambda_A$
System
Feedback $(x_i, u_i^{def}(\cdot))$
Predict
Open-loop problem

Figure 5.1. An overview of the ergodic control process. One major difference between the proposed ergodic control algorithm and traditional MPC approaches is that the open-loop problem can be solved without employing nonlinear programming solvers [**7**] by using hybrid systems theory. In order to solve (5.8), the algorithm follows four steps as illustrated above.

The open-loop problem $\mathcal{P}_{\mathcal{E}}$ that is solved in each iteration of the receding horizon strategy can now be defined as follows[1].

$$\mathcal{P}_{\mathcal{E}}(t_i,\ x_i, T, J): \tag{5.8}$$

Find action $A$ such that

$$J_{\mathcal{E}}(x(t; x_i, u_i^*(\cdot))) - J_{\mathcal{E}}(x(t; x_i, u_i^{def}(\cdot))) < C_{\mathcal{E}} \tag{5.9}$$

subject to

$$u_i^*(t) = \begin{cases} u_A & \tau_A \leq t \leq \tau_A + \lambda_A \\ u_i^{def}(t) & \text{else} \end{cases},$$

and (5.6) with $t \in [t_i, t_i + T]$ and $x(t_i) = x_i$.

where $C_{\mathcal{E}}$ is a quantity that guarantees stability (see Section 5.3) and $u_i^{def}$ and $x_i^{def}$ are defined below.

[1]From now on, subscript $i$ will denote the $i$-th time step, starting from $i = 0$.

**Definition 6.** *Default control $u_i^{\text{def}} : [t_i, t_i + T] \to \mathcal{U}$, is defined as*

$$u_i^{\text{def}}(t) = \begin{cases} u_{i-1}^*(t) & t_i \le t \le t_i + T - t_s \\ u_i^{\text{nom}}(t) & t_i + T - t_s < t \le t_i + T \end{cases}, \tag{5.10}$$

with $u_0^{def}(\cdot) \equiv u_0^{nom}(\cdot)$, $u_{i-1}^* : [t_{i-1}, t_{i-1} + T] \to \mathcal{U}$ the output of $\mathcal{P}_{\mathcal{E}}(t_{i-1}, x_{i-1}, T, J)$ from the previous time step $i-1$—corresponding to $x_{i-1}^*(\cdot)$—and $t_s = t_i - t_{i-1}$ the sampling period (Fig. 5.1c). The system trajectory corresponding to application of default control will be denoted as $x(t; x(t_i), u^{def}(\cdot))$ or $x^{def}(\cdot)$ for brevity. The following proposition is necessary before going though the steps for solving $\mathcal{P}_{\mathcal{E}}$.

**Proposition 6.** *Consider the case where the system* (5.6) *evolves according to default control dynamics $f(t, x(t), u_i^{\text{def}}(t))$, and action $u_A$ is applied at time $\tau$ (dynamics switch to $f(t, x(t), u_A)$) for an infinitesimal duration $\lambda \to 0$ before switching back to default control. In this case, the ergodic mode insertion gradient $\frac{\partial J_{\mathcal{E}}}{\partial \lambda}$ evaluated at $t = \tau$ measures the first-order sensitivity of the ergodic cost* (5.7) *to infinitesimal application of control action $u_A$ and is calculated as*[2]

$$\left.\frac{\partial J_{\mathcal{E}}}{\partial \lambda}\right|_{\tau} = \rho_{\mathcal{E}}(\tau) \left[ f(\tau, x_i^{\text{def}}(\tau), u_A) - f(\tau, x_i^{\text{def}}(\tau), u_i^{def}(\tau)) \right] \tag{5.11}$$

*with*

$$\dot{\rho}_{\mathcal{E}} = -\ell(t, x_i^{\text{def}})^T - D_2 f(t, x_i^{\text{def}}, u_i^{\text{def}})^T \cdot \rho_{\mathcal{E}} \tag{5.12}$$

$$\textit{subject to} \;\; \rho_{\mathcal{E}}(t_i + T) = \mathbf{0}$$

$$\textit{and} \quad \ell(t, x) = \frac{2Q}{t_i + T - t_0^{erg}} \sum_{k \in \mathcal{K}} \left\{ \Lambda_k [c_k^i - \phi_k] \frac{\partial F_k(x(t))}{\partial x(t)} \right\}.$$

[2]$D_i$ denotes derivative with respect to $i^{th}$ argument.

**Proof.** The proof of Proposition 6 is as follows. To make explicit the dependence on action $A$, we write inputs $u : \mathbb{R} \times \mathbb{R}^{+} \times \mathbb{R} \times U \to U$ of the form of $u_i^*(t)$ in (5.8) as

$$u(t; \lambda_A, \tau_A, u_A) = \begin{cases} u_A & \tau_A \leq t \leq \tau_A + \lambda_A \\ u_i^{def} & \text{else.} \end{cases}$$

When $\lambda_A = 0$, it is $u(t; 0, \cdot, \cdot) \equiv u_i^{def}$, i.e., no action is applied. Accordingly, we define $\bar{J}_{\mathcal{E}}(\lambda_A, \tau_A, u_A) := J_{\mathcal{E}}(x(t; t_0, x_0, u(t; \lambda_A, \tau_A, u_A)))$ so that the performance cost depends directly on the application parameters of an iSAC action. Assuming $t_0^{erg} = t_0$ and defining $\beta := \frac{1}{t_i+T-t_0} \int\limits_{t_0}^{t_i+T} F_k(x(t))dt - \phi_k$, it is

$$\frac{\partial J_{\mathcal{E}}}{\partial \lambda} = \frac{\partial J_{\mathcal{E}}}{\partial \beta} \frac{\partial \beta}{\partial \lambda} \tag{5.13}$$

where

$$\frac{\partial J_{\mathcal{E}}}{\partial \beta} = 2Q \sum_{k \in \mathcal{K}} \Lambda_k \Big[ \underbrace{\frac{1}{t_i + T - t_0} \int\limits_{t_0}^{t_i+T} F_k(x(\sigma))d\sigma - \phi_k}_{c_k^i} \Big] \tag{5.14}$$

and

$$\frac{\partial \beta}{\partial \lambda} = \frac{1}{t_i + T - t_0} \int\limits_{\tau}^{t_i+T} \frac{\partial F_k(x(t))}{\partial x(t)} \frac{\partial x(t)}{\partial \lambda} dt \tag{5.15}$$

where the integral boundary changed from $t_0$ to $\tau$ because the derivative of $x(t)$ with respect to $\lambda$ is zero when $t < \tau$. Then, expression (5.13) can be rearranged, pulling $\frac{\partial J_{\mathcal{E}}}{\partial \beta}$ into the integral over $t$, and switching the order of the integral and summation, to the following:

$$\frac{\partial J_{\mathcal{E}}}{\partial \lambda} = \int_{\tau}^{t_i+T} \underbrace{\frac{2Q}{t_i + T - t_0} \sum_{k \in \mathcal{K}} \left\{ \Lambda_k [c_k^i - \phi_k] \frac{\partial F_k(x(t))}{\partial x(t)} \right\}}_{\ell(t,x)} \frac{\partial x(t)}{\partial \lambda} dt \tag{5.16}$$

with[3]

$$\frac{\partial x(t)}{\partial \lambda} = \Phi(t,\tau)\Big[ f(\tau, x(\tau), u_A) - f(\tau, x(\tau), u_i^{def}(\tau)) \Big] \tag{5.17}$$

where $\Phi(t,\tau)$ is the state transition matrix of the linearized system dynamics (5.6) with $A = D_x f$. Therefore,

$$\frac{\partial J_{\mathcal{E}}}{\partial \lambda} = \int_{\tau}^{t_i+T} \ell(t,x) \cdot \Phi(t,\tau) dt \ \cdot \Big[ f(\tau, x(\tau), u_A) - f(\tau, x(\tau), u_i^{def}(\tau)) \Big]. \tag{5.18}$$

Finally, notice that $\int_{\tau}^{t_i+T} \ell(t,x) \cdot \Phi(t,\tau) dt$ is the convolution equation for the system

$$\dot{\rho}_{\mathcal{E}} = -\ell(t,x)^T - D_2 f\Big(t, x, u_i^{def}\Big)^T \rho_{\mathcal{E}} \tag{5.19}$$

$$\text{subject to } \ \rho_{\mathcal{E}}(t_i + T) = \mathbf{0}$$

where $\ell$ is defined in (5.16). Therefore, we end up with the expression for the mode insertion gradient of the ergodic cost at time $\tau$:

$$\left.\frac{\partial J_{\mathcal{E}}}{\partial \lambda}\right|_{\tau} = \rho_{\mathcal{E}}(\tau) \Big[ f(\tau, x(\tau), u_A) - f(\tau, x(\tau), u_i^{def}(\tau)) \Big]. \tag{5.20}$$

This concludes the proof. □

[3]Expression (5.17) is a direct result of applying the fundamental theorem of calculus on the system equations (5.6).

The steps for solving $\mathcal{P}_{\mathcal{E}}$ are then listed and explained in the following.

**5.2.0.1. Predict.** In this step, the algorithm evaluates the system (5.6) from the current state $x_i$ and time $t_i$, with $u_i^{def}(t)$ for $t \in [t_i,\ t_i + T]$. In addition, it uses the predicted state trajectory to backward simulate $\rho_{\mathcal{E}}$ that satisfies (5.12).

**5.2.0.2. Compute optimal action schedule $u_s^*(\cdot)$.** In this step, we compute a schedule $u_s^* : [t_i, t_i + T] \to \mathbb{R}^m$ which contains candidate infinitesimal actions. Specifically, $u_s^*(\cdot)$ contains candidate action values and their corresponding application times, but assumes $\lambda \to 0^+$ for all. The final $u_A$ and $\tau_A$ will be selected from these candidates in step three of the solution process such that $u_A = u_s^*(\tau_A)$, while a finite duration $\lambda_A$ will be selected in the final step. The optimal action schedule $u_s^*(\cdot)$ is calculated by minimizing

$$J_{u_s} = \frac{1}{2}\int_{t_i}^{t_i+T} \left[\frac{dJ_{\mathcal{E}}}{d\lambda}(t) - \alpha_d\right]^2 + \|u_s(t)\|_R^2\, dt, \tag{5.21}$$

$$\frac{dJ_{\mathcal{E}}}{d\lambda}(t) = \rho_{\mathcal{E}}(t)^T \left[f(t, x_i^{def}(t), u_s(t)) - f(t, x_i^{def}(t), u_i^{def}(t))\right]$$

where the quantity $\frac{dJ_{\mathcal{E}}}{d\lambda}(\cdot)$ (see Proposition 6), called the mode insertion gradient [**62**], denotes the rate of change of the cost with respect to a switch of infinitesimal duration $\lambda$ in the dynamics of the system. In this case, $\frac{dJ_{\mathcal{E}}}{d\lambda}(\cdot)$ shows how the cost will change if we introduce a single infinitesimal switch from $f(t, x_i^{def}(t), u_i^{def}(t))$ to $f(t, x_i^{def}(t), u_s(t))$ at some point in the time window $[t_i, t_i + T]$. The parameter $\alpha_d \in \mathbb{R}^-$ is user specified and allows the designer to influence how aggressively each action value in the schedule $u_s^*(t)$ improves the cost.

Based on the evaluation of the dynamics (5.6), and (5.12) completed in the prediction step (Section 5.2.0.1), minimization of (5.21) leads to the following closed-form expression for the optimal action schedule:

$$u_s^*(t) = (\Lambda + R^T)^{-1} \left[\Lambda\, u_i^{def}(t) + h\big(t, x_i^{def}(t)\big)^T \rho_{\mathcal{E}}(t)\, \alpha_d\right], \tag{5.22}$$

where $\Lambda \triangleq h\big(t, x_i^{def}(t)\big)^T \rho_{\mathcal{E}}(t) \rho_{\mathcal{E}}(t)^T h\big(t, x_i^{def}(t)\big)$. The infinitesimal action schedule can then be directly saturated to satisfy any min/max control constraints of the form $u_{min,k} < 0 < u_{max,k}\ \forall k \in \{1, \ldots, m\}$ such that $u_s^* \in \mathcal{U}$ without additional computational overhead (see [**81**] for proof).

**5.2.0.3. Determine application time $\tau_A$ (and thus $u_A$ value).** Recall that the curve $u_s^*(\cdot)$ provides the values and application times of possible infinitesimal actions that the algorithm could take at different times to optimally improve system performance from that time. In this step the algorithm chooses one of these actions to apply, i.e., chooses the application time $\tau_A$ and thus an action value $u_A$ such that $u_A = u_s^*(\tau_A)$. To do that, $u_s^*(\cdot)$ is searched for a time $\tau_A$ that minimizes

$$J_t(\tau) = \left.\frac{dJ_{\mathcal{E}}}{d\lambda}\right|_{\tau}, \tag{5.23}$$

$$\left.\frac{dJ_{\mathcal{E}}}{d\lambda}\right|_{\tau} = \rho_{\mathcal{E}}(\tau)^T \left[f(\tau, x_i^{def}(\tau), u_s^*(\tau)) - f(\tau, x_i^{def}(\tau), u_i^{def}(\tau))\right]$$

$$\text{subject to } \tau \in [t_i, t_i + T].$$

Notice that the cost (5.23) is actually the ergodic mode insertion gradient evaluated at the optimal schedule $u_s^*(\cdot)$. Thus, minimization of (5.23) is equivalent to selecting the infinitesimal action from $u_s^*(\cdot)$ that will generate the greatest cost reduction relative to only applying default control.

**5.2.0.4. Determine control duration $\lambda_A$.** The final step in synthesizing an ergodic control action is to choose how long to act, i.e., a finite control duration $\lambda_A$, such that condition (5.9) is satisfied. From [**62, 143**], there is a non-zero neighborhood around $\lambda \to 0^+$ where the mode insertion gradient models the change in cost in (5.9) to first order, and thus, a finite duration $\lambda_A$

exists that guarantees descent. In particular, for *finite* durations $\lambda$ in this neighborhood we can write

$$J_{\mathcal{E}}(x(t;x_i,u_i^*(\cdot))) - J_{\mathcal{E}}(x(t;x_i,u_i^{nom}(\cdot))) \tag{5.24}$$
$$= \Delta J_{\mathcal{E}} \approx \frac{dJ_{\mathcal{E}}}{d\lambda}\bigg|_{\tau_A} \lambda.$$

Then, a finite action duration $\lambda_A$ can be calculated by employing a *line search* process [**143**].

After computing the duration $\lambda_A$, the control action $A$ is fully specified (it has a value, an application time and a duration) and thus the solution $u_i^*(t)$ of problem $\mathcal{P}_{\mathcal{E}}$ has been determined. By iterating on this process (Section 5.2.0.1 until Section 5.2.0.4), we rapidly synthesize piecewise continuous, input-constrained ergodic control laws for nonlinear systems.

**Algorithm 5** Reactive RHEE for varying $\Phi(x)$

```
**Define** ergodic memory $M_{erg}$, distribution sampling time $t_\phi$.
**Initialize** current time $t_{curr}$, current state $x_{curr}$.
While $t_{curr} < \infty$
    (1) Receive/compute current $\Phi_{curr}(x)$.
    (2) $t_0^{erg} \leftarrow t_{curr} - M_{erg}$
    (3) $t_{final} \leftarrow t_{curr} + t_\phi$
    (4) $\bar{x}_{cl} = RHEE(t_{curr}, x_{curr}, t_0^{erg}, t_{final}, \Phi_{curr}(x))$
    (5) $t_{curr} \leftarrow t_{curr} + t_\phi$
    (6) $x_{curr} = \bar{x}_{cl}(t_{final})$
end while
```

The Receding-Horizon Ergodic Exploration (RHEE) process for a dynamically varying $\Phi(x)$ is given in Algorithm 5. The re-initialization of RHEE when a new $\Phi(x)$ is available serves two purposes: first, it allows for the new coefficients $\phi_k$ to be calculated[4] and second, it allows the update of the ergodic initial time $t_0^{erg}$.

The ergodic initial time $t_0^{erg}$ is particularly important for the algorithm performance because it regulates how far back in the past the algorithm should look when comparing the spatial statistics of the trajectory (parameterized by $c_k$) to the input spatial distribution (parameterized by $\phi_k$). If the spatial distribution is regularly changing to incorporate new data (for example in the case that the distribution represents expected information density in target localization as we will see in Chapter 8), it is undesirable for the algorithm to use state trajectory data all the way since the beginning of exploration. At the same time, "recently" visited states must be known to the algorithm so that it avoids visiting them multiple times during a short time frame. To specify our notion of "recently" depending on the application, we use the parameter $M_{erg}$ in Algorithm 5 which we call "ergodic memory" and simply indicates how many time units in the past the algorithm has access to, so that it is $t_0^{erg} = t_0 - M_{erg}$ every time RHEE is re-initialized at time $t_0$.

Remarks on computing $c_k^i$. The cost $J_\mathcal{E}$ in (5.7) depends on the full state trajectory from a defined initial time $t = t_0^{erg} \leq t_i$ in the past (instead of $t_i$ as in common tracking objectives) to $t = t_i + T$, which could arise concerns with regard to execution time and computational cost. Here, we show how to compute $c_k^i$ in a way that avoids integration over an infinitely increasing time duration as $t_i \rightarrow \infty$. To calculate trajectory coefficients $c_k^i$ at time step $t_i$ with $k \in \mathcal{K}$, and thus cost $J_\mathcal{E}$, notice that:

[4]This corresponds to the general case when the distribution time evolution is unknown. If, however, $\Phi(x)$ is a time-varying distribution with known evolution in time, we can pre-calculate the coefficients $\phi_k$ offline to reduce the computational cost further.

$$c_k^i = \frac{1}{t_i + T - t_0^{erg}} \int_{t_0^{erg}}^{t_i+T} F_k(x(t))dt = \tag{5.25}$$

$$= \underbrace{\frac{1}{t_i + T - t_0^{erg}} \int_{t_0^{erg}}^{t_i} F_k(x(t))dt}_{\bar{c}_k^{(i)}} + \frac{1}{t_i + T - t_0^{erg}} \int_{t_i}^{t_i+T} F_k(x(t))dt$$

where recursively

$$\bar{c}_k^{(i)} = \frac{t_{i-1} + T - t_0^{erg}}{t_i + T - t_0^{erg}} \bar{c}_k^{(i-1)} + \frac{1}{t_i + T - t_0^{erg}} \int_{t_{i-1}}^{t_i} F_k(x(t))dt$$

$\forall \; i \geq 1, \;\; k \in \mathcal{K}$ with $\bar{c}_k^{(0)} = 0$.

Therefore, only the current open-loop trajectory $x(t)$ for all $t \in [t_i, t_i + T]$ and a set of $(K+1)^v$ coefficients $\bar{c}_k^{(i)} \in \mathbb{R}$ are needed for calculation of $c_k^i$ and thus $J_{\mathcal{E}}$ at the $i^{th}$ time step. Coefficients $\bar{c}_k^{(i)} \in \mathbb{R}$ can be updated at the end of the $i^{th}$ time step and stored for use in the next time step at $t_{i+1}$. This provides the advantage that although the cost depends on an integral over time with an increasing upper limit, the amount of stored data needed for cost calculation does not increase but remains constant as time progresses.

## 5.3. Stability

In this section, we establish the requirements for ergodic stability of the closed-loop system resulting from the receding-horizon strategy in Algorithm 4. To achieve closed-loop stability for Algorithm 4, we apply a contractive constraint [**144–147**] on the cost. Contractive constraints have been widely used in the MPC literature to show closed-loop stability as an alternative to methods relying on a terminal (region) constraint [**54**, **105**, **148**]. First, we define stability in the ergodic sense[5].

**Definition 7.** *Let $\mathcal{X}^v \subset \mathbb{R}^v$ be the set of states to be ergodically explored. The closed-loop solution $x_v(t) : \mathbb{R} \to \mathcal{X}^v$ resulting from an ergodic control strategy applied on* (5.6) *is ergodically stable if the difference $C(x) - \Phi(x)$ for all $x$ with $C(x)$ defined in* (5.1) *(see Section 5.1) converges to a zero density function* $0(x)$. *Using Fourier parameterization as shown in equations* (5.2) *and* (5.3)*, this requirement is equivalent to $c_k(x_v) - \phi_k(x_v) \to 0$ for all $k$ as $t \to \infty$.*

The following assumptions will be necessary in proving stability.

**Assumption 2.** *The dynamics $f$ in* (5.6) *are continuous in $u$, piecewise-continuous in $t$, and continuously differentiable in $x$. Also, $f$ is compact, and thus bounded, on any given compact sets $\mathcal{X}$ and $\mathcal{U}$. Finally, $f(\cdot, 0, 0) = 0$.*

**Assumption 3.** *Let $Q$ be the set of trajectories $x(\cdot) : \mathbb{R} \to \mathcal{X}$ in* (5.6) *and $Q_d \subset Q$ the subset that satisfies $c_k(x(\cdot)) - \phi_k = 0$ for all $k$. Also, suppose $L(x(\cdot), u, t) : Q \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ is defined as follows:*

$$
\begin{aligned}
L(x(\cdot), u, t) :=& \frac{2Q}{t - t_0^{erg}} \sum_k \Bigg\{ \Lambda_k \Big[ c_k(x(\cdot), t) - \phi_k \Big] \cdot \\
&\cdot \Big[ F_k(x(t)) - c_k(x(\cdot), t) + f(x(t), u, t)^T \int_{t_0^{erg}}^{t} \frac{\partial F_k(x(s))}{\partial x(s)} ds \Big] \Bigg\}
\end{aligned}
$$

[5]Note how this definition differs from the definition of asymptotic stability about an equilibrium point as we now refer to stability of a motion instead of stability of a single point.

*where $c_k(x(\cdot), t) = \frac{1}{t-t_0^{erg}} \int_{t_0^{erg}}^{t} F_k(x(s))ds$ denote the Fourier-parameterized spatial statistics of the state trajectory up to time t. Through simple computation, we can verify that $L(x_d(\cdot), 0, \cdot) = 0$ when $x_d(\cdot) \in \mathcal{Q}_d$. Then, there is a continuous positive definite—with respect to the set $\mathcal{Q}_d$—and radially unbounded function $\mathcal{M} : \mathcal{Q} \times \mathbb{R} \to \mathbb{R}_+$ such that $L(x(\cdot), u, t) \geq \mathcal{M}(x(\cdot), t)$ for all $u \in \mathbb{R}^m$.*

**Assumption 4.** *The ergodic open-loop problem improves the ergodic cost at each time step by an amount specified by the condition* (5.9) *with $C_{\mathcal{E}}$ defined as*

$$C_{\mathcal{E}} = -\int_{t_{i-1}+T}^{t_i+T} L(x_i^{\text{def}}(\cdot), u_i^{\text{def}}(t), t)\, dt. \tag{5.26}$$

**Theorem 1.** *Let assumptions 1-3 hold for all time steps i. Then, the closed-loop system resulting from the receding-horizon ergodic control strategy is ergodically stable in the sense that $c_k(x_v) - \phi_k(x_v) \to 0$ for all k as $t \to \infty$.*

**Proof.** Note that the ergodic metric (5.7) can be written as $J_{\mathcal{E}} = \mathcal{B}(t_i+T, x(\cdot))$ with $\mathcal{B}(t, x(\cdot)) := Q \sum_{k \in \mathcal{K}} \Lambda_k \Big[ c_k(x(\cdot), t) - \phi_k \Big]^2$ where $c_k(x(\cdot), t) = \frac{1}{t-t_0^{erg}} \int_{t_0^{erg}}^{t} F_k(x(s))ds$. Using this definition and converting $J_{\mathcal{E}}$ from Mayer to Lagrange form yields $J_{\mathcal{E}} = \int_{t_0^{erg}}^{t_i+T} L(x(\cdot), u, t)dt$ with $L(x(\cdot), u, t) = \frac{d}{dt}\mathcal{B}(t, x(\cdot))$ resulting in the expression in Assumption 3. Going back to Assumption 4 and the ergodic open-loop problem (5.8) in Section 5.2, we note that condition (5.9) with $C_{\mathcal{E}}$ in (5.26) is a contractive constraint applied in order to generate actions that sufficiently improve the cost between time steps. To see that this is true, one can rewrite (5.9) as

$$J_{\mathcal{E}}(x_i^*(\cdot)) - J_{\mathcal{E}}(x_{i-1}^*(\cdot)) \leq -\int_{t_{i-1}}^{t_i} L(x_{i-1}^*(\cdot), u_{i-1}^*(t), t)\, dt \tag{5.27}$$

since from (5.10), $u_i^{def}(t) \equiv u_{i-1}^*(t)$ in $[t_i, t_{i-1} + T]$. This contractive constraint directly proves that the integral $\int_{t_0^{erg}}^{t} \mathcal{M}(x(\cdot), s)ds$ is bounded for $t \to \infty$, which, according to a well known lemma found, e.g., in [**149**], guarantees asymptotic convergence, i.e., that $x(\cdot) \to Q_d$ or equivalently that $c_k(x(\cdot)) - \phi_k \to 0$ as $t \to \infty$. □

## 5.4. Extension to Multi-Agent Control

Assume we have $N$ number of agents $\zeta = 1, ..., N$, each with its own computation, collectively exploring a terrain to track an assigned spatial distribution $\Phi(x)$. Each agent $\zeta$ performs RHEE as described in Algorithm 4. At the end of each algorithm iteration $i$ (and thus every $t_s$ seconds), each agent $\zeta$ communicates the Fourier-parameterized statistics of their exploration trajectories $c_{k,\zeta}^i$ up to time $t_i$ to all the other agents. By communicating this information, the agents have knowledge of the collective coverage up to time $t_i$ and can use this to avoid exploring areas that have already been explored by other agents. This ensures that the exploration process is coordinated so that the spatial statistics of the combined agent trajectories collectively match the distribution.

