OPTIMAL, MULTI-MODAL CONTROL WITH APPLICATIONS IN ROBOTICS

A Dissertation
Presented to
The Academic Faculty

By

Tejas R. Mehta

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2007

OPTIMAL, MULTI-MODAL CONTROL WITH APPLICATIONS IN ROBOTICS

Approved by:

Dr. Magnus Egerstedt, Advisor
Associate Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Spyros Reveliotis
Associate Professor, School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Anthony Yezzi
Associate Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Chin-Hui Lee
Professor, School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Bonnie H. Ferri
Professor, Graduate Affairs Associate Chair, School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 3rd, 2007

This work is dedicated to ...

... my parents, Rajendra and Nila,

for dedicating their life to me and

always supporting me in all my endeavors ...

... and to my love, Maya,

for being my constant source of

inspiration and encouragement.

ACKNOWLEDGMENTS

This thesis is a culmination of my academic career at the Georgia Institute of Technology. I have been fortunate to have excellent teachers and mentors throughout my entire education, and to all of them I owe my deepest gratitude.

First and foremost