To use this information, each agent $\zeta$ updates its trajectory coefficients $c_{k,\zeta}^i$ at time $t_i$ to include the received coefficients $c_{k,j}^{i-1}$ from the previous algorithm iteration $i - 1$ so that now the collective agent coefficients $\mathbf{c}_{k,\zeta}^i$ are defined to be:

$$\mathbf{c}_{k,\zeta}^i = c_{k,\zeta}^i + \frac{1}{N-1} \cdot \sum_{j=1, j\neq\zeta}^{N} c_{k,j}^{i-1} \tag{5.28}$$

where $c_{k,\zeta}^i$ are the coefficients of the agent $\zeta$ state trajectory at time step $t_i$ calculated as in (5.25), and $c_{k,j}^{i-1}$ are the coefficients of the remaining agents state trajectories at the previous time step $t_{i-1}$ also calculated as in (5.25). So now, agent $\zeta$ computes the ergodic cost (5.7) at $t_i$ based on all the agents' past trajectories and Algorithm 4 is guaranteed to compute a control action that will optimally improve it. Note that expression (5.28) expands to:

$$\mathbf{c}^i_{k,\zeta} = \frac{1}{t_i + T - t_0^{erg}} \int_{t_0^{erg}}^{t_i+T} F_k(x_\zeta(t))dt + \tag{5.29}$$
$$\frac{1}{(N-1)(t_{i-1} + T - t_0^{erg})} \sum_{j=1, j\neq\zeta}^{N} \int_{t_0^{erg}}^{t_{i-1}+T} F_k(x_j(t))dt$$

with $\mathbf{c}^0_{k,\zeta} = \frac{1}{t_i+T-t_0^{erg}} \int_{t_0^{erg}}^{t_0+T} F_k(x_\zeta(t))dt$ where $x_\zeta(t)$ with $\zeta = 1, ..., N$ is the agent $\zeta$ state trajectory. Therefore expression (5.28) calculates the combined statistics of the current agent's trajectory $x_\zeta(t)$, $\forall t \in [t_0^{erg}, t_i + T]$ and of the state trajectories that all the other agents have executed up to current time $t_i$ and temporarily intend to execute from $t_i$ (now) to $t_{i-1} + T$ based on their open-loop trajectories at the previous time step $t_{i-1}$.

*Computational complexity and communication requirements:* This multi-agent ergodic control process exhibits time complexity $\mathcal{O}(1)$ (i.e., the amount of time required for one algorithm cycle does not scale with $N$) because Algorithm 1 is executed by each agent in *parallel* in a distributed manner. Computational complexity also remains constant for each agent ($\mathcal{O}(1)$, i.e., the total number of computer operations does not scale with $N$). However, each agent's computational unit needs to communicate with a central transmitter/receiver through a (deterministic) communication channel with receiving capacity that scales linearly in $N$ ($\mathcal{O}(N)$) and with constant transmitting capacity ($\mathcal{O}(1)$). In particular, at every time step $t_i$, each agent needs to receive $(K + 1)^\nu$ coefficients corresponding to $c^{i-1}_{k,j}$, by each of the remaining $N - 1$ agents. In addition, each agent is responsible to transmit their own $(K + 1)^\nu$ coefficients to the rest of the robot

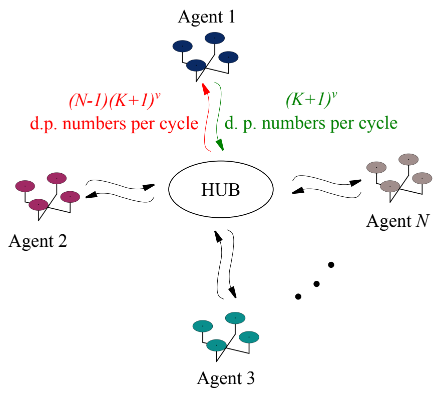


Figure 5.2. Communication network for multi-agent ergodic exploration using a hub configuration. Agents are equipped with independent computational units for local control calculation but exchange information that may influence each other's subsequent actions. The HUB is simply a network component and has no computational capacity. Assuming that a double precision (d.p.) number has 64 bits and an algorithm cycle completes in $t_s$ seconds, the transmitting bit rate of each individual communication channel should be at least $\frac{(K+1)^v \cdot 64}{t_s}$ bits/s and receiving bit rate equal or higher than $(N-1)\frac{(K+1)^v \cdot 64}{t_s}$ bits/s.

network (see Fig. 5.2). Thus, assuming a constant highest order of coefficients $K$ and number of ergodic variables $v$, transmitting capacity of each agent's communication channel is constant while its receiving capacity scales linearly with $N$. While, in this communication paradigm, we assumed a star network configuration (Fig. 5.2), note that a fully connected network can also be employed. In any case, the minimum amount of information needed by the team of agents for coordinated exploration is $N$ sets of $(K+1)^v$ coefficients per algorithm cycle. Because of this requirement of collective data exchange between agents at each time step, multi-agent ergodic exploration can be characterized as semi-distributed in that each agent executes RHEE (Algorithm 4) independently but shares information with the other agents after each algorithm cycle.

CHAPTER 6

# Area Search and Coverage using Distribution-Driven Exploration

An area search function is required by many operations, including search-and-rescue [**119**, **120**], hazard detection [**108**], agricultural spraying [**121**], solar radiation tracking [**106**] and monitoring of environmental phenomena, such as water quality in lakes [**122**]. In addition, complete area coverage navigation that requires the agent to pass through every region of the workspace [**123**] is an essential issue for cleaning robots [**114**, **124**], autonomous underwater covering vehicles [**125**, **126**], demining robots [**127**], automated harvesters [**128**], etc. Although slightly different in concept, both applications—search and coverage—involve motion planning for tracking a distribution of information across the terrain. For purposes of area search, this terrain spatial distribution indicates probability of detection and usually exhibits high variability in both space and time as it dynamically absorbs new information about the target's whereabouts. In area coverage applications, on the other hand, the terrain distribution shows regions of interest and is normally near-uniform with possible occlusions.

This Chapter provides an extensive list of examples where Algorithm 4 performs area search and coverage. The examples aim to highlight the efficiency of the algorithm and its adaptability to a variety of search scenarios, including agents with different dynamical features, time-varying probabilities-of-detection and multi-agent coverage.

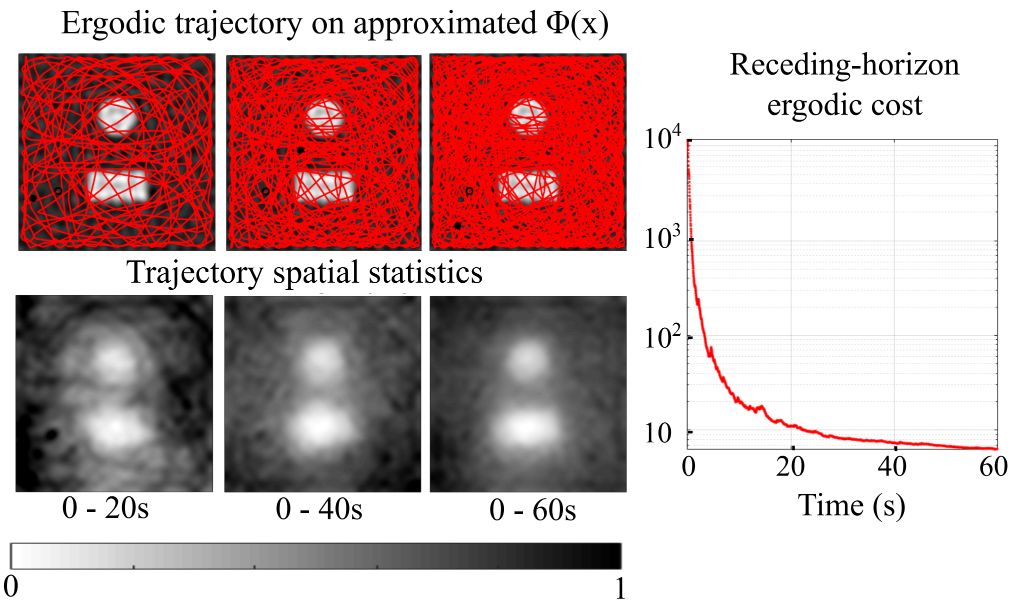


Figure 6.1. Ergodic area coverage in an occluded environment (Algorithm 4 with time horizon $T = 0.1s$ and sampling time $t_s = 0.02s$). White regions in $\Phi(x)$ (top row) indicate low to no probability of detection (occlusions), for example due to sensor failure or physical entities obscuring visibility. Note that occlusions are not obstacles that should be completely avoided. Bottom row shows the spatial statistics $\Phi_x^i(x)$ of the followed trajectory from $t = 0$ to $t = t_i$ calculated as $\Phi_x^i(x) = \sum_{k=\{0\}^\nu}^{\{K\}^\nu} \{\Lambda_k c_k^i F_k(x)\}$ with $\nu = 2$ and $K = 20$. By the end of the simulation at $t = 60$, the trajectory spatial statistics $\Phi_x^{60}(x)$ closely match the initial terrain spatial distribution $\Phi(x)$, accomplishing the objective of ergodicity as expected. The ergodic cost (5.7) is shown to decrease on logarithmic scale over time. Small cost fluctuations result from numerical errors.

## 6.1. Single-Agent Coverage

### 6.1.1. Motivating example - Double Integrator

In this first example, we control an agent to explore an occluded environment in order to achieve a uniform probability of detection across a square terrain, using Algorithm 4. The shaded regions (occlusions) $\mathcal{O}$ comprise a circle and a rectangle in Fig. 6.1 and they exhibit zero probability of detection i.e. $\Phi(x) = 0 \forall x \in \mathcal{O}$. Such situations can arise in vision-based UAV exploration with occlusions corresponding to shaded areas that limit visibility, or in surveillance by mobile sensors where the shaded regions can be thought of as areas where no sensor measurements can be made due to foliage. It is assumed that the agent has second-order dynamics with $n = 4$ states $x = [x_1, x_2, x_3, x_4]$, $m = 2$ inputs $u = [u_1, u_2]$, and $f(x, u) = [x_2, u_1, x_4, u_2]$ in (5.6). Forcing saturation levels are set as $u_{min} = -50$ and $u_{max} = 50$.

Snapshots of the agent exploration trajectory is shown in Fig. 6.1. As time progresses from $t = 0$ to $t = 60$ the spacing between the trajectory lines is decreasing, meaning that the agent successfully and completely covers the square terrain by the end of the simulation. This is also reflected in the spatial statistics of the performed trajectory that eventually closely match the desired probability of detection. A similar example was used by Mathew et al. in [**138**] for evaluation of their ergodic control method that was specific to double integrator systems. Our results serve as proof of concept, showing that ergodic iSAC—although designed to control complex nonlinear systems—can still handle simple systems efficiently and achieve full area coverage, as expected.

### 6.1.2. Unmanned Ground Vehicle

Here, we control an agent with simple Unmanned Ground Vehicle (UGV) dynamics to search a terrain with respect to a bimodal Gaussian probability-of-detection distribution. The kinematic UGV model has 3 states ($n = 3$ in system (5.6)), consisting of the position and orientation $[x_c, y_c, \theta_c]$ of the vehicle. Control inputs are the translational and rotational velocities, $u = [v, \omega]$. It is $f(x, u) = [v \cdot cos(\theta_c), v \cdot sin(\theta_c), \omega]$ in (5.6). The resulting ergodic trajectory is illustrated in Fig. 6.2.

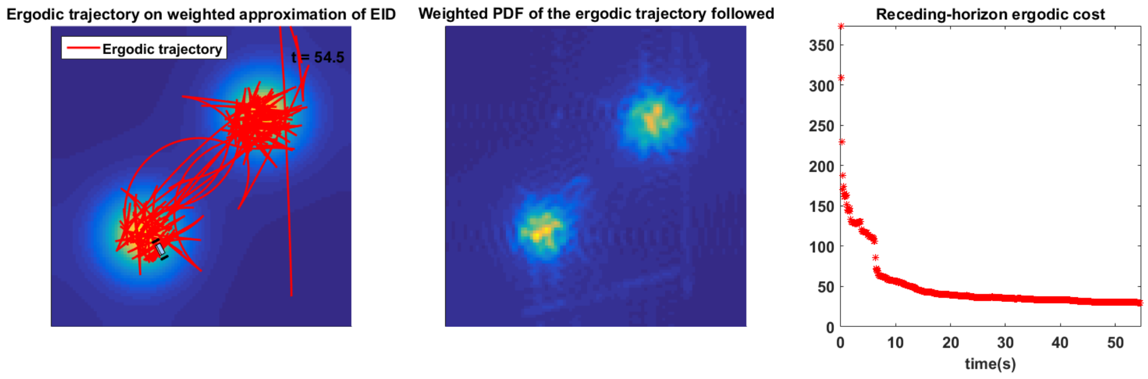


Figure 6.2. Bimodal Gaussian distribution coverage using a UGV. On the left, the exploration trajectory is shown in red on top of the bimodal distribution that indicates probability of detection. Middle figure shows the spatial statistics of the ergodic trajectory, calculated as in Fig. 6.1 with $K = 20$. The objective of ergodic exploration is for the trajectory spatial statistics (center) to match the distribution $\Phi(x)$ (left). On the right, the open-loop ergodic control cost is shown to reduce in time.

### 6.1.3. Unmanned Aerial Vehicle

We will utilize a 12-dimensional quadrotor model to demonstrate the algorithm's efficiency in planning trajectories for agents governed by higher-dimensional nonlinear dynamics. The quadrotor model [**150–152**] has 12 states ($n = 12$ in system (5.6)), consisting of the position $[x_q, y_q, z_q]$ and velocity $[\dot{x}_q, \dot{y}_q, \dot{z}_q]$ of its center of mass in the inertial frame, and the roll, pitch, yaw angles $[\phi_q, \theta_q, \psi_q]$ and corresponding angular velocities $[\dot{\phi}_q, \dot{\theta}_q, \dot{\psi}_q]$ in the body frame. Each of the 4 motors produces the force $u_i$, $i = 1, ..., 4$ ($m = 4$ in (5.6)), which is proportional to the square of the angular speed, that is, $u_i = k\omega^2$. Saturation levels are set as $u_{min} = 0$ and $u_{max} = 12$ in (5.6). Nominal control in Algorithm 4 is a PD (proportional-derivative) controller that regulates the agent's height to maintain a constant value.

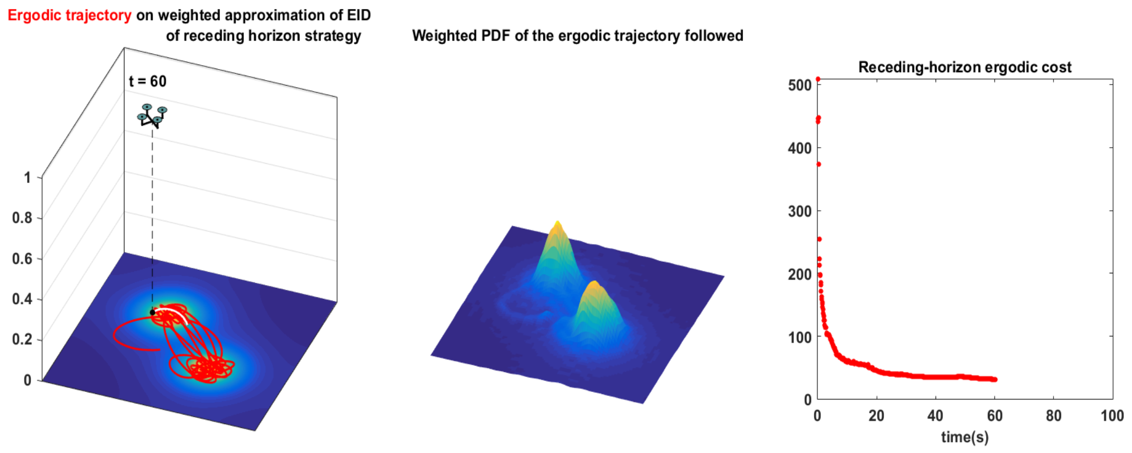


Figure 6.3. Bimodal Gaussian distribution exploration by a quadrotor. The trajectory spatial statistics (center) match closely the distribution under exploration (left) as desired. The controllers adapts to changes in dynamics without any modification required (compared to Fig. 6.2).

Fig. 6.3 shows the resulting ergodic trajectory of a quadrotor (UAV) that is controlled to search a terrain with respect to a bimodal Gaussian distribution. Notice how the controller (Algorithm 4) naturally adapts to the change in dynamics, compared to the UGV implementation shown in Fig. 6.2. Fig. 6.4 shows the result of a quadrotor exploring a Gaussian distribution that varies in time, using Algorithm 5. The algorithm adapts to changes in the distribution in real time. This result verifies that the algorithm is efficient in tracking rapidly changing distributions making it an ideal candidate for information-driven exploration for symbols, as we will see in Part 3.

### 6.1.4. Robotic Fish-like Vehicle

This section applies Algorithm 4 to a robotic fish-like vehicle (Fig. 6.5a) developed by the Smart Microsystems Lab at Michigan State University [**109**]. The robotic fish model has 6 states ($n = 6$ in system (5.6)), consisting of the planar position and orientation $[X, Y, \psi]$, and

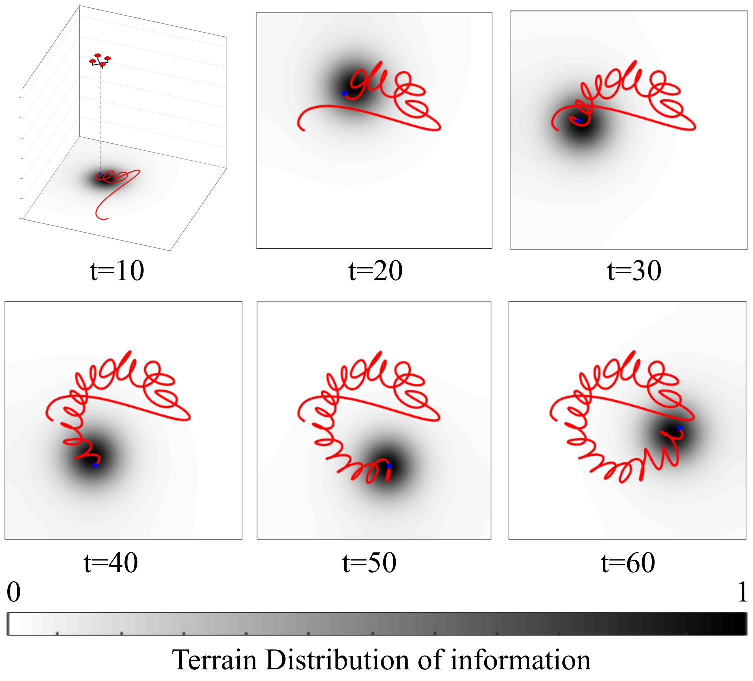


Figure 6.4. Time-varying distribution exploration by a quadrotor, using Algorithm 5.

the translational and rotational velocity of the center of mass $[V_x, V_y, \omega]$, i.e. surge, sway and yaw. The system has two control inputs ($m = 2$) that represent functions of the tail-beat pattern parameters, i.e. bias, amplitude, and frequency of the tail beat, respectively. See [**22**] for a detailed derivation of the robot dynamical equations.

Fig. 6.5 illustrates the resulting ergodic trajectories for searching a Gaussian distribution in two different settings. Fig. 6.5b shows the ergodic trajectory executed by the robotic fish without any constraints on control frequency. However, in the original system, implemented in the lab, the rate of change of tail beat patterns can't be higher than $1Hz$. Fig. 6.5c illustrates the resulting trajectory that satisfies this constraint. This example highlights how Algorithm 4 naturally incorporates the system dynamics and constraints in trajectory generation, so that the resulting trajectories are always ergodic with respect to the distribution. This result is important

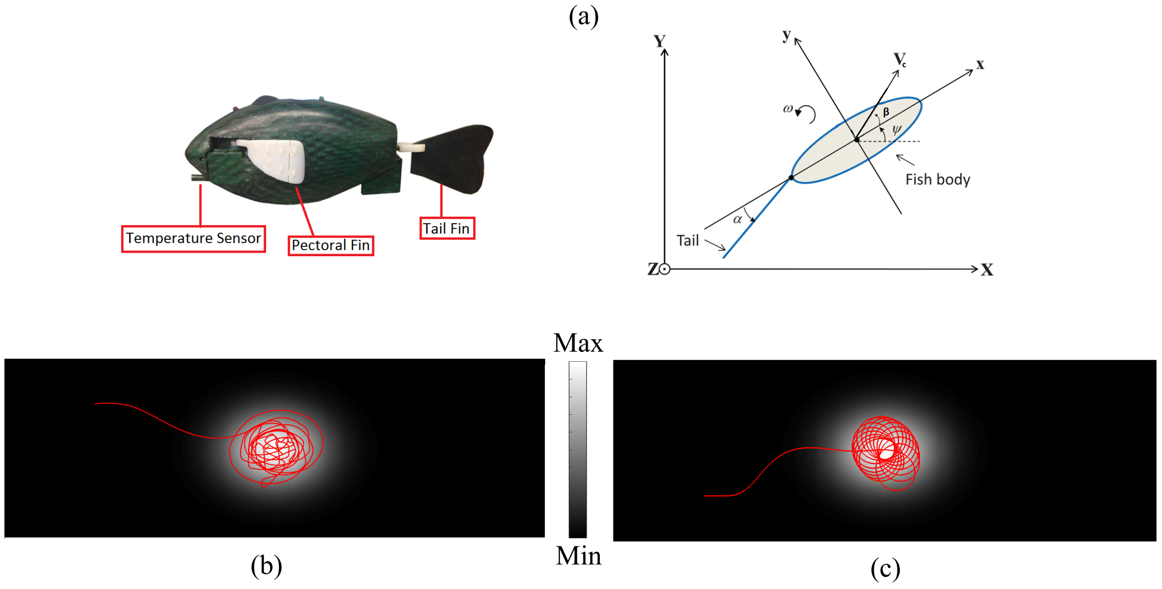


Figure 6.5. (a) The robotic fish developed by the Smart Microsystems Lab at Michigan State University, and its planar configuration. (b) Exploring a Gaussian distribution using the robotic fish and Algorithm 4 without additional constraints. (c) Exploring a Gaussian distribution with added control frequency and tail pattern constraints, as imposed by the experimental setup.

as the first indication that the algorithm is successful under realistic hardware limitations, such as the control frequency constraint.

## 6.2. Multi-Agent Coverage

As pointed out previously, multi-agent exploration is semi-distributed in that each agent executes RHEE (Algorithm 4) independently (using their own computational unit) but shares information with the other agents prior to algorithm execution as a result of relationship (5.28). Here, we use the 12-dimensional quadrotor model as in Section 6.1.3, to show examples of multi-agent ergodic coverage. Fig. 6.6 presents two quadrotors searching a trimodal Gaussian distribution. In Fig. 6.7, we use five aerial vehicles to collectively explore a terrain based on a arbitrary distribution of information. In both examples, we notice that agents avoid exploring

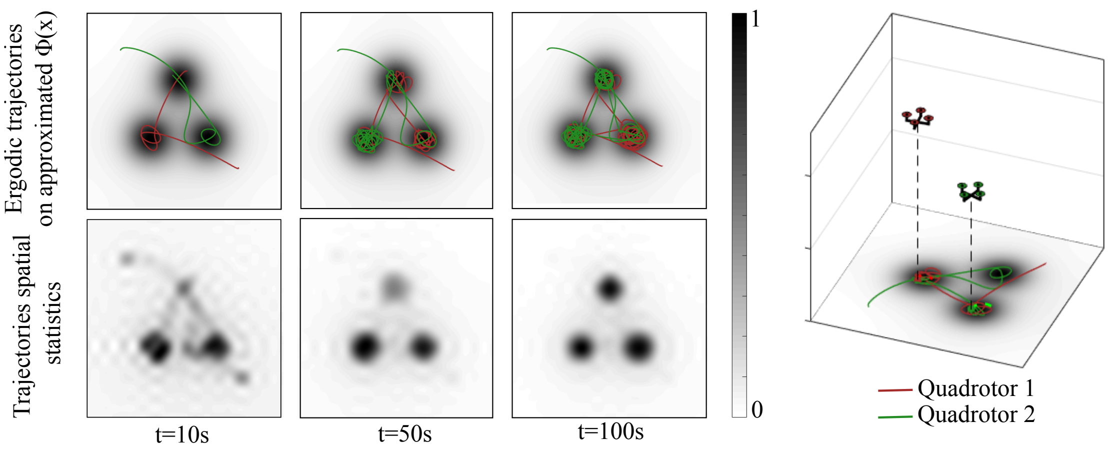


Figure 6.6. Two quadrotors exploring a trimodal Gaussian distribution. The resulting spatial statistics closely match the search distribution, as desired.

the same regions simultaneously and only return back to areas that the other agents has explored after some time has passed. In addition, the spatial statistics of the collective agents exploration closely match the initial distribution of information. It is important to notice here that each agent is not separately assigned to explore a single distribution peak (as a heuristic approach would entail) but rather all agents are provided with the same spatial distribution as a whole and their motion is planned simultaneously in real time to achieve best exploration on the areas with highest probability of detection.

This simulation example was coded in C++ and executed at a Linux-based laptop with an Intel Core i7 chipset. Assuming that each quadrotor executes Algorithm 4 in parallel, the execution time of the $120s$ simulation is approximately $\sim 70s$l, running about two times faster than real time.

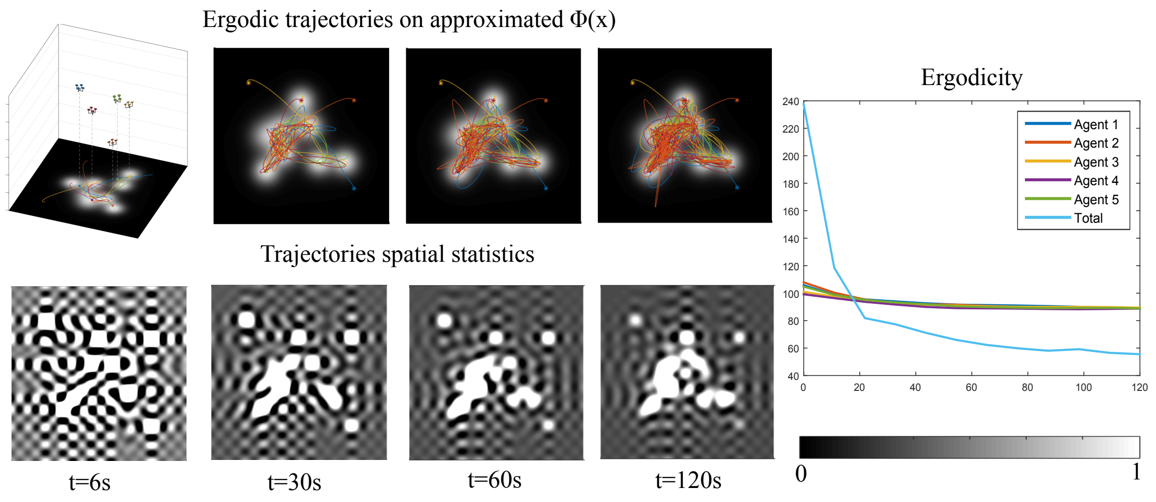


Figure 6.7. Multi-agent UAV exploration (each agent executes Algorithm 4 with trajectory coefficients calculated as in (5.28) with $N = 2$, time horizon $T = 1.3s$ and sampling time $t_s = 0.1s$). Five quadrotor models collectively explore a terrain to track a spatial distribution $\Phi(x)$ (top row). Highest order of coefficients is $K = 12$. Note that agents naturally avoid exploring the same region simultaneously and only return back to already visited areas when sufficient time has passed. Bottom row shows the spatial statistics of the combined agent trajectories calculated as described in Fig. 6.1 caption. As expected, by the end of simulation, the collective spatial statistics match closely the initial spatial distribution. Plot on the right shows the ergodicity measure of trajectories as they evolve in time. Ergodicity of each agent's trajectory at time $t$ is calculated as $\sum_k \{\Lambda_k [\frac{1}{t-t_0^{erg}} \int_{t_0^{erg}}^{t} F_k(x_\zeta(s))ds - \phi_k]^2\}$ for the ergodic trajectories $x_\zeta(t)$ with $\zeta = 1, ..., N$. Total ergodicity of the collective trajectories is calculated as $\sum_k \{\Lambda_k [\frac{1}{t-t_0^{erg}} \int_{t_0^{erg}}^{t} \sum_{j=1}^{N} F_k(x_j(s))ds - \phi_k]^2\}$.

# Part 3

# Sensing Symbols in Real Time

CHAPTER 7

# Introduction: Abstract Sensing of Action Symbols

Chapter 3 in Part 1 proposed a method for generating finite state machines with action symbols. In particular, we employed cellular decomposition techniques to subdivide the state space to $n$-orthotopes (i.e. $n$-dimensional rectangles). To further reduce computational cost, instead of using a multi-resolution grid, could we use informed samples of the state space by performing abstract distributed exploration of the $n$-dimensional state space? Chapter 4.1 introduced this concept as motivation towards building a reliable exploration strategy.

Abstract exploration for sensing action symbols is about controlling abstract agents to perform search on the $n$-dimensional state space while taking measurements, i.e. computing control actions, at a certain frequency. This idea of exploring while taking measurements should be familiar to the reader as a part of target localization maneuvering and sensing for UAVs and other mobile robotic platforms. Where in target localization the terrain is usually the 2D ground area where we expect the target to lie in, here the terrain is the bounded multi-dimensional state space. Accordingly, where measurements are sensor inputs, here measurements correspond to calculated control values for cost improvement (as though we had a sensor that could measure a control value $u$ when being at state $x$). Although no actual exploration and measurement acquisition is performed in this process, we will use these terms in order to connect an abstract concept to the concrete application of space exploration and target localization.

We provide a preliminary evaluation of this approach using the example of cart-pendulum inversion. The cart-pendulum system, shown in Fig. 3.3(a), is assumed to have two states,

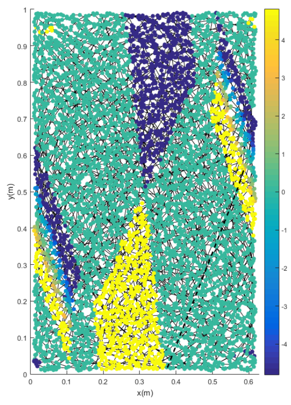


Figure 7.1. Control map for cart-pendulum inversion using infinite number of symbols. X-axis is the angle $\theta$ with range $[-\pi, \pi]$ and y-axis is the angle velocity with range [-6,6]. Colors from yellow to blue correspond to scaled control actions, i.e. cart acceleration values. The map is the result of abstract ergodic exploration using Algorithm 4.

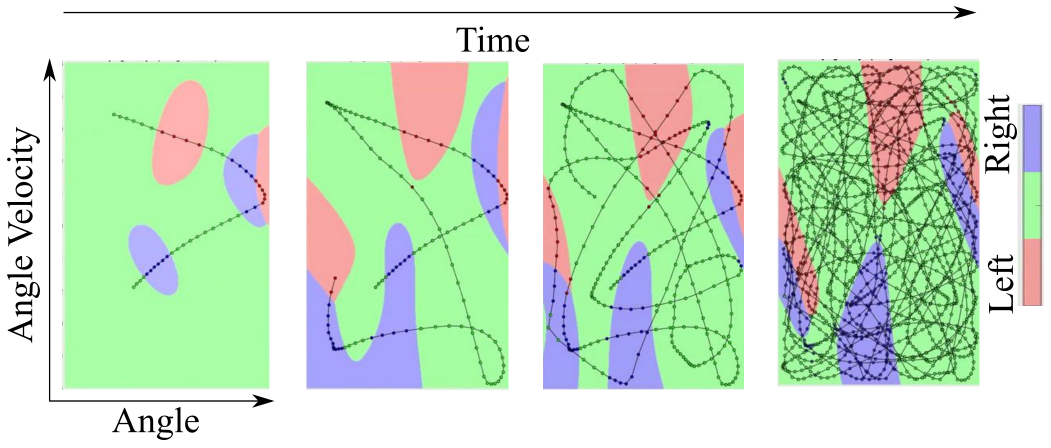


Figure 7.2. Building a control map for cart-pendulum inversion using ergodic exploration (Algorithm 4) and probabilistic classification methods for shape estimation. The exploration trajectory is ergodic with respect to a uniform distribution.

pendulum angle $\theta$ and angle velocity $\dot{\theta}$, with dynamic equations given in 3.10. The objective is to build a control map that maps 2-dimensional states to cart acceleration controls, so that angle $\theta$ is driven to zero and the pendulum is inverted. Fig. 7.1 shows a control map that achieves cart-pendulum inversion using an infinite set of symbols. The coloring is a result of an abstract

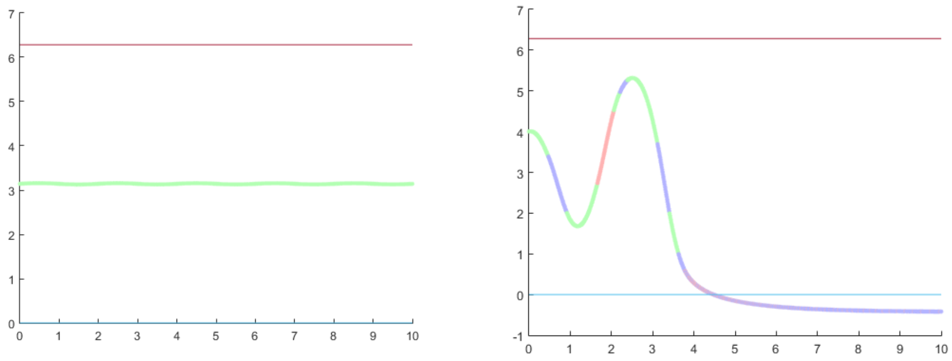


Figure 7.3. Two resulting $\theta$ trajectories of the cart-pendulum when the generated control map of Fig. 7.2 is used for online control. X-axis is “time” in seconds and Y-axis is angle $\theta$ in *rad*. On the left, the cart-pendulum is initialized at the downward position ($[\theta, \dot{\theta}] = [\pi, 0.05]]$) and the control policy fails to invert the system. On the right, the cart pendulum is initialized from a random position and the control policy is successful.)

exploration trajectory on the 2-dimensional state space, generated using Algorithm 4 with $\Phi(x)$ defined as a uniform distribution and double integrator dynamics. We can see that if we allow the agent to explore long enough, the map will look exactly as though we had set a fine grid on the state space and calculated the controls on each grid point. This observation is encouraging in the sense that abstract exploration works, but disappointing when it comes to efficiency: so what is the advantage of exploration versus cell decomposition if we end up acquiring the same number of measurements in both cases?

To answer this question, notice that the coloring of the map in Fig. 7.1 indicates that symbols are in fact more than numbers; they are abstract shapes on the $n$-dimensional state space encompassing information about a state-dependent control policy. Therefore, we can afford to modify our objective slightly: instead of computing a mapping from states to symbols, we now aim to estimate the *shape* of symbols on the state space. This brings forth the need for an additional step in the exploration process that would allow us to build maps with much fewer sensing

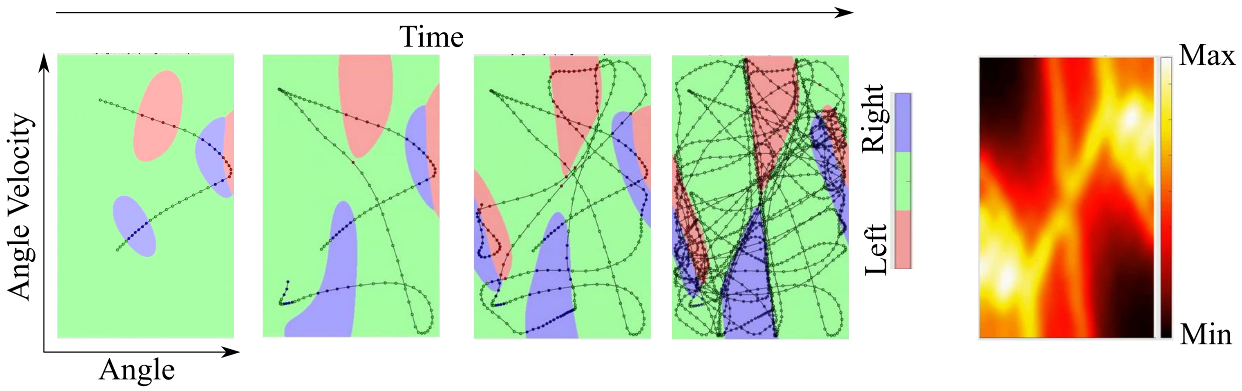


Figure 7.4. Building a control map for cart-pendulum inversion using ergodic exploration (Algorithm 4) and probabilistic classification methods for shape estimation. The exploration trajectory is ergodic with respect to the likelihood of crossing a symbol boundary. This likelihood distribution is shown on the right. Notice how information-driven exploration in this example, compared to uniform exploration in Fig. 7.2, spends more time in areas where more information can be extracted (i.e., the boundaries of symbol shapes).

points. The result of this approach is illustrated in Fig. 7.2, where three symbols are now used. Here, we continue exploring the state space using Algorithm 4 on a uniform distribution but now we use probabilistic classification methods to estimate the shape boundaries based on the accumulated measurements (see Section 9.1.1 for further description of this methodology, now used for estimation of physical object shapes). Fig. 7.3 shows the resulting angle trajectories from two different initial position. In the first case, we initialize the cart-pendulum at the downward stable equilibrium ($[\theta, \dot{\theta}] = [\pi, 0.05]]$) . As we can see, the control policy fails to invert the pendulum. However, choosing a random initial position in the second case, results in successful cart-pendulum inversion with a small offset. This outcome indicates that uniform exploration needs more time to more accurately refine the symbol boundaries on the state space.

To improve this result, instead of performing a uniform exploration, we can actively vary the spatial distribution of exploration during the process of acquiring information about action symbols. The results of this implementation are shown in Fig. 7.4. Here, we design the controller to be ergodic with respect to the likelihood distribution of crossing a “shape” boundary. This allows us to refine the shape estimate in significantly less time than before. In particular, Fig. 7.5 shows how the control policy is successful even when the system is initialized at the downward position. We further evaluate the control policies resulting from uniform and informed exploration, by performing a Monte Carlo test at two different instances during state-space exploration: once at the start of exploration, when only 200 measurements have been acquired and once after 2000 measurements have been acquired. Specifically, we run 1000 trials for each policy, simulating the cart-pendulum system from random initial states with angles in the range $[1.5, 2\pi - 1.5]$ and angle velocities in the range $[-2, 2]$. The results are illustrated in Fig. 7.6. As we can see, informed exploration leads to 96.8% success with only 200 measurements acquired when uniform exploration only results in 53.6% success. After 2000 measurements, the informed exploration policy leads to 100% succesful trials. When using the uniform exploration policy, on the other hand, we can see that trials starting from around the downward equilibrium fail to converge to the upright equilibrium, resulting in 97.5% success rate. Notice that the maps in both Fig. 7.2 and Fig. 7.4 are equivalent to the map generated using cell subdivision in Chapter 3 (Fig.3.5). This indicates that abstract exploration in combination with machine learning estimation techniques can be a valid and promising solution for building control alphabet policies.

Next, we explore the possibility of the system itself exploring its state space in order to build a task-specific policy. In particular, instead of abstractly exploring the state space using double integrator dynamics, we generate dynamically feasible exploration trajectories, i.e. trajectories that are constrained by the cart-pendulum system dynamics and thus the card pendulum can

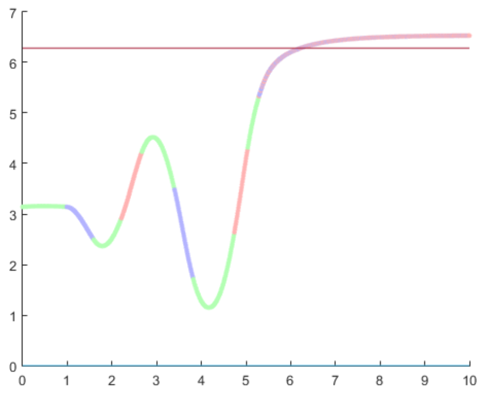


Figure 7.5. The resulting $\theta$ trajectory of the cart-pendulum when the generated control map of Fig. 7.4 is used for online control. X-axis is "time" in seconds and Y-axis is angle $\theta$ in *rad*. The cart-pendulum is initialized at the downward position ($[\theta, \dot{\theta}] = [\pi, 0]$] and the control policy is successful in inverting the system with a small offset.) Comparing this result with Fig. 7.3 indicates that exploration with respect to a dynamically varying expected information density results in better refinement of the symbol boundaries on the state space.

physically perform to self-explore its space space in practice. Although the concept is similar to our previous methodologies, incorporating more complex dynamics in ergodic control complicates the exploration strategy and might as well render exploration of certain state space regions infeasible. An example that highlights the challenges of this approach is illustrated in Fig. 7.7. One can notice that the exploration trajectory covers the state space by following the system's vector field, orbiting around the stable equilibrium. However, actuation limits as well as instability around the upward equilibrium don't allow the system to adequately explore the regions around the desired inverted state, resulting in an incomplete map.

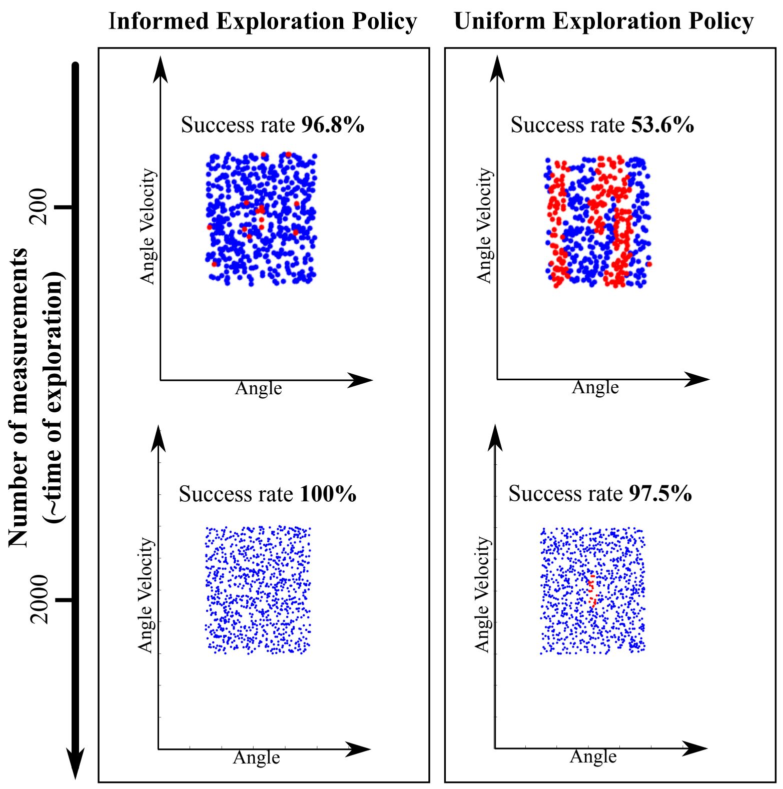


Figure 7.6. Monte Carlo tests for evaluating the control policies resulting from uniform and informed state-space exploration at two different instances during exploration: at the start of exploration when only 200 measurements have been acquired and after 2000 measurements have been acquired. Each dot on the state-space maps corresponds to a single trial (1000 trials in total per map). Blue dots indicate successful trials, while red indicates failed trials.

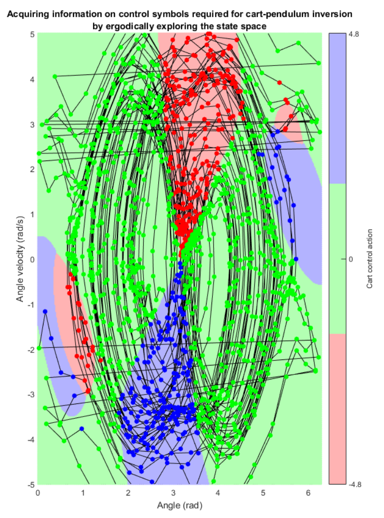


Figure 7.7. Control map for cart-pendulum inversion using three symbols and dynamically feasible exploration trajectories. X-axis is the angle $\theta$ with range $[-\pi, \pi]$ and y-axis is the angle velocity with range [-5,5]. Colors correspond to symbolic control actions, i.e. cart acceleration values. The map is the result of abstract ergodic exploration using Algorithm 4 and cart-pendulum dynamics (instead of double integrator dynamics in Fig. 7.2). Notice that the exploration trajectory fails to accurately cover the state-space region around the unstable equilibrium ($\theta = 0$ and $\theta = 2\pi$), resulting in an incomplete map around these regions (compare with Fig. 7.2). This is due to the exploration dynamics that don't allow the system to "linger" around the unstable equilibrium to accurately get information about the state-space region around it without violating actuation and state constraints.

CHAPTER 8

# Tracking Symbols: Bearing-Only Localization of Targets

As we will see later in Section 8.1, targets are uniquely associated with an information signature that is extracted through their measurement model. According to the symbol definition given in Section 1.2, this statement by itself is enough to identify any kind of target as symbol. A target then is a symbol because it encompasses information about the way it is sensed. This chapter presents how we can search for and track symbols with known measurement models using information-driven exploration as introduced in Chapter 5. After deriving a methodology for exploration that is efficient regardless of the information level, the next most crucial step is generating the information signature based on a known measurement model and the current estimate on symbol state.

Ergodic control for localization of static or moving targets is essentially an application of reactive RHEE in Algorithm 5 with the following specifics: 1) the agent takes sensor measurements every $t_m$ seconds while exploring the distribution in Step 4, and 2) belief of targets' state is updated online and used for computation of $\Phi(x)$ in Step 1. The frequency of measurements $f_m = \frac{1}{t_m}$ can be higher or equal to the frequency of $\Phi(x)$ update $f_\phi = \frac{1}{t_\phi}$, i.e. $f_m \geq f_\phi$. When the target is static and frequency of measurements is relatively high (depending on the sensor capacity), it is preferred that $f_m \gg f_\phi$ so that the agent has sufficient time to explore the current EID and get useful measurements with high information. For moving targets, however, it is desirable that $f_m \approx f_\phi$ to ensure that the agent always explores the area taking into account the best estimate of the target position.

### 8.0.1. Existing Strategies for Target Tracking

Target localization/tracking refers to the process of acquiring and using sensor measurements to estimate the state of a single or multiple targets. One of the main challenges involved with target localization is developing a sensor trajectory such that high information measurements are acquired. To achieve this, some methods perform information maximization (IM) usually by compressing an information metric (such as Fisher Information Matrix (FIM) [**111**] and Bayesian utility functions [**9**]) to a scalar cost function (e.g. using the determinant [**11**], trace [**10**], or eigenvalue [**153**] of the information matrix) and then generating trajectories that optimize this cost. The most general approaches to solving these problems involve numerical techniques [**154**], classical optimal control theory [**97**], and dynamic programming [**122, 155**], which tend to be either computationally intensive or application specific (e.g. consider only static and/or constant velocity targets). As opposed to IM approaches that control the agent to move to locations of maximum information, we propose ergodically exploring an expected information density map as in [**101**]. Thus, the sensor will spend more time exploring regions of space with higher expected information, where useful measurements are most likely to be found. We will see that solving the target localization problem using ergodic exploration, allows tracking multiple moving targets without added complexity as opposed to IM techniques that would need to plan a motion for each target separately.

Compared to common IM techniques [**8–11, 97–99, 156**], ergodic exploration is more robust to the existence of local maxima in the expected information density. While IM methods can get stuck in a local maximum if not initialized properly, ergodic control is bound to explore all density maxima. A common workaround to the negative effects of information maximization algorithms is planning paths for agents with constant speed—this ensures that the agent

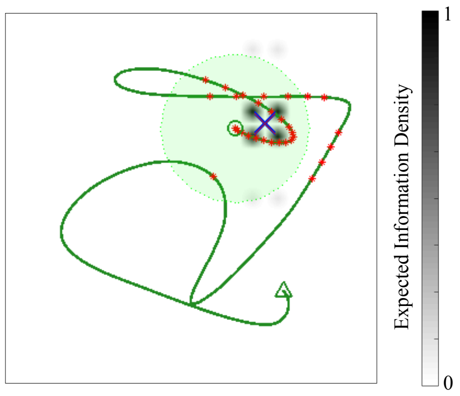


Figure 8.1. Top-view illustration of the ergodic trajectory (green curve) performed by an aerial vehicle while localizing a target (blue marker) in a square terrain. Light green circle shows the camera sensor range around the current UAV position. Red dots correspond to bearing-only measurements of the target. One can observe the benefits of trajectory optimization—pursued by our approach—as opposed to path planning [**8–11**]. Although temporal frequency of measurements is constant at $10Hz$, the spacing between consecutive measurements becomes smaller when the UAV approaches the target. This means that the agent is controlled to slow down when higher information areas are encountered and to speed up otherwise.

will not just reach the state with maximum information and remain there, but instead continue moving [**8–11, 100**]. As a result, the agent's velocity magnitude is arbitrarily selected and not optimized, i.e. only path—and not trajectory—planning is performed. Additionally, the constant speed assumption entails that the temporal frequency of sensor measurements must be controlled/optimized separately. The approach presented in this chapter controls velocity proportionally to the amount of expected information, thus allowing the agent to slow down when higher information areas are encountered (reducing the "spatial" rate of measurements while the temporal frequency remains constant) and speeding up otherwise (see Fig. 8.1).

Another challenging aspect of target localization is maintaining reliable performance in sensor occlusion situations. These can be cases when the sensor has limited sensing range or when physical entities block sensing capacity e.g. when high buildings and/or hills obscure the target during vision-based localization by air [**157**]. In such cases, IM can converge to an area where no target can be sensed, especially if there is high uncertainty over the target's true position. However, ergodically exploring the information density instead of tracking its maximum, allows the sensor to eventually work around the occlusions even without knowledge of their existence, and without the need for specialized occlusion avoidance techniques.

Because IM techniques tend to exhibit prohibitive execution times for moving targets, alternative methods of diverse nature have been proposed for use in real-world applications. A non-continuous grid decomposition strategy for planning parameterized paths for UAVs is proposed in [**157**] with the objective to localize a single target by maximizing the probability of detection when the target motion is modeled as a Markov process. Standoff tracking techniques are commonly used to control the agent to achieve a desired standoff configuration from the target usually by orbitting around it [**12**, **158**, **159**]. A probabilistic planning approach for localizing a group of targets using vision sensors is detailed in [**160**]. In [**161**], a UAV is used to track a target in constant wind considering control input constrains, but the planned path is predefined to converge to a desired circular orbit around the target. A rule-based guidance strategy for localizing moving targets is introduced in [**162**]. In [**163**], the problem of real-time path planning for tracking a single target while avoiding obstacles is addressed through a combination of methodologies. In general, the above approaches focus on and are only applicable in special real-world situations and don't generalize directly to random multiple-target tracking situations. For a complete and extensive comparison of ergodic localization to other motion planning approaches, the reader is referred to [**101**].

In this chapter, we use as an example the problem of bearing-only localization. Many real-world systems use angle-only sensors for target localization, such as submarines with passive sonar, mobile robots with directional radio antenna [**164**], and unmanned aerial vehicles (UAVs) using images from an optical sensor [**112**]. Bearing-only systems require some amount of maneuver to measure range with minimum uncertainty [**97**]. Here, we show in simulation how we can ergodically control quadrotor UAVs in real time to autonomously perform vision-based bearing-only target localization using a gimballed camera. The majority of existing solutions for UAV bearings-only target motion planning, produce circular trajectories above the target's estimated position [**110, 165**]. However, this solution is only viable if there is low uncertainty over the target's position. In addition, if the target is moving, the operator may not know what the best vehicle path is.

*Cooperative Target Localization:* The greatest body of work in the area of cooperative target localization is comprised by standoff tracking techniques. Vector fields [**12**], nonlinear feedback [**166**] and nonlinear model predictive control [**158**] are some of the control methodologies that have been used for achieving the desired standoff configuration for a target. The motion of the robotic agents is coordinated in a geometrical manner: two robotic agents orbit the target at a nominal standoff distance and maintain a specified angular separation; when more agents are considered, they are controlled to achieve a uniform angular separation on a circle around the target. The main concern in using this strategy is scalability to multiple targets, as the robots are separately controlled to fly a circular orbit around each single target [**167**]. A dynamic programming technique that minimizes geolocation error covariance is proposed in [**168**]. However, the solution is not generalizable to multiple robotic agents and targets. The robots are controlled to seek informative observations by moving along the gradient of mutual information in [**156**].

## 8.1. Expected Information Density

Exploration using Algorithm 4 is information-driven in that it controls agent to be ergodic with respect to a distribution that represents expected information density. Here, we focus on the part of calculating this distribution $\Phi(x)$ (for now on referred to as Expected Information Density, EID) given the current targets belief and a known measurement model. It is important to point out that the following process for computing the EID depends only on the measurement model; the methodology for belief state representation and update can be arbitrary (e.g. Bayesian methods, Kalman filter, particle filter etc.) and does not alter the ergodic target localization process. The objective is to estimate the unknown parameters $\boldsymbol{\alpha}_k \in \mathbb{R}^M$ describing the $M$ coordinates of a target at time step $k$. A measurement $\mathbf{z} \in \mathbb{R}^\mu$ is made according to a known measurement model $\mathbf{z} = \Upsilon(\boldsymbol{\alpha}, \mathbf{x}) + \delta$, where $\Upsilon(\cdot)$ is a function of sensor configuration and parameters, and $\delta$ represents zero mean Gaussian noise with covariance $\Sigma$, i.e. $\delta \sim \mathcal{N}(0, \Sigma)$.

As in [**101**], we will use the Fisher Information Matrix (FIM) [**169, 170**] to calculate the EID. Often used in maximum likelihood estimation, Fisher information $\mathcal{I}(x, \boldsymbol{\alpha})$ is the amount of information a measurement provides at location $x$ for a given estimate of $\boldsymbol{\alpha}$. It quantifies the ability of a set of random variables, in our case measurements, to estimate the unknown parameters. For estimation of parameters $\boldsymbol{\alpha} \in \mathbb{R}^M$, the Fisher information is represented as a $M \times M$ matrix. Assuming Gaussian noise, the $(i, j)$th FIM element is calculated as

$$\mathcal{I}_{i,j}(x, \boldsymbol{\alpha}) = \frac{\partial \Upsilon(\boldsymbol{\alpha}, x)}{\partial \alpha_i}^T \Sigma^{-1} \frac{\partial \Upsilon(\boldsymbol{\alpha}, x)}{\partial \alpha_j} \tag{8.1}$$

where $\Upsilon(\boldsymbol{\alpha}, \mathbf{x}) : \mathbb{R}^M \times \mathbb{R}^n \to \mathbb{R}^\mu$ is the measurement model with Gaussian noise of covariance $\Sigma \in \mathbb{R}^\mu$. Since the estimate of the target position $\boldsymbol{\alpha}$ is represented as a probability distribution function $p(\boldsymbol{\alpha})$, we take the expected value of each element of $\mathcal{I}(x, \boldsymbol{\alpha})$ with respect to the joint distribution $p(\boldsymbol{\alpha})$ to calculate the expected information matrix, $\boldsymbol{\Phi}_{i,j}(x)$. The $(i, j)$th element of $\boldsymbol{\Phi}_{i,j}(x)$ is then

$$\boldsymbol{\Phi}_{i,j}(x) = \int_{\boldsymbol{\alpha}} \mathcal{I}_{i,j}(x, \boldsymbol{\alpha}) p(\boldsymbol{\alpha})\, d\boldsymbol{\alpha}. \tag{8.2}$$

To reduce computational cost, this double integration is performed numerically by discretization of the estimated parameters on a grid and a double nested summation. Note that target belief $p(\boldsymbol{\alpha})$ might incorporate estimates of multiple targets depending on the application. For that reason, this EID derivation process is independent of the number of targets and method of targets belief update.

In order to build a density map using the information matrix (8.2), we need to compress it to a scalar function so that each state $x$ is assigned a single information value. We will use the following mapping:

$$\Phi(x) = \det \boldsymbol{\Phi}(x). \tag{8.3}$$

The FIM determinant (D-optimality) is widely used in the literature, as it is invariant under re-parameterization and linear transformation [**171**]. A drawback of D-optimality is that it might result in local minima and maxima in the objective function, which makes optimization difficult when maximizing information. In our case though, local maxima do not pose an issue as our objective is to approximate the expected information density instead of maximizing it.

### 8.1.1. Remarks on Localization with Limited Sensor Range

The efficiency of planning sensor trajectories by maximizing information metrics like the Fisher Information Matrix in (8.1) is highly dependent on the true target location [**171**]: if the true target location is known, the optimized trajectories are guaranteed to acquire the most useful measurements for estimation; if not, the estimation and optimization problems must be solved simultaneously and there is no guarantee that useful measurements will be acquired especially when the sensor exhibits limited range.

A limited sensor range serves as an occlusion during localization, in that large regions are naturally occluded while taking measurements. Because of this, how we plan the motion of the agent according to the current target estimate is critical; if, at one point, the current target belief largely deviates from the true target position, the sensor might completely miss the actual target (out of range), never acquiring new measurements in order to update the target's estimate. This would be a possible outcome if we controlled the agent to move towards maximum information (IM). In this section, we explain how receding-horizon ergodic target localization (Algorithm 5) with limited sensor range can overcome this drawback under a single assumption.

**Assumption 5.** *Let $r \in \mathbb{R}^+$ be the radius defining sensor range so that a sensor positioned at $\boldsymbol{x}_s \in \mathcal{X}_v$ can only take measurements of targets whose true target location $\boldsymbol{\alpha}_{true} \in \mathbb{R}^v$ satisfies $||\boldsymbol{x}_s - \boldsymbol{\alpha}_{true}||_v < r$. An occlusion $O$ is defined as the region where no sensing occurs i.e., $O = \{x_s \in \mathcal{X}_v : ||\boldsymbol{x}_s - \boldsymbol{\alpha}_{true}||_v > r\}$. At all times $t_{curr} < \infty$ in Algorithm 5, there is $x_q \in \mathcal{X}_v$ that simultaneously satisfies $||\boldsymbol{x}_q - \boldsymbol{\alpha}_{true}||_v < r$ and $\Phi_{curr}(x_q) > 0$, where $\Phi_{curr}(x) \forall x \in \mathcal{X}_v$ is the expected information density computed as in* (8.3) *at time $t_{curr}$.*

**Proposition 7.** *Let Assumption 5 hold. Also, let $x_v(\cdot) : [t_{curr}, \infty) \to \mathcal{X}_v$ denote the exploration trajectory of an agent performing ergodic target localization (Algorithms 4 and 5) with expected information density $\Phi_{curr}(x) \forall x \in \mathcal{X}_v$, equipped with a sensor of range $r \in \mathbb{R}$. Then, there will be time $t_s \in [t_{curr}, \infty)$ where the agent's state satisfies $x_v(t_s) \in \mathcal{X}_v \setminus O$, so that new measurements are acquired and $\Phi_{curr}(x)$ is updated.*

**Proof.** Due to Assumption 5, at time $t_{curr}$ there is $x_q \in \mathcal{X}_v \setminus O$ that satisfies $\Phi_{curr}(x_q) > 0$. According to Theorem 7 and Definition 7, it is $C(x) - \Phi_{curr}(x) \to 0$ for all $x \in \mathcal{X}_v$ as $t \to \infty$. Therefore, at some time $t \in [t_{curr}, \infty)$, we know that $C(x_q) = \Phi_{curr}(x_q) > 0$ that is equivalent to $\frac{1}{t-t_{curr}} \int_{t_{curr}}^{t} \delta[x_q - x_v(\tau)]d\tau > 0$ from Eq. (5.1). This leads to the conclusion that $x_q \in x_v(\cdot)$ which directly proves the proposition.

□

Assumption 4—stating that information density is always non-zero in an arbitrarily small region around the true target—can be satisfied in various ways. For example, we can adjust the parameters of the estimation filter to achieve a sufficiently low convergence rate. Alternatively, in cases of high noise and variability, we can artificially introduce nonzero information values across the terrain so as to promote exploration as in the simulation and experimental examples.

## 8.2. Results

### 8.2.1. Simulation Results

In the following examples, we will use the 12-dimensional nonlinear quadrotor model from Section 6.1.3 to perform motion planning for vision-based static and moving target localization with bearing-only sensing through a gimbaled camera that always faces in the direction of

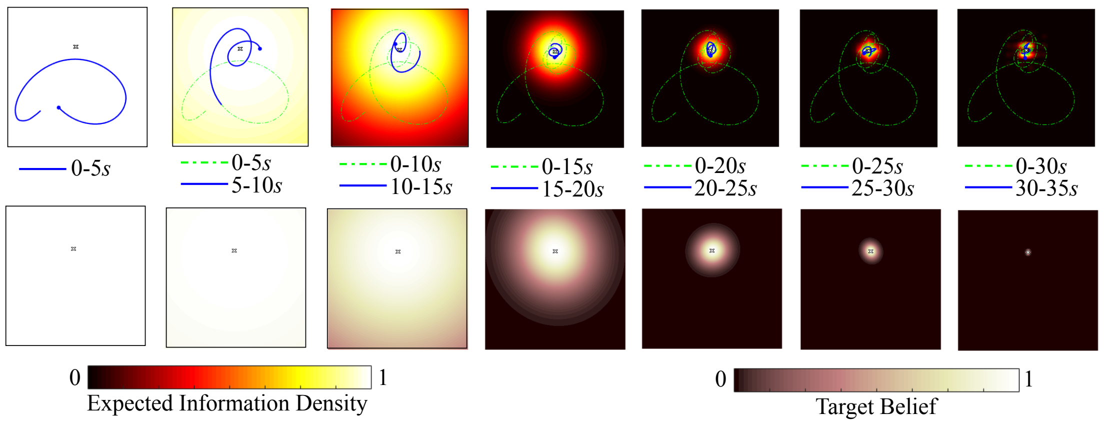


Figure 8.2. Bearing-only static target localization. Top row shows top-view snapshots of the ergodic trajectory followed by the quadrotor in different time windows along with the corresponding EID map. EID is updated every 5 seconds and a new sensor measurement is taken every second. The quadrotor explores the areas with highest information to acquire useful measurements. Although the geometry of the paths is not predefined, the resulting trajectories follow a cyclic, swirling pattern around the true target position, as one would naturally expect. The target belief, illustrated in bottom, converges to a normal spatial distribution with the mean at the true target position and low covariance.

gravity. Representing the estimate of target's position as a Gaussian probability distribution function, we use an Extended Kalman Filter (EKF) [**172**, **173**] to update the targets' position belief based on the sensor measurements. We use EKF because it is fast, easy to implement and regularly used in real-world applications but any other estimator (e.g. [**174**]) can be used instead with no change to the process of Algorithm 5. Note for example that reliable bearings-only estimation with EKF cannot be guaranteed as previous results indicate [**175**], so it might be desirable to use a more specialized estimator.

It is assumed that the camera uses image processing techniques (e.g. [**176**]) to take bearing-only measurements, measuring the azimuth and elevation angles from current UAV position $[x_q, y_q, z_q]$ to the detected target's position $[x_\tau, y_\tau, z_\tau]$. The corresponding measurement model is:

$$\Upsilon([x_\tau, y_\tau, z_\tau], [x_q, y_q, z_q]) = \begin{bmatrix} \tan^{-1}\left(\frac{x_q - x_\tau}{y_q - y_\tau}\right) \\ \tan^{-1}\left(\frac{z_q - z_\tau}{\sqrt{(x_q-x_\tau)^2 + (y_q - y_\tau)^2}}\right) \end{bmatrix}. \tag{8.4}$$

The measurement noise covariance is $\Sigma = diag\{[0.1, 0.1]\}$. As in the previous examples, a PD controller serves as nominal control, regulating UAV height $z_q$. In some of the next examples, we assume that the sensor has limited range of view, depicted as a circle around the target. The target transition model for EKF is expressed as $\boldsymbol{\alpha}_k = F(\boldsymbol{\alpha}_{k-1}) + \epsilon$ with $\epsilon$ representing zero mean Gaussian noise with covariance $C$, i.e. $\epsilon \sim \mathcal{N}(0, C)$. We assume that no prior behavior model of the target motion is available and thus the transition model is $F(\boldsymbol{\alpha}_{k-1}) \equiv \boldsymbol{\alpha}_{k-1}$. Importantly, the camera sensor has limited range of view, completely disregarding targets that are outside of a circle centered at the UAV position with constant radius (as depicted in Fig. 8.3).

**8.2.1.1. Bearing-only localization of a static target.** This first example aims to demonstrate how the process of exploring, taking measurements and updating an EID to track a target works in practice. The results are given in Fig. 8.2. The expected information density is updated every 5 seconds and a new sensor measurement is taken every second. The rate of change of EID is intentionally kept low to show the performance of Algorithm 5 in exploring each distribution.

**8.2.1.2. Bearing-only localization of a moving target with limited sensor range.** Here, we demonstrate an example where a quadrotor is ergodically controlled to localize a moving target, with frequency of measurements $f_m = 20$Hz and frequency of EID update at $f_\phi = 10$Hz. The 3D target position $[x_\tau, y_\tau, z_\tau]$ is localized so that $M = 3$. We assume that no prior behavior model of the target motion is available and thus the transition model is $F(\boldsymbol{\alpha}_{k-1}) \equiv \boldsymbol{\alpha}_{k-1}$ with covariance $C = diag\{[0.001, 0.001, 0.001]\}$ (i.e modeled as a diffusion process). Top-view snapshots of

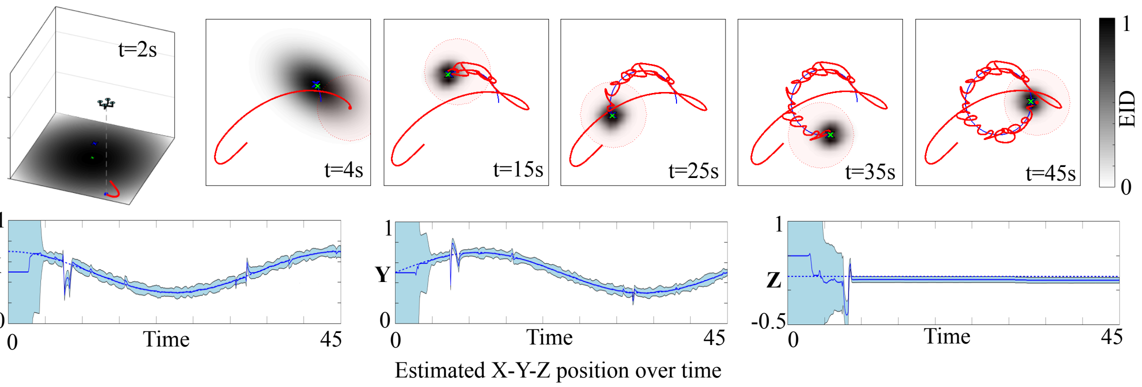


Figure 8.3. Bearing-only localization of a moving target. Top: Top-view snapshots of the UAV trajectory (red curve) where the true target position (blue X-mark) and path (blue curve), and the estimated target position (green X-mark) are also illustrated. The quadrotor can acquire vision-based measurements with a limited range of view that is illustrated as a light red circle around the current UAV position. No prior behavior model of the target motion is available for estimation using EKF. The limited sensor range serves as an occlusion as it naturally occludes large regions while taking measurements. The highest order of coefficients is $K = 10$. The quadrotor explores the areas with highest information to acquire useful measurements. Although the geometry of the paths is not predefined, the resulting trajectories follow a cyclic, swirling pattern around the true target position, as one would naturally expect — like in standoff tracking solutions for example [**12**]. Bottom: The target estimate (solid blue curve) is compared to the real target position (dashed blue curve) along with an illustration of the belief covariance (light blue area around estimated position) over time. The target belief converges to a normal spatial distribution with the mean at the true target position and low covariance.

the UAV motion are shown in Fig. 8.3. The agent detects the target without prior knowledge of its position, whereafter it closely tracks the target by applying Algorithm 5 to adaptively explore a varying expected information density $\Phi(x)$. Although the geometry of the paths is not predefined, the resulting trajectories follow a cyclic, swirling pattern around the true target position, as one would naturally expect.

This simulation example was coded in C++ and executed at a Linux-based laptop with an Intel Core i7 chipset. The execution time of the $45s$ simulation is approximately $\sim 25s$. This result is representative of the algorithm's computational cost and execution time, because it involves a high-dimensional, nonlinear system. The step that might arise concerns with regard to execution time is the calculation of the distribution Fourier coefficients $\phi_k$ using relationship (5.2) (first step in Algorithm 4). However, note that this computation can be reasonably fast if we discretize the terrain on a grid and use a double nested summation. Some parameters that can be tuned to further reduce the execution time are the control application period (or sampling time) $t_s$, the time horizon $T$ and the highest order of coefficients $K$. Localization is slower than pure exploration, mainly because it requires calculation of the expected information density every $t_\phi$ seconds using the expressions (8.1), (8.2) and (8.3). In practice however, calculation of updated EID in target localization can occur in parallel while the sensor vehicle is ergodically exploring a current EID and taking measurements, using of course a previous batch of measurements.

**8.2.1.3. Multi-agent simultaneous terrain exploration and target localization.** This simulation example is designed to demonstrate advanced features of the reactive RHEE algorithm by promoting both search for undetected targets (exploration) and localization of detected moving targets simultaneously, using two agents. This problem of when to explore for new information (search) and when to exploit the current information to get the best desired performance (localization) is widely known as the trade-off between exploration and exploitation [**177**]. Although our algorithm does not explicitly decide when to explore or when to exploit, the agents will naturally explore for new information while exploiting current information due to the fact that the trajectory is ergodic with respect to the given distribution [**178**].

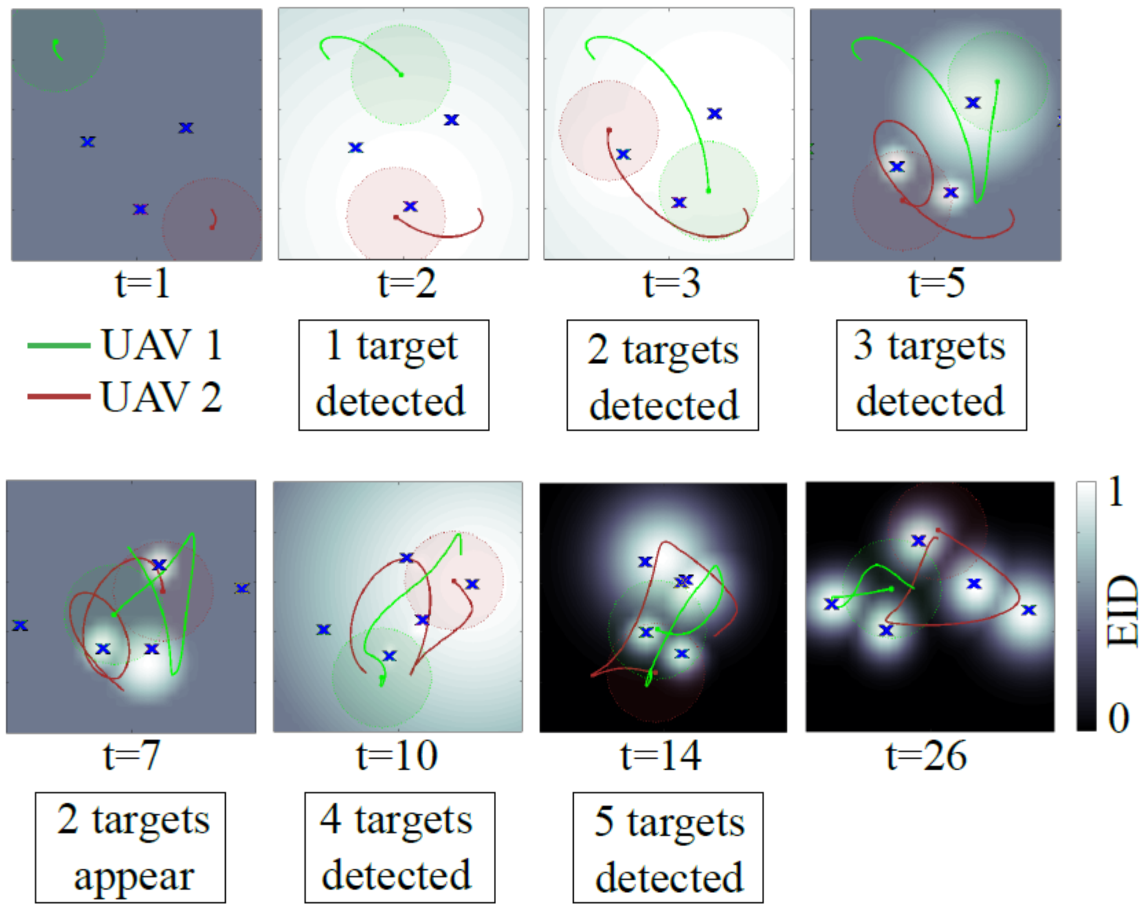


Figure 8.4. Multi-agent simultaneous exploration and targets localization. The problem of exploration vs exploitation is addressed by controlling two agents to localize detected targets while exploring for new undetected targets. The algorithm scales to multiple target localization without any modification, as it tracks a universal non-parametric information distribution instead of each target independently. For cleaner representation, only the UAV trajectories of the past $5s$ are shown in each snapshot. Highest order of coefficients is $K = 10$. Mean and standard deviation of targets belief (not shown here) fluctuate in a pattern similar to the experimental results in Fig. 8.8. Light green and red circles around the current UAV positions indicate the camera range of view. Notice that before all targets are detected, the EID value is set at a middle level (gray color) in areas where no high information measurements can be taken from the already detected targets. This serves to promote exploration for more targets.

A maximum of 5 targets must be localized by 2 UAVs whereafter exploration is discontinued and only localization is performed. Note that here we do not address the issue of cooperative sensing filters [**179**] for multiple sensor platforms: instead, we use a centralized Extended Kalman Filter for simplicity but any filter that provides an estimate of the target's state can be employed instead. Top-view snapshots of the multi-agent exploration trajectories are given in Fig. 8.4. At $t = 0$ when 3 targets are present in the terrain but none of them have been detected by the agents, the EID is a uniform distribution across the workspace. Information density is set at a middle level i.e. $\Phi(x) = 0.5$ for all $x$ (gray coloring). By $t = 5$ all present (three) targets have been detected and the EID map is computed based on Fisher Information using expressions (8.1), (8.2) and (8.3). Note however that information levels are still set at a middle level (instead of zero) in areas where information of target measurements is zero. This serves to promote exploration in addition to localization. In this special case, the terrain spatial distribution encodes both probability of detection (for the undetected targets) and expected information density (for the detected targets). This shows how we can manipulate the agents' performance simply by selecting an appropriate spatial distribution. At $t = 7$ the two last targets appear. By $t = 14$ all 5 targets have been detected whereafter the spatial distribution only encoded expected information density (note that $\Phi(x) = 0$ for all $x$ where information from measurements is zero).

This simulation example was coded in C++ and executed at a Linux-based laptop with an Intel Core i7 chipset. Assuming that each quadrotor executes Algorithm 4 in parallel, the execution time is approximately $\sim 30s$. Note that in simulation Algorithm 4 was executed in series for each agent and we only counted the maximum algorithm execution times among all agents to derive a realistic approximation of the simulation time.

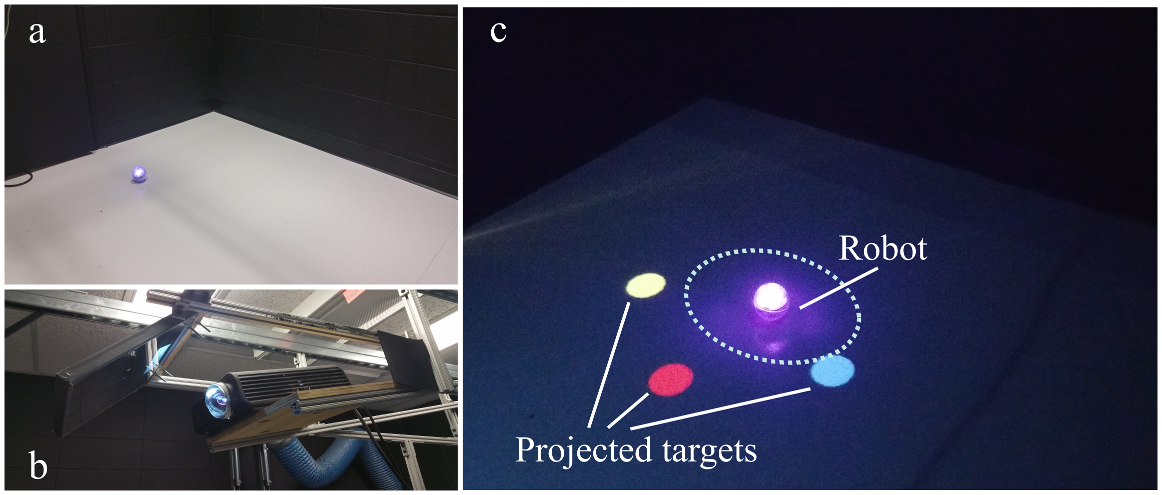


Figure 8.5. (a) The `sphero SPRK` robot is shown in the experimental setup. The internal mechanism shifts the center of mass by rolling and rotating within the spherical enclosure. RGB LEDs on the top of the `sphero SPRK` are utilized to track the odometry of the robot through a webcam using OpenCV for motion capture. The Robot Operating System (ROS, available online [**13**]) is used to transmit and collect data at 20 Hz. A projection (b) is used to project the targets onto the experimental floor shown in (c).

### 8.2.2. Experimental Results

We perform two bearing-only target localization experiments using a `sphero SPRK` robot [**104**] in order to verify real-time execution of Algorithm 4 and showcase the robustness of the algorithm in bearing-only localization. In addition to the robot, the experimental setup involves an overhead projector and a camera, and is further explained in Fig. 8.5. The overhead camera is used to acquire sensor measurements of current robot and target positions that are subsequently transformed to bearing-only measurements as in (8.4). We additionally simulate a limited sensor range as a circle of 0.2 m radius around the current robot position. As in the quadrotor simulation examples, we use an Extended Kalman Filter for bearing-only estimation. In all the following experiments, the ergodic controller runs at approximately 10Hz frequency, i.e., $t_s = 0.1s$ in Algorithm 4.

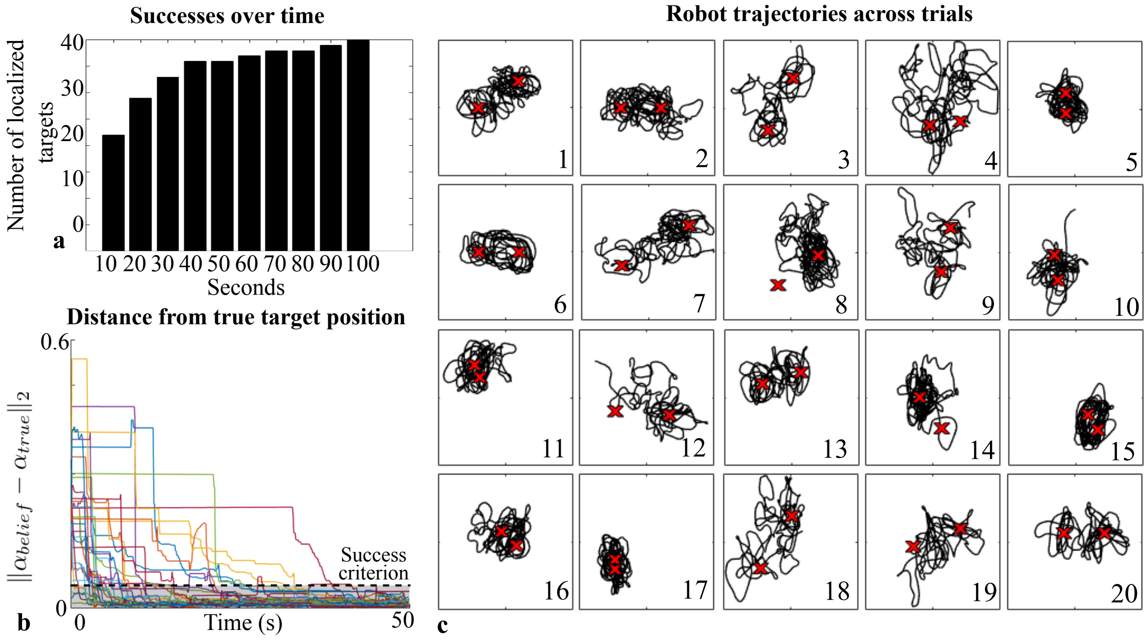


Figure 8.6. Twenty trials of localizing 2 random targets using the `sphero` robot at a 1m×1m terrain with simulated limited sensor range of 0.2m. a) Bar graph showing the number of successful target localizations in specified time intervals. The localization of a target is successful when the $\ell^2$-norm of the difference between the target's position belief and the real target position falls below 0.05, i.e. $||\alpha_{belief} - \alpha_{true}||_2 < 0.05$. Over 50% of the targets (40 in total) are successfully localized within the first 10 seconds and about 90% of the targets are localized by 50 seconds. Even when target detection is delayed or EKF fails to converge in a few iterations, the robot is successful in localizing all the targets by 100 seconds. b) The distance of the mean target estimate from true target position over time across all trials that were complete by the first 50 seconds. Distance remains constant for as long as the target is outside of the sensor range or it has not be detected yet. c) Top-view snapshots of the robot trajectories across trials.

### 8.2.3. Experiment 1

In this Monte Carlo experiment, we perform twenty trials of localizing two static targets randomly positioned in the terrain. For each trial, we consider the localization of a target successful when the $\ell^2$-norm of the difference between the target's position belief and the real target position falls below 0.05, i.e. $||\alpha_{belief} - \alpha_{true}||_2 < 0.05$. To promote variability, initial mean estimates

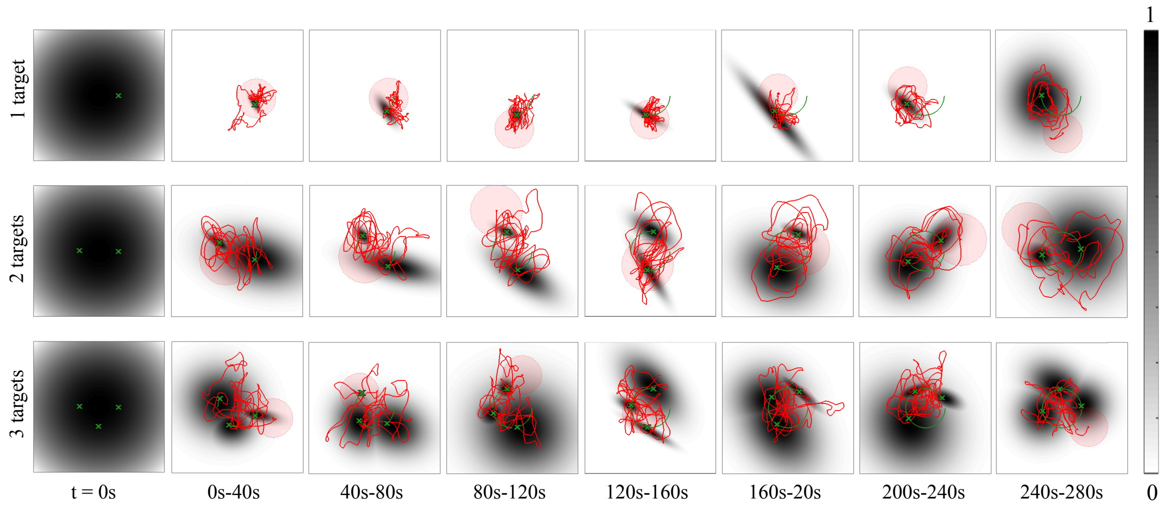


Figure 8.7. Experimental trials localizing 1, 2, and 3 moving targets using the `sphero` robot. The top-view snapshots depict the robot trajectories over a time window of 40 seconds (in red) as well as the targets and their past trajectories (in green). The spatial distribution indicates the targets belief $p(\alpha)$. The robot robustly localizes the moving targets by tracking the expected information density.

of target positions are also randomly selected for each trial. Initial distribution $\Phi(x)$ is uniform inside the terrain boundaries.

The robot simultaneously explores the terrain for undetected targets and localizes detected targets until both targets have been detected, whereafter only localization based on Fisher Information is performed. As in the simulation example of Section 8.2.1.3, we achieve simultaneous exploration and localization by setting the probability of detection (i.e. distribution $\Phi(x)$) across the terrain at a nonzero value. For most trials, targets are successfully localized in less than 60 seconds. We see that even in the few cases when target detection is delayed due to limited sensor range or when EKF fails to converge (as expected for bearing-only models [**175**]) (see trials with time to successful localization higher than 60s in Fig. 8.6a), the robot manages to eventually localize both targets by fully exploring the EID instead of moving towards its maximum

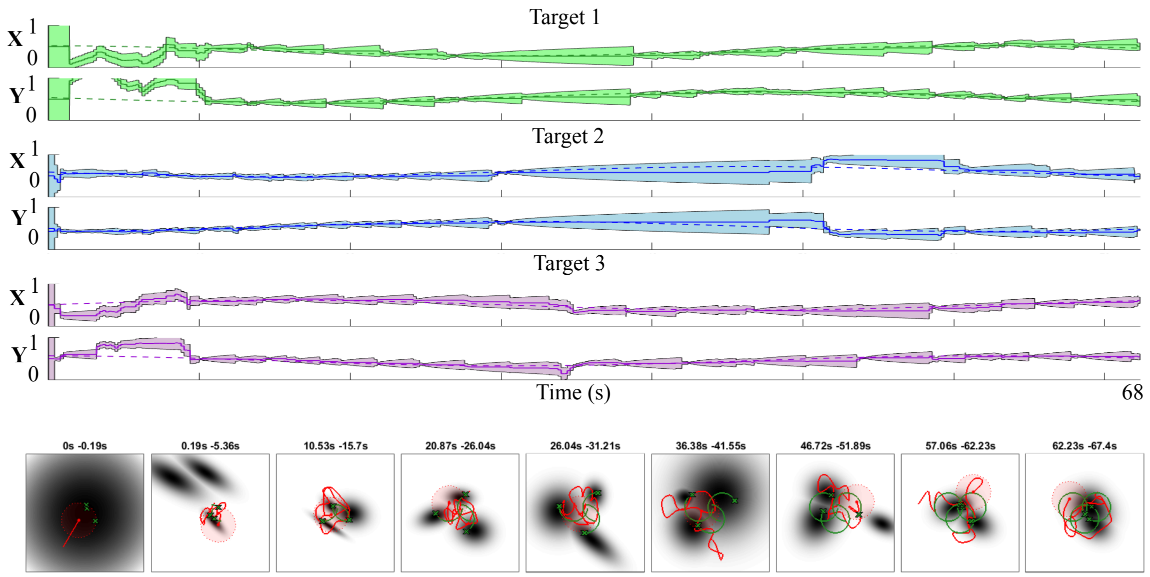


Figure 8.8. Localization of 3 moving targets using the `sphero SPRK` robot. The target estimates (solid curves) are compared to the real target locations (dashed curves) along with an illustration of the belief covariance (shaded area around estimated position) over time. Because the targets are constantly moving and the sensor range is limited, the standard deviation of the targets belief fluctuates as time progresses. The agent localizes each target alternately; once variance on one target estimate is sufficiently low, the agent moves to the next target. Importantly, this behavior is completely autonomous, resulting naturally from the objective of improving ergodicity. Note that we can only decompose the targets belief into separate target estimates because of our choice to use EKF where each target's belief is modeled as a normal distribution. This would not be necessarily true, had we used a different estimation filter (e.g., particle filter). Bottom row shows top-view snapshots of the robot and target's motion. A video of this experiment is available in the supporting multimedia files.

as in information maximization techniques. This result validates Proposition 7 that provides convergence guarantees even with poor target estimates and limited sensor range.

### 8.2.4. Experiment 2

With this experiment, we aim to demonstrate the robustness of the algorithm in localizing increasing number of moving targets. The resulting robot trajectories for localizing 1, 2 and 3 targets are shown in Fig. 8.7, while Fig. 8.8 shows the results for localizing 3 targets moving at a different pattern. As in the simulation examples, the motion of each target is modeled as a diffusion process. Note that because the targets are constantly moving and the sensor range is limited, the standard deviation of the targets belief fluctuates as time progresses (see Fig. 8.8). The agent localizes each target alternately; once variance on one target estimate is sufficiently low, the agent moves to the next target. Importantly, this behavior is completely autonomous, resulting naturally from the objective of improving ergodicity.

CHAPTER 9

# Learning Symbols: Tactile Exploration and Estimation of Physical Object Shapes

In Chapter 8, we made an important assumption when searching for targets-symbols: we assumed that their measurement model, and thus information signature, is known with some uncertainty. This however is hardly ever true. In most cases, we are searching for symbols that have not be identified as such yet and thus their information signature is unknown. Consider for example a mobile robot, equipped with range sensors, moving in an unknown environment while measuring distance from the closest wall. Initially and since we have no previous knowledge of the terrain morphology, the measurements are merely abstract signals. However, as the robot keeps exploring and accumulates information, it essentially builds the information signature of a "symbol" that was previously unknown to it. This symbol is the sensed shape of the physical objects encountered in the terrain. The information signature associated with the symbol can then be communicated to a network of robots, to be added to their internal database (i.e., alphabet) of known and identifiable symbols. For the rest of this chapter, a symbol $s$ is, then, defined as an information signature that uniquely identifies a single physical object or a setting of multiple physical objects on the 2- or 3- dimensional space, based on their shape properties.

## 9.1. Symbol Identification using Ergodic Tactile Exploration

This section describes how the information-driven exploration process developed in Chapter 5 is crucial in achieving real-time shape estimation using the method proposed by Abraham et al. in [**180**]. Tactile exploration—often used in conditions where visual sensing may be limited—employs the sense of touch to search for objects. The richness of touch as a sensing modality is underscored by the development of novel tactile sensors for use in a myriad of applications ranging from robot-assisted tumor detection, to texture recognition, and feature localization. These advances in tactile sensor technology require corresponding advances in active exploration algorithms and the interpretation of tactile-based sensor data.

In this work, we assume that only binary (collision or no-collision) measurements are registered by an agent. This aims to outline how exploration with Algorithm 4 successfully estimates shapes even with low-resolution sensing. In particular, it shows that a binary form of tactile sensing (i.e., collision detection) has enough information for shape estimation, when combined with an active exploration algorithm that automatically takes into account regions of shape information.

The proposed algorithm automatically encodes dynamical constraints without any overhead spatial discretization or motion planning. In addition, the algorithm incorporates sensor measurement information to actively adjust shape estimates and synthesize tactile information based control actions. As a result, the algorithm automatically adjusts the control synthesis for multiple objects in an environment. Finally, ergodic exploration uses non-contact motion data (sensor motion not in contact with an object) to improve the shape estimate.

The measurement model of a symbol $s$ for collision detection measurements (i.e., the mapping from agent states $x$ to sensor measurements $z$) is

$$\Upsilon(s,x) = \begin{cases} 1 & \phi_s(x) \leq 0 \\ 0 & \phi_s(x) > 0 \end{cases} \tag{9.1}$$

where $\phi_s(x)$ is a boundary function that determines a transition state. Then, measurements are taken according to

$$z = \Upsilon(s, x) + \delta \tag{9.2}$$

where $\delta$ represents zero mean Gaussian noise with covariance $\Sigma$, i.e., $\delta \sim \mathcal{N}(0, \Sigma)$. Our objective is to determine the measurement model $\Upsilon(s, x)$, and thus identify symbol $s$, through exploration.

### 9.1.1. Binary Measurements for Shape Estimation

A major advantage of identifying symbols using ergodic exploration is that the process is independent of the probabilistic classification method used for shape estimation. Here, given a set of measurements $z_k \in [0, 1]$ at time indexed by $k$ sampled at the corresponding set of agent state $x_k$ , shape estimation is accomplished by binary classification using support vector machines [**181–183**]. In particular, a decision boundary is approximated as $\phi(x) \approx \sum_k \alpha_k z_k K(x_k, x) + b$, where $K(x_k, x)$ is the kernel basis function that determines the basis shape of the decision boundary and the parameters $\alpha_k$ and $b$ are optimized parameters based on the set of $z_k$ and $x_k$. Because arbitrary shape is desired, a Gaussian kernel basis is chosen, such that $K(x_k, x) = \exp -\frac{(x_k - x)^2}{\sigma^2}$. A kernel of this form maps data into infinite dimensional feature space which provides flexibility for decision boundaries.

Another option for shape estimation is the use of radial basis function neural networks [**184**]. This would not change the ergodic exploration process and would generate results similar to support vector machines described above.

**9.1.1.1. Expected Information Density.** Exploration using Algorithm 4 is information-driven in that it controls agent to be ergodic with respect to a distribution that represents expected

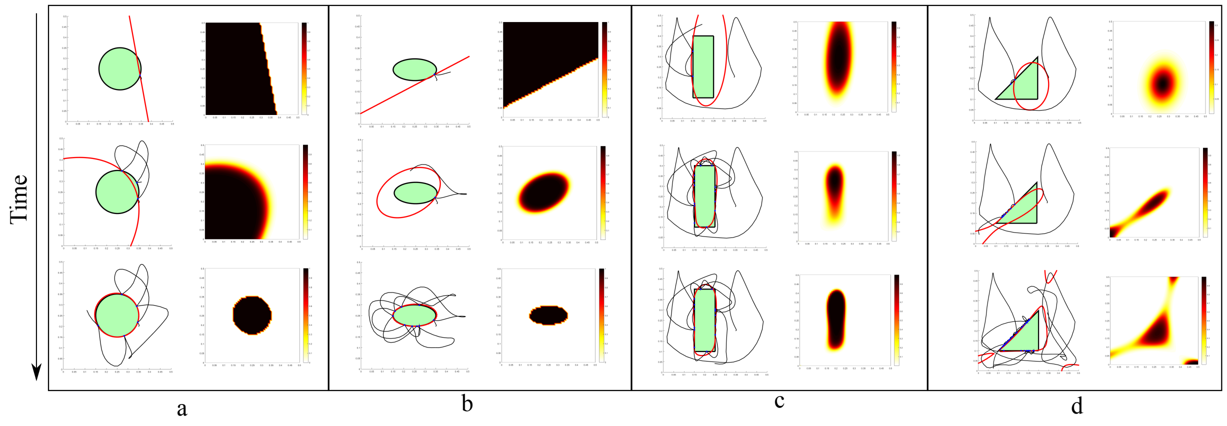


Figure 9.1. Estimating the 2D shapes of (a) a circle, (b) an ellipse, (c) a rectangle, and (d) a triangle using ergodic exploration and contact sensors. The current shape estimate in each snapshot is drawn in red. The trajectories performed by the agent are ergodic with respect to the varying collision likelihood distribution shown on the right column for each shape.

information density. In Section 8.1, we calculated the expected information density using the Fisher Information of the symbol measurement model. Here, however, the measurement model is unknown. Therefore, we design the controller to be ergodic with respect to the likelihood distribution of getting a collision measurement. The posterior probability estimate for collision measurements is calculated via Platt scaling [**185**], defined as $P(z_k = 1|x) = \frac{1}{1+\exp A\phi_s(x)+B}$ where A and B are computed through a regression fit and $\phi_s(x)$ is the current estimate of the boundary function at the $k$-th time step. This probability is used to update the expected information density at each time step.

### 9.1.2. Exploration Outcome: Information Signature of Symbols

For physical objects to be fully sensed and regarded as symbols, there needs to exist some way of identifying them no matter how many times they are explored possibly under different circumstances—e.g., with potential distractor objects on the field. There are a couple of quantities resulting from exploration that are unique to the object $s$ being explored and thus can be used as information signatures that are directly associated with specific symbols. The most obvious is the measurement model $Y(s,x)$ that relates sensor measurements to the object under exploration $s$ and the current robot state $x$. This identifier is coupled with the object shape in two- or three-dimensional exploration. In SVM classification, the measurement model is extracted based on the calculated boundary $\phi(x)$ associated with $s$. Interestingly, information included in measurement models—just like object shape—is invariant under transformation or scaling.

Although this is the most reasonable choice for defining a symbol identity—especially since here symbols *are* object shapes, ergodic exploration provides us with an additional option because of its distributed nature. Specifically, the spatial statistics of the object exploration trajectory $C(x) = \frac{1}{T}\int\limits_{t_0}^{t_0+T} \delta[x - x_v(t)]dt$ parameterized with Fourier coefficients $c_k = \frac{1}{T}\int_{t_0}^{t_0+T} F_k(x_v(t))dt$ are unique to each symbol/object shape. Therefore, we could detect previously explored symbols based on a dictionary of exploratory robot trajectories and their statistics. What makes this approach interesting is that, in addition to shape information, we also incorporate information about the robot agent's dynamics in the symbol identity so that detection can be successful even for complex shapes that are hard to estimate using binary classification or other machine learning techniques.

### 9.1.3. Results

**9.1.3.1. Simulation Results.** In this section, simulation examples in both 2D and 3D are presented to highlight the advantages for using Algorithm 5 for shape estimation.

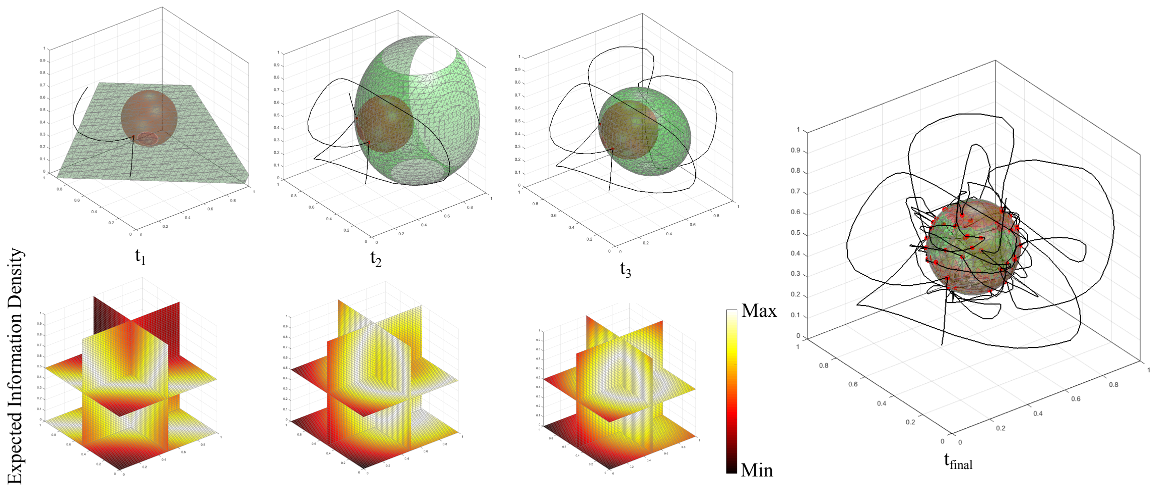


Figure 9.2. Estimating the shape of a sphere using collision measurements. Actual sphere is depicted in light red and the estimated shape in light green. The agent's trajectory up to the current time step is also shown, with the collision measurements highlighted in red. The agent is controlled to be ergodic with respect to the likelihood distribution, illustrated at the bottom row. The final shape matches closely the sensed sphere.

*2D Examples:* Symbol identification through active shape estimation in $\mathbb{R}^2$ is done with double integrator dynamics given as $\dot{x} = f(x, u) = [\dot{\mathrm{x}}, \dot{\mathrm{y}}, u_1, u_2]^T$ with the state $x = [\mathrm{x}, \mathrm{y}, \dot{\mathrm{x}}, \dot{\mathrm{y}}]^T$ and x, y the 2D position coordinates of the agent. Fig. 9.1 shows the process of estimating four common shapes using ergodic exploration. The expected information density, calculated as the likelihood of acquiring collision measurements, is also shown for each step. We can see that the ergodic controller (Algorithm 5) naturally adapts to changes in the distribution $\Phi(x)$ as the shape estimate changes in time. Algorithm 5 explores the current shape estimate online without the requirement of a pre-optimized exploration trajectory. Distributed-information exploration (as opposed to maximum-information) allows us to pursue non-collision measurements, in addition to collision measurements. Furthermore, the ergodic controller is shown to be reliable even when operating in unmodeled environments: although control calculation in Algorithm 4

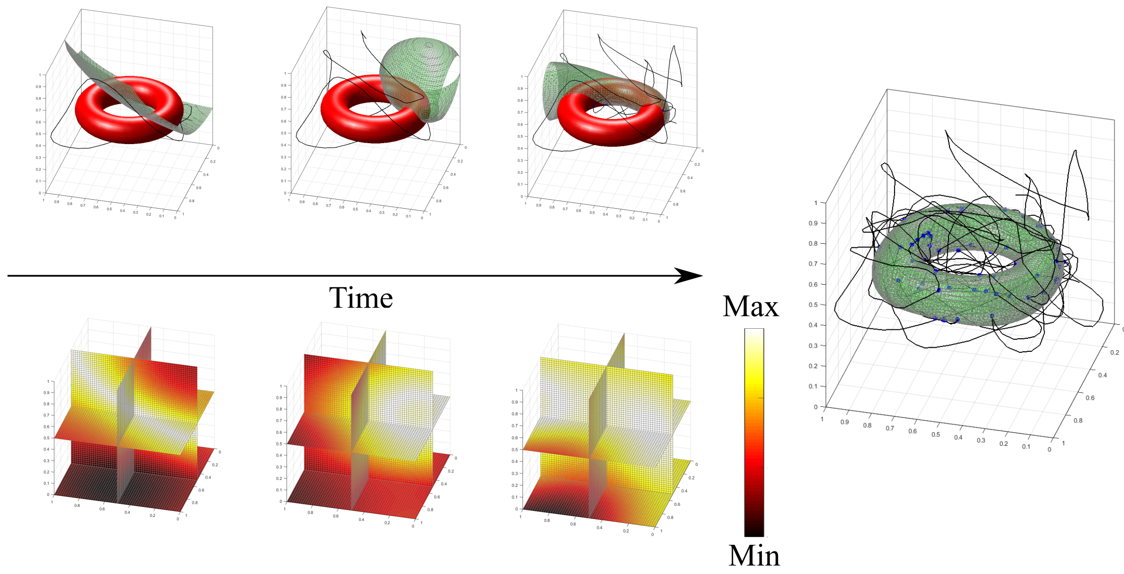


Figure 9.3. Estimating the shape of a torus using collision measurements. Actual object is depicted in red and the estimated shape in light green. The agent's trajectory up to the current time step is also shown, with the collision measurements highlighted in red. The agent is controlled to be ergodic with respect to the likelihood distribution, illustrated at the bottom row. The final shape matches closely the sensed torus.

assumes a collision-free environment, the controller is reactive and responds to collisions online as they happen.

*3D Examples:* Symbol identification through active shape estimation in $\mathbb{R}^3$ is done with double integrator dynamics given as $\dot{x} = f(x,u) = [\dot{\mathrm{x}}, \dot{\mathrm{y}}, \dot{\mathrm{z}}, u_1, u_2, u_3]^T$ with the state $x = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \dot{\mathrm{x}}, \dot{\mathrm{y}}, \dot{\mathrm{z}}]^T$ and $\mathrm{x}, \mathrm{y}, \mathrm{z}$ the 3D position coordinates of the agent. Fig. 9.2 and 9.3 show the results of estimating the shapes of a sphere and a torus, respectively, using collision measurements. In both cases, the resulting 3D shapes closely match the objects explored. The results in Fig. 9.4 indicate how Algorithm 5 can successfully explore and estimate the shapes of multiple objects on the terrain without indefinitely focusing on a single object. This is due to the

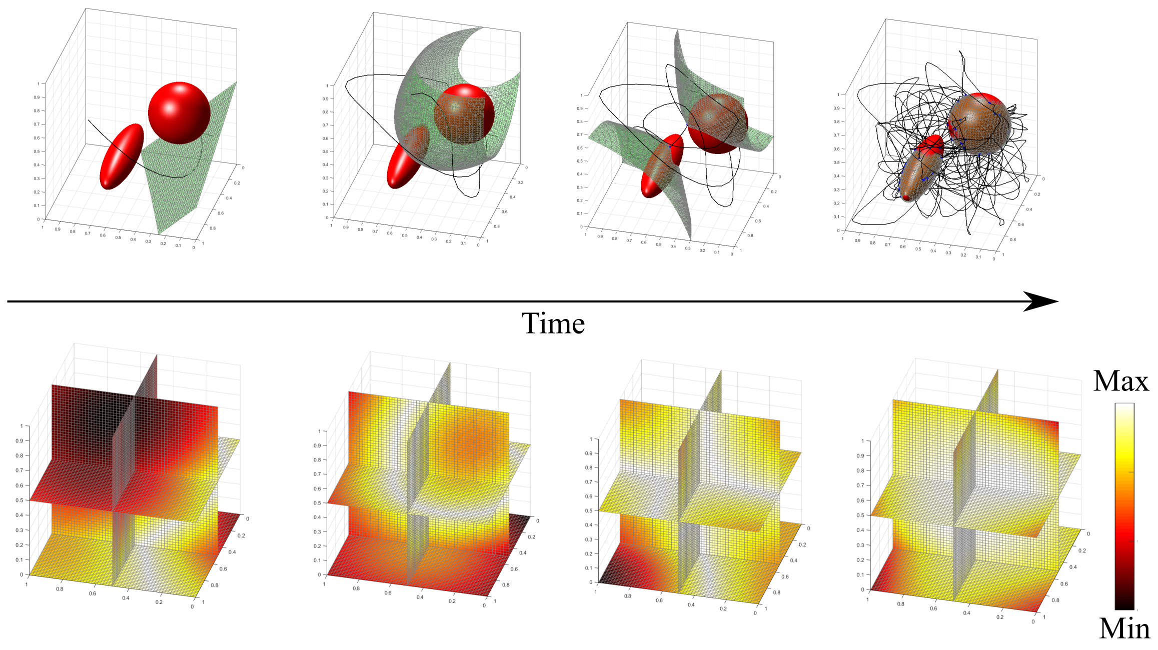


Figure 9.4. Estimating the shapes of two objects simultaneously using collision measurements. The final shapes matches closely the sensed objects. Ergodic exploration promotes estimation of multiple symbols because it is distribution-driven.

fact that the exploration algorithm tracks an information distribution instead of the maximum information.

**9.1.3.2. Experimental Results.** We use the `sphero SPRK` robot, shown in Fig. 9.5, to explore a terrain and estimate the 2D shapes of physical objects. The robot is equipped with a collision detection feature that we use to acquire binary (collision/no-collision) measurements in real time. Our objective is to "learn" an unknown environment—i.e., extract the measurement model $\Upsilon(s, x)$ and store it as a symbol, so that a robot can subsequently use it to localize itself based on the symbol configuration. This last process of estimating a previously-explored symbol's transformation (and, thus, the robot pose with respect to this symbol) highlights the power of

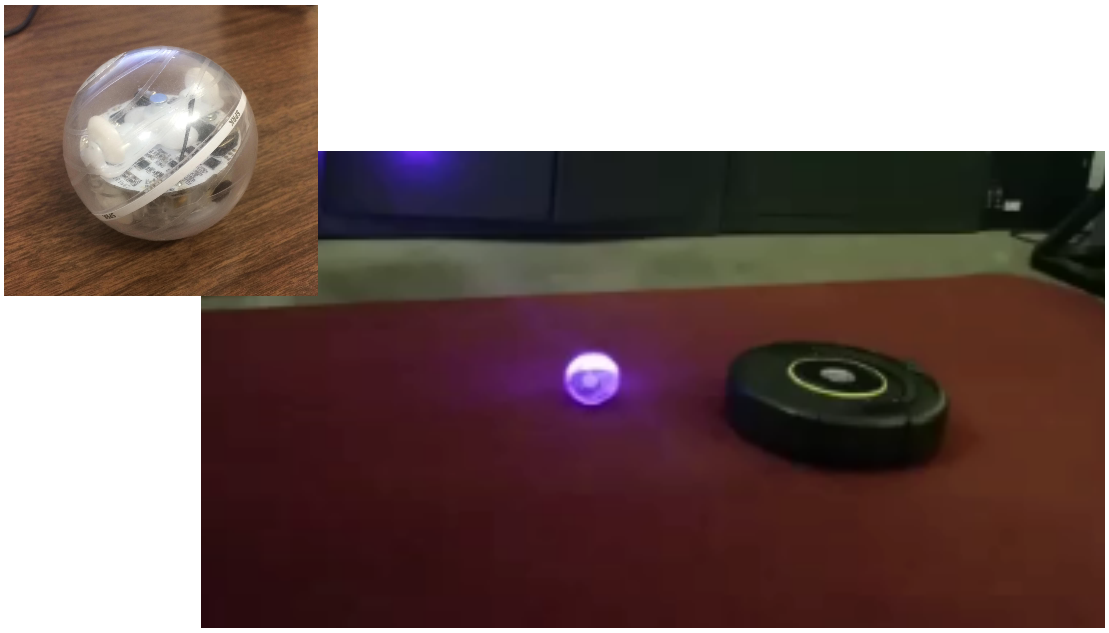


Figure 9.5. The sphero SPRK robot and the experiment setup (the Roomba robot is used only as the object of estimation). The sphero SPRK internal mechanism shifts the center of mass by rolling and rotating within the spherical enclosure. RGB LEDs on the top of the sphero SPRK are utilized to track the odometry using a Microsoft Kinect. The robot is equipped with a collision detection feature that we use to acquire binary (collision/no-collision) measurements in real time. The Robot Operating System (ROS, available online [**13**]) is used to transmit and collect data at 20 Hz.

information equivalence, in that it enables exploration and detection of information signatures independently of the factor of transformation.

Fig. 9.6 shows snapshots of the sphero robot estimating the circular shape of a Roomba. The robot is controlled to be ergodic with respect to the likelihood distribution $\Phi(x)$ in Algorithm 5, calculated as indicated in Section 9.1.1.1, and also shown in Fig. 9.6.

In the next experiment, the sphero robot estimates the shape of multiple objects (representing one symbol) positioned in the terrain. The results are shown in Fig. 9.7. Notice how the

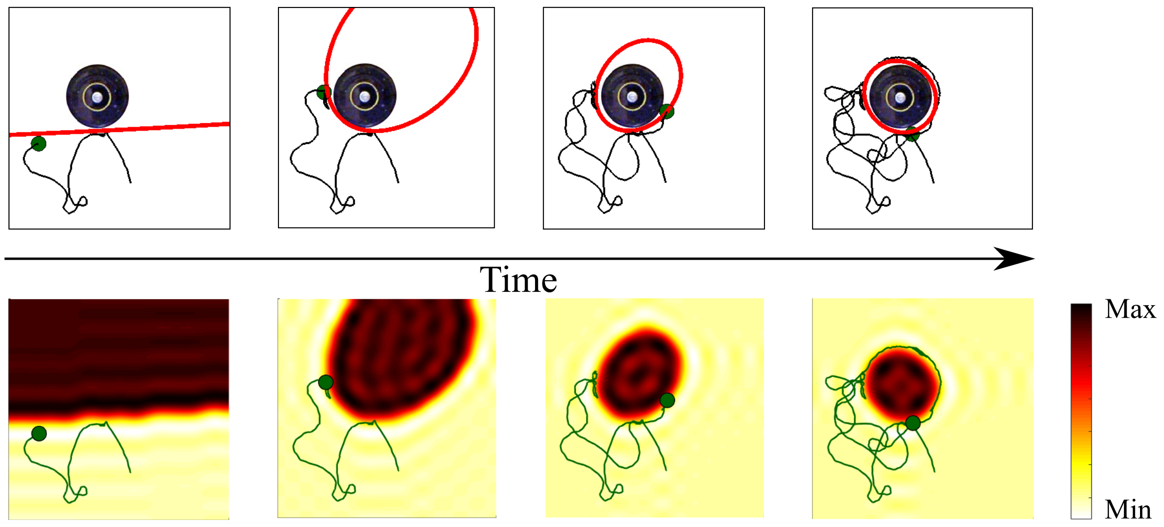


Figure 9.6. Estimating the shape of a circular object using the sphero robot and collision measurements. A picture of the experimental setup is shown in Fig. 9.5. The shape estimate at each snapshot is drawn in red. Bottom row shows the collision likelihood distribution. The sphero is controlled to be ergodic with respect to this distribution.

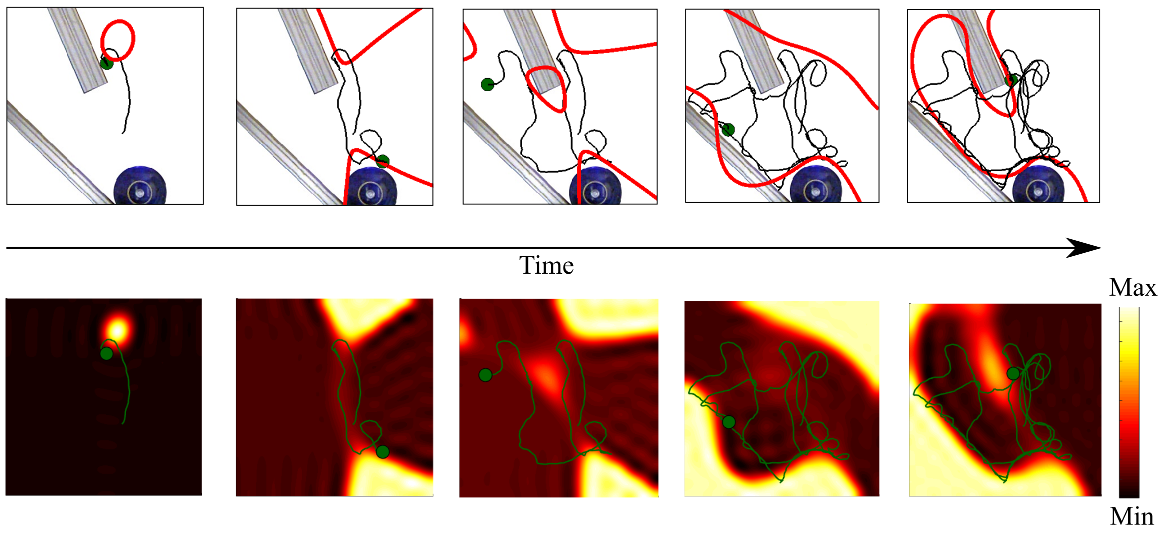
symbol estimate is gradually evolving as the robot acquires more collision and no-collision measurements. Although the final estimate is not a perfect outline of the object shapes, it can still be used as the symbol information signature that would allow a robot to detect the symbol's transformation in a different configuration.

## 9.2. Information Equivalence in Ergodic Exploration

### 9.2.1. Why Information Equivalence Matters (but only Combined with Exploration)

We define the concept of information equivalence in automated systems, as follows:

**Definition 8.** *Two candidate symbols $s_1$ and $s_2$ are "information equivalent" when an information optimizing control $u$—and corresponding exploration trajectory—is the same across the state $x$ measured with respect to the local symbol frame (that is, $u_1(x) = u_2(x)$).*



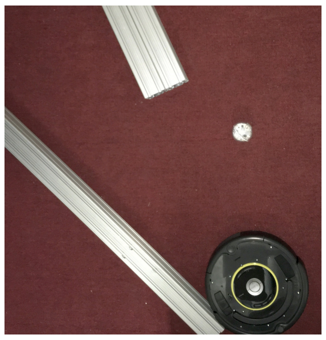


Figure 9.7. Bottom: Photo of the experiment setup. Top: Time series of the exploration and estimation process from left to right. The red boundaries indicate the estimated shapes at each time step. Sphero robot is shown as a green circle. The robot trajectory up to current time is also shown.

The importance of information equivalence stems from the need to perform decision and control tasks in a manner that is independent of factors such as scaling and transformation. In Section 8.1, we showed how to compute the information signature of a target using Fisher information. This information is equivalently explored in all settings; it is transformation independent in the context of exploration. The concept of information equivalence, in this case, allowed us to track the target using the same information regardless of its state or the agent's state. Therefore, information equivalence matters but it only does because we hold the knowledge of exploring with respect to different levels of information in a distributed manner (see Part 2). In this section, we go further into investigating how the concept of information equivalence allows us to track symbols that have been generated through exploration (using the methodology described in Section 9.1). In particular, we take the extracted information signature (consisting primarily of the symbol's measurement model) and use it to estimate the symbol transformation (translation and rotation) during a new exploration process, possibly by a different agent.

### 9.2.2. Exploration for Estimating Symbol Transformation

Let $x_{\mathcal{W}} \in \mathcal{X}$ be the state of a dynamical agent which ergodically explores a symbol on a $\nu$-dimensional terrain with respect to a world coordinate frame $\mathcal{W}$. A symbol $s$ can be a single physical object or a setting of multiple physical objects on the $2-$ or $3-$dimensional space ($\nu = 2$ and $\nu = 3$ respectively). We assume that the symbol has been previously explored by an agent, resulting in the extraction of the symbol measurement model $\Upsilon(s, x_{\mathcal{S}})$ that maps robot states $x_{\mathcal{S}}$ with respect to the symbol coordinate frame $\mathcal{S}$ to robot sensor measurements. The objective is to estimate the transformation $T_{\mathcal{WS}} \in SE(\nu)$ from the world coordinate frame $\mathcal{W}$ to the symbol coordinate frame $\mathcal{S}$. Due to information equivalence, a measurement is always made according to the symbol measurement model $\Upsilon(s, x_{\mathcal{S}})$, regardless of the transformation $T_{\mathcal{WS}}$. However, we can only measure the agent's state $x_{\mathcal{W}}$ in the world frame and thus we express measurements $z$ as follows:

$$z = \Upsilon(s, T_{\mathcal{WS}}^{-1} \cdot x_{\mathcal{W}}) + \delta, \tag{9.3}$$

where $\delta$ represents zero mean Gaussian noise with covariance $\Sigma$, i.e., $\delta \sim \mathcal{N}(0, \Sigma)$.

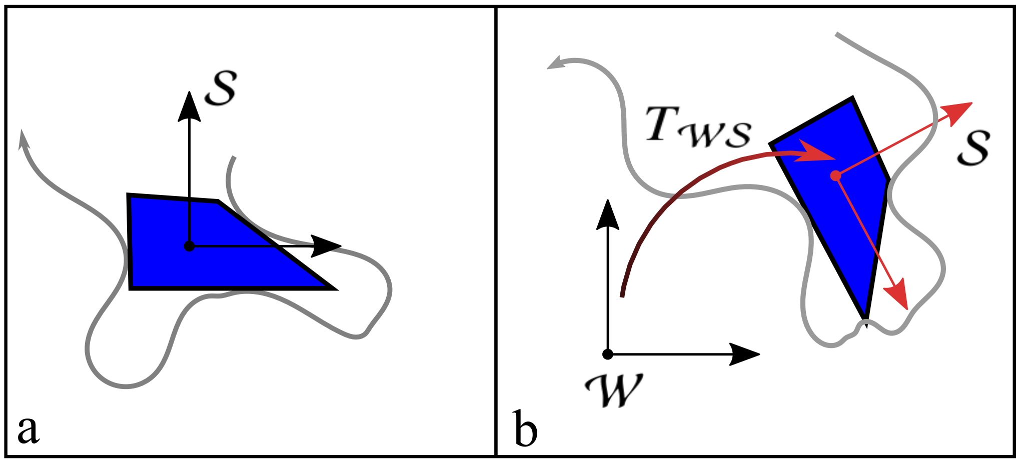


Figure 9.8. Estimating symbol transformation: Illustration of the problem statement. (a) A symbol is initially explored by an agent to extract its information signature (i.e. measurement model) with respect to the symbol coordinate frame. (b) An agent explores the extracted symbol acquiring measurements with respect to a world coordinate frame. The agent's objective is to estimate the Euclidean transformation from the world coordinate frame to the symbol's coordinate frame, so that it can localize itself in the environment using the unmodified symbol as reference.

**9.2.2.1. Expected Information Density.** We focus on calculating the expected information measurement $EID(x)$ given the current belief on transformation $T_{\mathcal{WS}}$ and the measurement model (9.3). It is important to point out that the following process for computing the EID depends only on the measurement model, meaning that the methodology for belief state representation and update can be arbitrary (e.g., Bayesian methods, Kalman filter, particle filter etc.).

As in [**101**], we will use the Fisher Information Matrix (FIM) [**169, 170**] to calculate the EID. Often used in maximum likelihood estimation, Fisher information $\mathcal{I}(x, q)$ is the amount of information a measurement provides at location $x$ for a given estimate of $q$. It quantifies the ability of a set of random variables, in our case measurements, to estimate the unknown parameters. We know that transformation $T_{\mathcal{WS}}$ can be parameterized by $q \in \mathbb{R}^{\mu}$ with $\mu = \frac{1}{2}\nu(\nu +$

1), denoting the translational and rotational degrees of freedom. Then, the Fisher information is represented as a $\mu \times \mu$ matrix and the $(i, j)$th FIM element is calculated as

$$\mathcal{I}_{i,j}(x_{\mathcal{W}}, q) = \frac{\partial \Upsilon(s, T^{-1}_{\mathcal{WS}}(q) \cdot x_{\mathcal{W}})}{\partial q_i}^T \Sigma^{-1} \frac{\partial \Upsilon(s, T^{-1}_{\mathcal{WS}}(q) \cdot x_{\mathcal{W}})}{\partial q_j} \tag{9.4}$$

Since the estimate of transformation parameters $q$ is represented as a probability distribution function $p(q)$, we take the expected value of each element of $\mathcal{I}(x_{\mathcal{W}}, q)$ with respect to the joint distribution $p(q)$ to calculate the expected information matrix, $\mathbf{\Phi}_{i,j}(x_{\mathcal{W}})$. The $(i, j)$th element of $\mathbf{\Phi}_{i,j}(x_{\mathcal{W}})$ is then

$$\mathbf{\Phi}_{i,j}(x_{\mathcal{W}}) = \int_q \mathcal{I}_{i,j}(x_{\mathcal{W}}, q) p(q)\, dq. \tag{9.5}$$

To reduce computational cost, this integration is performed numerically by discretization of the estimated parameters on a grid and a nested summation.

In order to build a density map using the information matrix (8.2), we need a metric so that each state $x$ is assigned a single information value. We will use the following mapping:

$$\Phi(x_{\mathcal{W}}) = \det \mathbf{\Phi}(x_{\mathcal{W}}). \tag{9.6}$$

The FIM determinant (D-optimality) is widely used in the literature, as it is invariant under re-parameterization and linear transformation [**171**]. A drawback of D-optimality is that it might result in local minima and maxima in the objective function, which makes optimization difficult when maximizing information. In our case though, local maxima do not pose an issue as our purpose is to approximate the expected information density using ergodic trajectories instead of maximizing it.

**9.2.2.2. Transformation Estimate Update.** To update the transformation estimate, we use the contact manifold particle filter introduced in [**186**]. Contact sensors are unique because they discriminate between contact and no-contact configurations. As a result, the set of symbol states that activates the sensor constitutes a lower-dimensional contact manifold in the space of the transformation. This causes conventional state estimation methods, such as the particle filter, to perform poorly during periods of contact due to particle starvation. The manifold particle filter avoids particle starvation during contact by adaptively sampling particles that reside on the contact manifold from the dual proposal distribution.

**9.2.2.3. Relation to Landmark-based Localization.** A key component of a mobile robot system is the ability to localize itself accurately. A prominent way of achieving this is by using landmarks. Most early successful approaches proposed using artificial landmarks [**187**], such as bar-code reflectors, ultrasonic beacons, etc. However, these methods require processed and properly modified environments to function correctly. Lately, researchers have turned their focus to using stable natural landmarks in unmodified environments [**188**, **189**]. The main challenge in achieving this is that the selected landmarks must be identifiable using the available sensors, and subsequently, invariant to transformations (rotation and translation). When equipped with visual sensors for example, robots can use scale-invariant image features generated by known image processing techniques such as SIFT (scale-invariant feature transform) [**188**].

The method proposed in this chapter—first, exploration for information signature extraction and then, exploration for transformation estimation—is essentially a robot pose estimation method using natural, previously explored, landmarks. Physical object shapes are recognized as landmarks and the measurement model extracted during exploration is the signature that allows

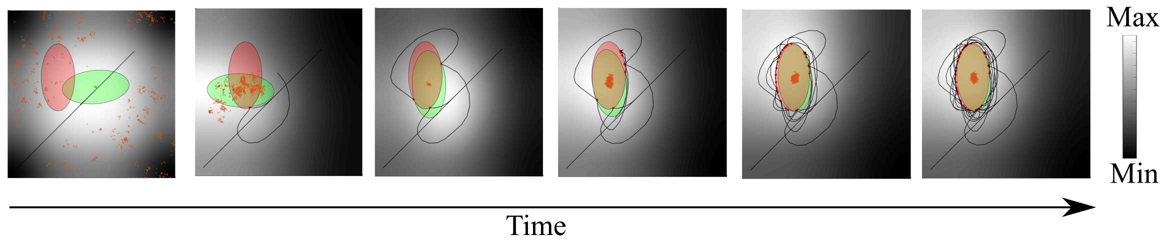


Figure 9.9. Estimating the transformation of a previously explored ellipse. The actual ellipse (position and orientation) is shown in light red. The current transformation estimate is shown as a light green ellipse. The position ($[t_x, t_y]]$) particle filters, also depicted here, are shown to converge to the center of the explored ellipse.

us to track them under transformation using ergodic control. By estimating $T_{WS}$ in 9.3, we can localize the robot based on a known landmark of an unmodified environment.

### 9.2.3. Results

**9.2.3.1. Simulation Results.** In this section, we apply the method described in Section 9.2.2 in order to estimate the transformation of previously explored symbols. Transformation is parameterized by the rotation angle $\theta$ and translation in x- and y-direction $t_x$ and $t_y$, respectively, so that $q = [\theta, t_x, t_y]$ in (9.5)) . The agent is controlled to be ergodic with respect to an expected information density calculated as in (9.6). We use the contact manifold particle filter proposed in [**186**] to update the transformation estimate using collision and no-collision measurements. Fig. 9.9 shows the results of estimating the transformation of a previously explored ellipse. In Fig. 9.10, we go further into estimating the transformation of a set of object shapes that serve as one symbol (i.e. landmark). Notice how the agent adapts its exploration trajectory to the current transformation estimate by being ergodic with respect to the expected information density shown in the bottom row.

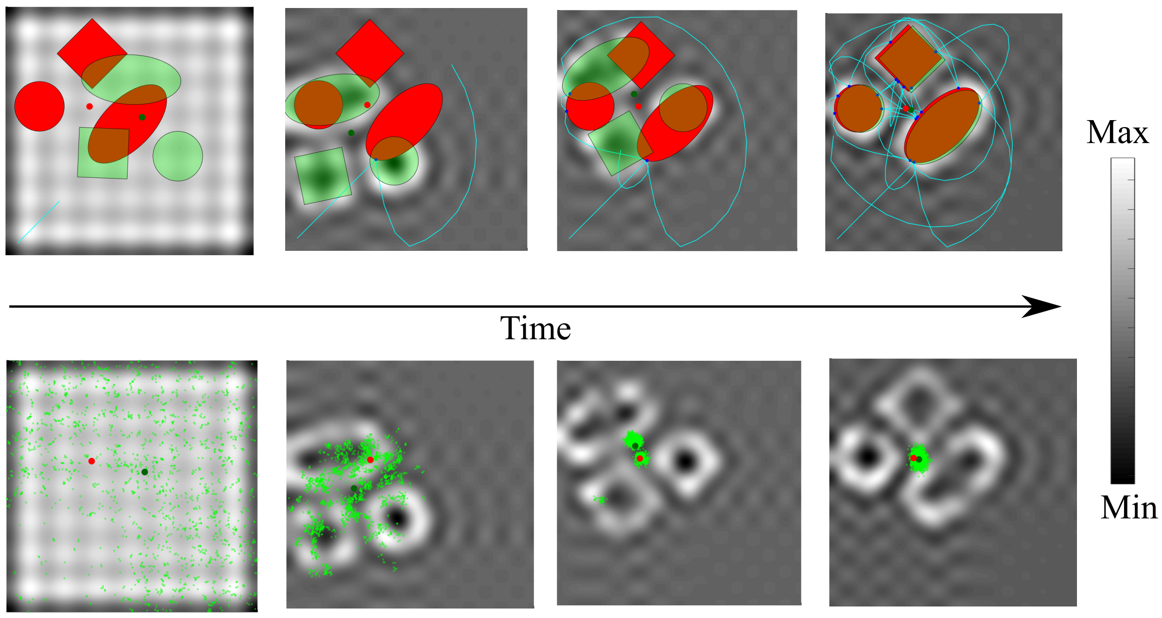


Figure 9.10. Estimating the transformation of a set of objects representing a single symbol (i.e. landmark). The actual symbol is drawn in red, while the estimated symbol is drawn in green. The bottom row shows the position particle filters ($[t_x, t_y]]$). The agent estimates the symbol configuration by using information equivalence to explore with respect to the information signature of the symbol. Once the symbol transformation has been successfully estimated, the agent can use it to localize itself with respect to the unmodified landmark instead of a randomly generated world coordinate frame.

**9.2.3.2. Experimental Results.** Here, we use the experimental setup from Section 9.1.3.2, depicted in Fig. 9.5, in order to estimate the transformation of previously explored symbols using collision measurements. First, the robot is controlled to ergodically explore a terrain in order to estimate the transformation of the circular object originally explored in Fig. 9.6. The results are shown in Fig. 9.11. Note how the robot adaptively explores with respect to the expected information density of the estimated symbol transformation (drawn in purple). In Fig. 9.12, the robot is controlled to estimate the transformation of the environment originally explored in Fig. 9.6. As noted before, although the extracted information signature does not match exactly the shape outline of the object, the robot is still able to successfully estimate the transformation

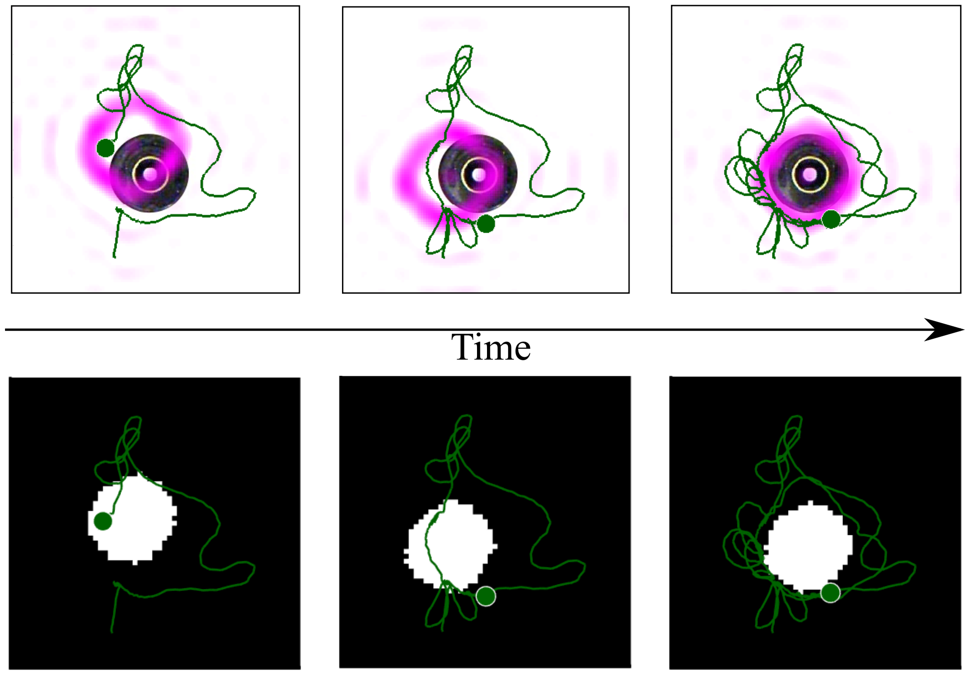


Figure 9.11. The `sphero SPRK` robot estimates the transformation of the symbol explored in Fig. 9.6 using collision measurements. Top row shows the actual object, the robot trajectory up to current time, as well as the expected information density calculated as in Section 9.2.2.1. Bottom row shows the configuration of the estimated symbol at each time step (in white). The robot is successful in estimating the current position of the circular object.

of the landmark/symbol. The reason is that controller tracks an information distribution instead of the maximum information, and thus is more robust to information discrepancies. This example is a preliminary indication that information-driven exploration can be the key to robot self-localization using stable natural landmarks.

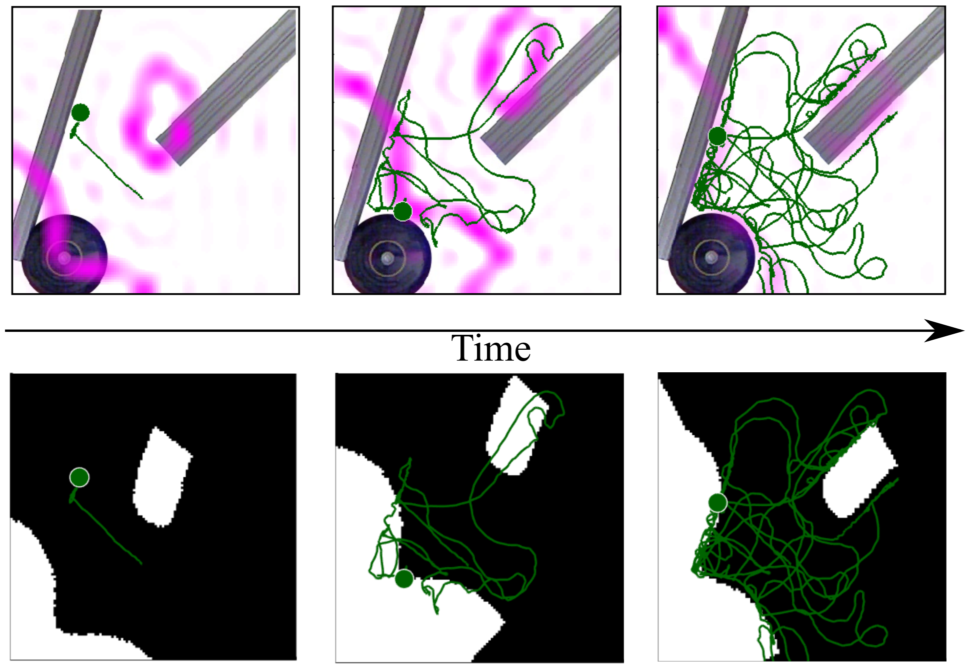


Figure 9.12. The `sphero SPRK` robot estimates the transformation of the symbol (a set of multiple physical objects) explored in Fig. 9.7 using collision measurements. Top row shows the actual objects setup that is being estimated, the robot trajectory up to current time, as well as the expected information density calculated as in Section 9.2.2.1. Bottom row shows the configuration of the estimated symbol at each time step (in white). The robot is successful in estimating the current configuration of the symbol and thus can use it as a stable landmark for self-localization.

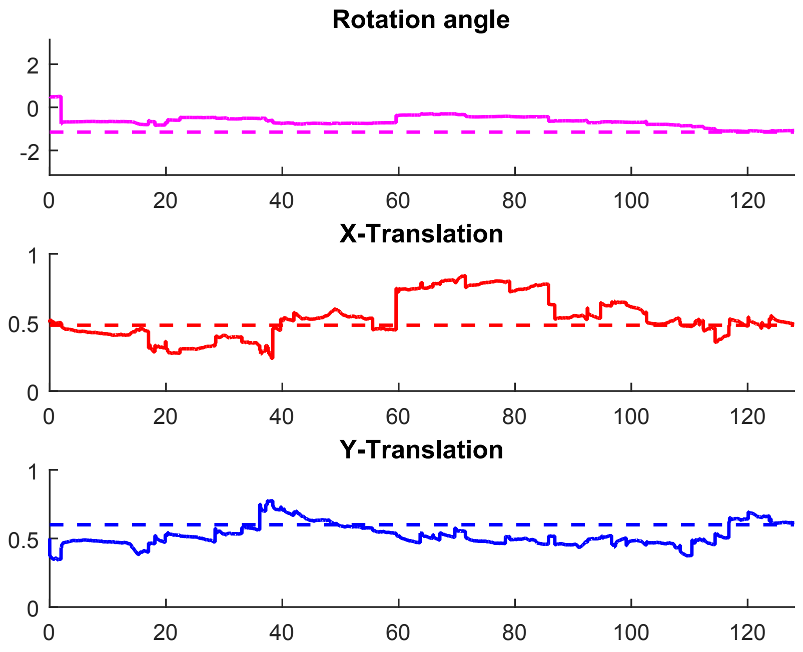


Figure 9.13. Time evolution of estimated symbol rotation and translation against true symbol transformation (dashed lines). Fluctuations in estimated parameters result from the contact manifold particle filter used for estimation, as new particles are generated once there is a contact.

CHAPTER 10

# Conclusion

The contribution of this thesis spans three aspects of symbol-based automation, namely action, exploration and sensing.

In symbolic action, our objective was to achieve fast and consistent, real-time mode scheduling by taking advantage of linearity of a switched system. In general, mode scheduling is challenging due to the fact that both the mode sequence and the set of switching times must be optimized jointly. We addressed this by introducing an algorithm (SIOMS) for scheduling the modes of linear time-varying switched systems subject to a quadratic cost functional. By solving a single set of differential equations offline, open-loop SIOMS required no online simulations while closed-loop SIOMS only involved an integration over a limited time interval rather than the full time horizon. The proposed algorithm is fast and free of the trade-off between execution time and approximation errors. To verify the efficacy of receding-horizon SIOMS in real-world applications, we performed a real-time experiment using ROS. Our experimental work demonstrated that a cart and suspended mass system can be regulated in real time using closed-loop hybrid control signals.

Furthermore, we presented a method for synthesis of control alphabet policies, given continuum descriptions of physical systems and tasks. During synthesis, symbolic policies were directly encoded into finite state machines using a cell subdivision approach. As opposed to existing automata synthesis methods, controller synthesis was based entirely on the original nonlinear system dynamics and thus did not rely on but rather resulted in a lower-complexity symbolic representation. The method was validated for the cart-pendulum inversion problem, the double-tank system and the SLIP model. The approach presents an opportunity for real-time task-oriented control of complex robotic platforms using exclusively sensor data with no online computation involved.

In exploration, we exploited the advantages of hybrid systems theory to formulate a receding-horizon ergodic control algorithm that can perform real-time exploration, adaptively using sensor feedback to update the expected information density. In target localization, this ergodic motion planning strategy controls the robots to track a non-parameterized information distribution across the terrain instead of individual targets independently, thus being completely decoupled from the estimation process and the number of targets. We demonstrated—in simulation with a 12-dimensional UAV model and in experiment using the `sphero SPRK` robot—that ergodically controlled robotic agents can reliably track moving targets in real time based on bearing-only measurements even when the number of targets is not known a priori and the targets motion is only modeled as a diffusion process. Finally, the simulation and experiment examples served to highlight the importance of and to verify stability of the ergodic controls with respect to the expected information density, as proved in our theoretical results.

Finally, in symbolic sensing, we showed how an agent can learn the information signature of unknown symbols, in the context of tactile exploration for estimating physical object shapes. The ability to perform distribution-driven exploration online, allowed us to take advantage of information equivalence and track previously explored shapes/symbols with varied configurations. This final result highlighted the importance of exploration in symbolic sensing and opened the road for using previously explored symbols as natural landmarks for robot self-localization.

# References

[1] A. D. Wilson, J. Schultz, and T. D. Murphey, “Trajectory optimization for well-conditioned parameter estimation,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 28–36, 2015.

[2] ——, “Trajectory synthesis for Fisher information maximization,” *IEEE Transactions on Robotics*, vol. 30, pp. 1358–1370, 2014.

[3] M. Mazo and P. Tabuada, “Symbolic approximate time-optimal control,” *Systems & Control Letters*, vol. 60, no. 4, pp. 256–263, 2011.

[4] G. Pola, A. Girard, and P. Tabuada, “Approximately bisimilar symbolic models for nonlinear control systems,” *Automatica*, vol. 44, no. 10, pp. 2508 – 2516, 2008.

[5] M. Broucke, M. D. Di Benedetto, S. Di Gennaro, and A. Sangiovanni-Vincentelli, “Theory of optimal control using bisimulations,” in *Hybrid Systems: Computation and Control*. Springer, 2000, pp. 89–102.

[6] A. Girard and G. Pappas, “Approximation metrics for discrete and continuous systems,” *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[7] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[8] P. Skoglar, U. Orguner, and F. Gustafsson, "On information measures based on particle mixture for optimal bearings-only tracking," in *IEEE Aerospace conference*, 2009, pp. 1–14.

[9] F. Bourgault, A. Göktogan, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated search for a lost target in a Bayesian world," *Advanced Robotics*, vol. 18, no. 10, pp. 979–1000, 2004.

[10] S. S. Ponda, R. M. Kolacinski, and E. Frazzoli, "Trajectory optimization for target localization using small unmanned aerial vehicles," in *AIAA Guidance, Navigation, and Control Conference*, 2009, pp. 10–13.

[11] Y. Oshman and P. Davidson, "Optimization of observer trajectories for bearings-only target localization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 3, pp. 892–902, 1999.

[12] T. H. Summers, M. R. Akella, and M. J. Mears, "Coordinated standoff tracking of moving targets: Control laws and information architectures," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 1, pp. 56–69, 2009.

[13] (2014) Robot Operating System. Willow Garage. [Online]. Available: http://www.ros.org/

[14] S. Soatto, “Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control,” *arXiv preprint arXiv:1110.2053*, 2011.

[15] A. Mavrommati, J. Schultz, and T. D. Murphey, “Real-time dynamic-mode scheduling using single-integration hybrid optimization,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1385–1398, 2016.

[16] A. Mavrommati and T. D. Murphey, “Single-integration mode scheduling for linear time-varying switched systems,” in *American Control Conference*, 2014, pp. 3948–3953.

[17] X. Deng, L. Schenato, and S. S. Sastry, “Flapping flight for biomimetic robotic insects: Part II-flight control design,” *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 789–803, 2006.

[18] S. Lee and V. Prabhu, “A dynamic algorithm for distributed feedback control for manufacturing production, capacity, and maintenance,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 628–641, 2015.

[19] H.-W. Park, A. Ramezani, and J. Grizzle, “A finite-state machine for accommodating unexpected large ground-height variations in bipedal robot walking,” *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 331–345, 2013.

[20] A. Mavrommati and T. D. Murphey, “Automatic synthesis of control alphabet policies,” in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 313–320.

[21] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, “Real-time area coverage and target localization using receding-horizon ergodic exploration,” *IEEE Transactions on Robotics*, Submitted, 2017.

[22] M. Castano, A. Mavrommati, T. D. Murphey, and X. Tan, “Trajectory planning and tracking of robotic fish using ergodic exploration,” in *American Control Conference*, Accepted, 2017.

[23] A. Prabhakar, A. Mavrommati, J. Schultz, and T. D. Murphey, “Autonomous visual rendering using physical motion,” in *Workshop on the Algorithmic Foundations of Robotics*, 2016.

[24] Y. Wardi and M. Egerstedt, “Algorithm for optimal mode scheduling in switched systems,” in *American Control Conference*, 2012, pp. 4546–4551.

[25] S. Wei, K. Uthaichana, M. Žefran, R. A. DeCarlo, and S. Bengea, “Applications of numerical optimal control to nonlinear hybrid systems,” *Nonlinear Analysis: Hybrid Systems*, vol. 1, no. 2, pp. 264–279, 2007.

[26] T. Johansen, I. Petersen, J. Kalkkuhl, and J. Lüdemann, “Gain-scheduled wheel slip control in automotive brake systems,” *IEEE Transactions on Control Systems Technology*, vol. 11, no. 6, pp. 799–811, 2003.

[27] R. O’Flaherty and M. Egerstedt, “Low-dimensional learning for complex robots,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 19–27, 2015.

[28] Y. Qiao, N. Wu, and M. Zhou, “Real-time scheduling of single-arm cluster tools subject to residency time constraints and bounded activity time variation,” *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 3, pp. 564–577, 2012.

[29] S. Mariéthoz, S. Almér, M. Bâja, A. Beccuti, D. Patino, A. Wernrud, J. Buisson, H. Cormerais, T. Geyer, H. Fujioka, U. Jonsson, C.-Y. Kao, M. Morari, G. Papafotiou, A. Rantzer, and P. Riedinger, “Comparison of hybrid control techniques for buck and boost DC-DC converters,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 5, pp. 1126–1145, 2010.

[30] S. Bhattacharya, A. Khanafer, and T. Basar, “Switching behavior in optimal communication strategies for team jamming games under resource constraints,” in *IEEE International Conference on Control Applications*, 2011, pp. 1232–1237.

[31] M. Kamgarpour, W. Zhang, and C. J. Tomlin, “Modeling and optimization of terminal airspace and aircraft arrival subject to weather uncertainties,” in *AIAA Guidance, Navigation and Control Conference*, 2011, pp. 6516–6521.

[32] H. Gonzalez, R. Vasudevan, M. Kamgarpour, S. S. Sastry, R. Bajcsy, and C. Tomlin, “A numerical method for the optimal control of switched systems,” in *IEEE Conference on Decision and Control*, 2010, pp. 7519–7526.

[33] K. Deng, Y. Sun, S. Li, Y. Lu, J. Brouwer, P. Mehta, M. Zhou, and A. Chakraborty, “Model predictive control of central chiller plant with thermal energy storage via dynamic programming and mixed-integer linear programming,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 565–579, 2015.

[34] D. Gorges, M. Izak, and S. Liu, “Optimal control and scheduling of switched systems,” *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 135–140, 2011.

[35] Y. Wardi, M. Egerstedt, and M. Hale, “Switched-mode systems: Gradient-descent algorithms with Armijo step sizes,” *Discrete Event Dynamic Systems*, pp. 1–29, 2014.

[36] B. Passenberg, P. E. Caines, M. Sobotka, O. Stursberg, and M. Buss, “The minimum principle for hybrid systems with partitioned state space and unspecified discrete state sequence,” in *IEEE Conference on Decision and Control*, 2010, pp. 6666–6673.

[37] M. Zhang and S. Bhattacharya, “Scheduling and motion planning for autonomous grain carts,” in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3422–3427.

[38] T. M. Caldwell and T. D. Murphey, “Projection-based optimal mode scheduling,” in *IEEE Conference on Decision and Control*, 2013, pp. 5307–5314.

[39] ——, “Projection-based switched system optimization,” in *American Control Conference*, 2012, pp. 4552–4557.

[40] ——, “Projection-based switched system optimization: absolute continuity of the line search,” in *IEEE Conference on Decision and Control*, 2012, pp. 699–706.

[41] Z. Sun and S. S. Ge, “Analysis and synthesis of switched linear control systems,” *Automatica*, vol. 41, no. 2, pp. 181–195, 2005.

[42] W. Zhang and J. Hu, "On optimal quadratic regulation for discrete-time switched linear systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer-Verlag, 2008, vol. 4981, pp. 584–597.

[43] Y. Kouhi, N. Bajcinca, and R. G. Sanfelice, "Suboptimality bounds for linear quadratic problems in hybrid linear systems," in *IEEE European Control Conference*, 2013, pp. 2663–2668.

[44] A. Giua, C. Seatzu, and C. Van Der Mee, "Optimal control of switched autonomous linear systems," in *IEEE Conference on Decision and Control*, vol. 3, 2001, pp. 2472–2477.

[45] X. Xu and P. J. Antsaklis, "Optimal control of switched autonomous systems," in *IEEE Conference on Decision and Control*, vol. 4, 2002, pp. 4401–4406.

[46] T. M. Caldwell and T. D. Murphey, "Single integration optimization of linear time-varying switched systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1592–1597, 2012.

[47] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915–2934, 2006.

[48] M. Fanti, G. Iacobellis, A. Mangini, and W. Ukovich, "Freeway traffic modeling and control in a first-order hybrid Petri net framework," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 90–102, 2014.

[49] Q. Duan, J. Zeng, K. Chakrabarty, and G. Dispoto, "Real-time production scheduler for digital-print-service providers based on a dynamic incremental evolutionary algorithm,"

*IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 701–715, 2015.

[50] R. Vasudevan, H. Gonzalez, R. Bajcsy, and S. S. Sastry, "Consistent approximations for the optimal control of constrained switched systems—Part 1: A conceptual algorithm," *SIAM Journal on Control and Optimization*, vol. 51, no. 6, pp. 4463–4483, 2013.

[51] ——, "Consistent approximations for the optimal control of constrained switched systems—Part 2: An implementable algorithm," *SIAM Journal on Control and Optimization*, vol. 51, no. 6, pp. 4484–4503, 2013.

[52] E. Polak, *Optimization: Algorithms and consistent approximation*. Princeton University Press, 1997.

[53] B. Brogliato and V. Acary, "Numerical methods for nonsmooth dynamical systems," *Lecture Notes in Applied and Computational Mechanics*, 2008.

[54] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[55] H. Jing, Z. Liu, and H. Chen, "A switched control strategy for antilock braking system with on/off valves," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 4, pp. 1470–1484, 2011.

[56] O. Stursberg and S. Engell, "Optimal control of switched continuous systems using mixed-integer programming," in *15th IFAC World Congress of Automatic Control*, vol. 15, no. 1, 2002, pp. 558–558.

[57] S. Garrido, M. Abderrahim, A. Gimenez, R. Diez, and C. Balaguer, "Anti-swinging input shaping control of an automatic construction crane," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 3, pp. 549–557, 2008.

[58] N. Sun, Y. Fang, and H. Chen, "A new antiswing control method for underactuated cranes with unmodeled uncertainties: Theoretical design and hardware experiments," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 453–465, 2015.

[59] P. Cruz, M. Oishi, and R. Fierro, "Lift of a cable-suspended load by a quadrotor: A hybrid system approach," in *American Control Conference*, 2015, pp. 1887–1892.

[60] R. Goebel, R. G. Sanfelice, and A. Teel, "Hybrid dynamical systems," *IEEE Control Systems*, vol. 29, no. 2, pp. 28–93, 2009.

[61] J. Lygeros, "An overview of hybrid systems control," in *Handbook of Networked and Embedded Control Systems*. Springer, 2005, pp. 519–537.

[62] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 110–115, 2006.

[63] H. Gonzalez, R. Vasudevan, M. Kamgarpour, S. S. Sastry, R. Bajcsy, and C. J. Tomlin, "A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems," in *International Conference on Hybrid Systems: Computation and Control*, 2010, pp. 51–60.

[64] T. M. Caldwell and T. D. Murphey, “Single integration optimization of linear time-varying switched systems,” in *American Control Conference*, 2011, pp. 2024–2030.

[65] B. D. Anderson and J. B. Moore, *Optimal control: Linear quadratic methods*. Courier Corporation, 2007.

[66] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest, “Gradient descent approach to optimal mode scheduling in hybrid dynamical systems,” *Journal of Optimization Theory and Applications*, vol. 136, no. 2, pp. 167–186, 2008.

[67] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2009.

[68] P. Bouffard, A. Aswani, and C. Tomlin, “Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results,” in *IEEE International Conference on Robotics and Automation*, 2012, pp. 279–284.

[69] X. Liu and X. Kong, “Nonlinear model predictive control for dfig-based wind power generation,” *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1046–1055, 2014.

[70] L. Magni, G. De Nicolao, and R. Scattolini, “Output feedback and tracking of nonlinear systems with model predictive control,” *Automatica*, vol. 37, no. 10, pp. 1601–1607, 2001.

[71] R. G. Sanfelice, “Input-output-to-state stability tools for hybrid systems and their interconnections,” *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1360–1366, 2014.

[72] M. A. Müller and F. Allgöwer, “Improving performance in model predictive control: Switching cost functionals under average dwell-time,” *Automatica*, vol. 48, no. 2, pp. 402–409, 2012.

[73] J. Hu, J. Shen, and W. Zhang, “A generating function approach to the stability of discrete-time switched linear systems,” in *ACM International Conference on Hybrid Systems: Computation and Control*, 2010, pp. 273–282.

[74] F. A. Fontes, “A general framework to design stabilizing nonlinear model predictive controllers,” *Systems & Control Letters*, vol. 42, no. 2, pp. 127–143, 2001.

[75] R. Meyer, M. Žefran, and R. A. DeCarlo, “A comparison of the embedding method to multi-parametric programming, mixed-integer programming, gradient-descent, and hybrid minimum principle based methods,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1784–1800, 2014.

[76] K. A. Cunefare, S. De Rosa, N. Sadegh, and G. Larson, “State-switched absorber for semi-active structural control,” *Journal of Intelligent Material Systems and Structures*, vol. 11, no. 4, pp. 300–310, 2000.

[77] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, “Symbolic planning and control of robot motion [grand challenges of robotics],” *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[78] T. Wongpiromsarn, U. Topcu, and R. M. Murray, “Receding horizon temporal logic planning for dynamical systems,” in *IEEE Conference on Decision and Control*, 2009, pp. 5997–6004.

[79] P. Martin and M. B. Egerstedt, “Hybrid systems tools for compiling controllers for cyber-physical systems,” *Discrete Event Dynamic Systems*, vol. 22, no. 1, pp. 101–119, 2012.

[80] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, “Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures,” in *ACM/IEEE 41st International Symposium on Computer Architecture*. IEEE, 2014, pp. 97–108.

[81] A. R. Ansari and T. D. Murphey, “Sequential Action Control: Closed-form optimal control for nonlinear and nonsmooth systems,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.

[82] A. Mavrommati, A. R. Ansari, and T. D. Murphey, “Optimal control-on-request: An application in real-time assistive balance control,” in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5928–5934.

[83] A. Ansari, K. Flaßkamp, and T. D. Murphey, “Sequential action control for tracking of free invariant manifolds,” in *Conference on Analysis and Design of Hybrid Systems*, 2015.

[84] E. Tzorakoleftherakis, A. R. Ansari, A. Wilson, J. Schultz, and T. D. Murphey, "Model-based reactive control for hybrid and high-dimensional robotic systems," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 431–438, 2016.

[85] M. Dellnitz, G. Froyland, and O. Junge, "The algorithms behind GAIO—Set oriented numerical methods for dynamical systems," in *Ergodic theory, analysis, and efficient simulation of dynamical systems*. Springer, 2001, pp. 145–174.

[86] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[87] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[88] M. Egerstedt, Y. Wardi, and H. Axelsson, "Optimal control of switching times in hybrid systems," in *IEEE International Conference on Methods and Models in Automation and Robotics*, 2003.

[89] J. Lygeros, K. H. Johansson, S. N. Simić, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[90] D. F. Delchamps, "Stabilizing a linear system with quantized state feedback," *IEEE Transactions on Automatic Control*, vol. 35, no. 8, pp. 916–924, 1990.

[91] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the ARL-monopod II," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 974–986, 2006.

[92] G. E. Jan, C. Luo, L. P. Hung, and S. T. Shih, “A computationally efficient complete area coverage algorithm for intelligent mobile robot navigation,” in *2014 International Joint Conference on Neural Networks*, July 2014, pp. 961–966.

[93] Y. Stergiopoulos and A. Tzes, “Spatially distributed area coverage optimisation in mobile robotic networks with arbitrary convex anisotropic patterns,” *Automatica*, vol. 49, no. 1, pp. 232–237, 2013.

[94] D. E. Soltero, M. Schwager, and D. Rus, “Decentralized path planning for coverage tasks using gradient descent adaptive control,” *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 401–425, 2014.

[95] R. J. Meuth, E. W. Saad, D. C. Wunsch, and J. Vian, “Adaptive task allocation for search area coverage,” in *IEEE International Conference on Technologies for Practical Robot Applications*, 2009, pp. 67–74.

[96] M. Schwager, D. Rus, and J.-J. Slotine, “Decentralized, adaptive coverage control for networked robots,” *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[97] J.-M. Passerieux and D. Van Cappel, “Optimal observer maneuver for bearings-only tracking,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 34, no. 3, pp. 777–788, 1998.

[98] A. N. Bishop and P. N. Pathirana, "Optimal trajectories for homing navigation with bearing measurements," in *Proceedings of the 2008 International Federation of Automatic Control Congress*, 2008.

[99] X. Liao and L. Carin, "Application of the theory of optimal experiments to adaptive electromagnetic-induction sensing of buried targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 961–972, 2004.

[100] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-theoretic coordinated control of multiple sensor platforms," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 1521–1526.

[101] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2015.

[102] R. O'Flaherty and M. Egerstedt, "Optimal exploration in unknown environments," in *IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 5796–5801.

[103] L. M. Miller and T. D. Murphey, "Optimal planning for target localization and coverage using range sensing," in *IEEE International Conference on Automation Science and Engineering*, 2015, pp. 501–508.

[104] "`sphero` robot," http://www.sphero.com/, designed by Sphero.

[105] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

[106] P. A. Plonski, P. Tokekar, and V. Isler, “Energy-efficient path planning for solar-powered mobile robots,” *Journal of Field Robotics*, vol. 30, no. 4, pp. 583–601, 2013.

[107] J. A. Broderick, D. M. Tilbury, and E. M. Atkins, “Optimal coverage trajectories for a UGV with tradeoffs for energy and time,” *Autonomous Robots*, vol. 36, no. 3, pp. 257–271, 2014.

[108] M. Garzón, J. Valente, J. J. Roldán, L. Cancar, A. Barrientos, and J. Del Cerro, “A multirobot system for distributed area coverage and signal searching in large outdoor scenarios,” *Journal of Field Robotics*, vol. 33, no. 8, pp. 1087–1106, 2016.

[109] X. Tan, “Autonomous robotic fish as mobile sensor platforms: Challenges and potential solutions,” *Marine Technology Society Journal*, vol. 45, no. 4, pp. 31–40, 2011.

[110] M. Quigley, M. A. Goodrich, S. Griffiths, A. Eldredge, and R. W. Beard, “Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera,” in *IEEE international conference on robotics and automation*, 2005, pp. 2600–2605.

[111] K. Dogancay, “UAV path planning for passive emitter localization,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 2, 2012.

[112] S. M. Ross, R. G. Cobb, and W. P. Baker, “Stochastic real-time optimal control for bearing-only trajectory planning,” *International Journal of Micro Air Vehicles*, vol. 6, no. 1, pp. 1–27, 2014.

[113] J. Moore and R. Tedrake, “Control synthesis and verification for a perching UAV using LQR-Trees,” in *IEEE Conference on Decision and Control*, 2012, pp. 3707–3714.

[114] C. Luo and S. X. Yang, “A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments,” *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1279–1298, 2008.

[115] D. A. Shell and M. J. Matarić, “Ergodic dynamics for large-scale distributed robot systems,” in *International Conference on Unconventional Computation*. Springer, 2006, pp. 254–266.

[116] Z. Tang and U. Ozguner, “Motion planning for multitarget surveillance with mobile sensor agents,” *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, 2005.

[117] V. P. Jilkov and X. R. Li, “On fusion of multiple objectives for UAV search & track path optimization.” *Journal of Advances in Information Fusion*, vol. 4, no. 1, pp. 27–39, 2009.

[118] R. R. Pitre, X. R. Li, and R. Delbalzo, “UAV route planning for joint search and track missions An information-value approach,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2551–2565, 2012.

[119] C. Y. Wong, G. Seet, and S. K. Sim, “Multiple-robot systems for usar: key design attributes and deployment issues,” *International Journal of Advanced Robotic Systems*, vol. 8, no. 1, pp. 85–101, 2011.

[120] Y. Liu and G. Nejat, “Robotic urban search and rescue: A survey from the control perspective,” *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 147–165, 2013.

[121] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, “Aerial remote sensing in agriculture: A practical approach to area coverage and path

planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[122] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[123] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[124] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "BA*: An online complete coverage algorithm for cleaning robots," *Applied intelligence*, vol. 39, no. 2, pp. 217–235, 2013.

[125] N. K. Yilmaz, C. Evangelinos, P. F. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

[126] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2013.

[127] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *The International journal of robotics research*, vol. 22, no. 7-8, pp. 441–466, 2003.

[128] C. W. Bac, E. J. Henten, J. Hemming, and Y. Edan, “Harvesting robots for high-value crops: State-of-the-art review and challenges ahead,” *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, 2014.

[129] S. L. Smith, M. Schwager, and D. Rus, “Persistent robotic tasks: Monitoring and sweeping in changing environments,” *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[130] Y. Cao, W. Yu, W. Ren, and G. Chen, “An overview of recent progress in the study of distributed multi-agent coordination,” *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[131] W. Ren, R. W. Beard, and T. W. McLain, “Coordination variables and consensus building in multiple vehicle systems,” in *Lecture Notes in Control and Information Sciences*. New York: Springer-Verlag, 2004, vol. 309, pp. 171–188.

[132] J. Cortes, S. Martinez, and F. Bullo, “Spatially-distributed coverage optimization and control with limited-range interactions,” *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 4, pp. 691–719, 2005.

[133] Y. Kantaros, M. Thanou, and A. Tzes, “Distributed coverage control for concave areas by a heterogeneous Robot–Swarm with visibility sensing constraints,” *Automatica*, vol. 53, pp. 195–207, 2015.

[134] S. Rutishauser, N. Correll, and A. Martinoli, “Collaborative coverage using a swarm of networked miniature robots,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 517–525, 2009.

[135] A. Kwok and S. Martínez, “A distributed deterministic annealing algorithm for limited-range sensor coverage,” *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 792–804, 2011.

[136] M. Zhu and S. Martínez, “Distributed coverage games for energy-aware mobile sensor networks,” *SIAM Journal on Control and Optimization*, vol. 51, no. 1, pp. 1–27, 2013.

[137] A. Y. Yazicioglu, M. Egerstedt, and J. S. Shamma, “Communication-free distributed coverage for networked systems,” *IEEE Transactions on Control of Network Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[138] G. Mathew and I. Mezić, “Metrics for ergodicity and design of ergodic dynamics for multi-agent systems,” *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[139] D. E. Kirk, *Optimal control theory: An introduction*. Courier Corporation, 2012.

[140] L. M. Miller and T. D. Murphey, “Trajectory optimization for continuous ergodic exploration,” in *American Control Conference*, 2013, pp. 4196–4201.

[141] G. D. L. Torre, K. Flaßkamp, A. Prabhakar, and T. D. Murphey, “Ergodic exploration with stochastic sensor dynamics,” in *American Control Conference*, 2016, pp. 2971–2976.

[142] D. Liberzon, *Calculus of variations and optimal control theory: A concise introduction*. Princeton University Press, 2012.

[143] T. M. Caldwell and T. D. Murphey, "Projection-based iterative mode scheduling for switched systems," *Nonlinear Analysis: Hybrid Systems*, vol. 21, pp. 59–83, 2016.

[144] S. L. de Oliveira Kothare and M. Morari, "Contractive model predictive control for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 6, pp. 1053–1071, 2000.

[145] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems*, vol. 22, no. 1, pp. 44–52, 2002.

[146] F. Xie and R. Fierro, "First-state contractive model predictive control of nonholonomic mobile robots," in *American Control Conference*, 2008, pp. 3494–3499.

[147] G. Ferrari-Trecate, L. Galbusera, M. P. E. Marciandi, and R. Scattolini, "Model predictive control schemes for consensus in multi-agent systems with single-and double-integrator dynamics," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2560–2572, 2009.

[148] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2011.

[149] H. Michalska and R. Vinter, "Nonlinear stabilization using discontinuous moving-horizon control," *IMA Journal of Mathematical Control and Information*, vol. 11, no. 4, pp. 321–340, 1994.

[150] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, "Modeling the quad-rotor mini-rotorcraft," in *Quad Rotorcraft Control*. Springer, 2013, pp. 23–34.

[151] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[152] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, 2011.

[153] C. Leung, S. Huang, G. Dissanayake, and T. Furukawa, "Trajectory planning for multiple robots in bearing-only target localisation," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3978–3983.

[154] J. P. Helferty and D. R. Mudgett, "Optimal observer trajectories for bearings only tracking by minimizing the trace of the cramer-rao lower bound," in *IEEE Conference on Decision and Control*, 1993, pp. 936–939.

[155] N. Cao, K. H. Low, and J. M. Dolan, "Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 7–14.

[156] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[157] H. Yu, K. Meier, M. Argyle, and R. W. Beard, “Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 541–552, 2015.

[158] S. Kim, H. Oh, and A. Tsourdos, “Nonlinear model predictive coordinated standoff tracking of a moving ground vehicle,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 557–566, 2013.

[159] R. Anderson and D. Milutinović, “A stochastic approach to Dubins feedback control for target tracking,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3917–3922.

[160] J. R. Spletzer and C. J. Taylor, “Dynamic sensor planning and control for optimally tracking targets,” *The International Journal of Robotics Research*, vol. 22, no. 1, pp. 7–20, 2003.

[161] S. Zhu, D. Wang, and C. B. Low, “Ground target tracking using uav with input constraints,” *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 417–429, 2013.

[162] U. Zengin and A. Dogan, “Real-time target tracking for autonomous UAVs in adversarial environments: A gradient search algorithm,” *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 294–307, 2007.

[163] P. Yao, H. Wang, and Z. Su, “Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment,” *Aerospace Science and Technology*, vol. 47, pp. 269–279, 2015.

[164] P. Tokekar, J. Vander Hook, and V. Isler, "Active target localization for bearing based robotic telemetry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 488–493.

[165] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent and Robotic Systems*, vol. 47, no. 4, pp. 361–382, 2006.

[166] L. Ma and N. Hovakimyan, "Cooperative target tracking in balanced circular formation: Multiple UAVs tracking a ground vehicle," in *American Control Conference*, 2013, pp. 5386–5391.

[167] G. Gu, P. Chandler, C. Schumacher, A. Sparks, and M. Pachter, "Optimal cooperative sensing using a team of UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 4, pp. 1446–1458, 2006.

[168] S. A. Quintero, M. Ludkovski, and J. P. Hespanha, "Stochastic optimal coordination of small UAVs for target tracking using regression-based dynamic programming," *Journal of Intelligent & Robotic Systems*, vol. 82, no. 1, pp. 135–162, 2016.

[169] A. Emery and A. V. Nenarokomov, "Optimal experiment design," *Measurement Science and Technology*, vol. 9, no. 6, p. 864, 1998.

[170] B. R. Frieden, *Science from Fisher information: A unification*. Cambridge University Press, 2004.

[171] D. Ucinski, *Optimal measurement methods for distributed parameter system identification.* CRC Press, 2004.

[172] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[173] S. J. Julier and J. K. Uhlmann, “New extension of the Kalman filter to nonlinear systems,” in *AeroSense’97.* International Society for Optics and Photonics, 1997, pp. 182–193.

[174] E. S. Jones and S. Soatto, “Visual-inertial navigation, mapping and localization: A scalable real-time causal approach,” *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.

[175] V. J. Aidala, “Kalman filter behavior in bearings-only tracking applications,” *IEEE Transactions on Aerospace and Electronic Systems*, no. 1, pp. 29–39, 1979.

[176] T. Lee and S. Soatto, “Learning and matching multiscale template descriptors for real-time detection, localization and tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1457–1464.

[177] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[178] R. W. O’Flaherty, “A control theoretic perspective on learning in robotics,” 2015.

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

[180] I. Abraham, A. Prabhakar, M. J. Z. Hartmann, and T. D. Murphey, “Ergodic exploration using binary sensing for nonparametric shape estimation,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 827–834, 2017.

[181] T. F. Cootes and C. J. Taylor, “A mixture model for representing shape variation,” *Image and Vision Computing*, vol. 17, no. 8, pp. 567–573, 1999.

[182] M. Rousson and D. Cremers, “Efficient kernel density estimation of shape and intensity priors for level set segmentation,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2005, pp. 757–764.

[183] J. A. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[184] S. Chen, C. F. Cowan, and P. M. Grant, “Orthogonal least squares learning algorithm for radial basis function networks,” *IEEE Transactions on neural networks*, vol. 2, no. 2, pp. 302–309, 1991.

[185] J. Platt *et al.*, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[186] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, “Pose estimation for planar contact manipulation with manifold particle filters,” *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 922–945, 2015.

[187] J. Borenstein and L. Feng, “Measurement and correction of systematic odometry errors in mobile robots,” *IEEE Transactions on robotics and automation*, vol. 12, no. 6, pp. 869–880, 1996.

[188] S. Se, D. Lowe, and J. Little, “Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks,” *The international Journal of robotics Research*, vol. 21, no. 8, pp. 735–758, 2002.

[189] A. Eliazar and R. Parr, “DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks,” in *IJCAI*, vol. 3, 2003, pp. 1135–1142.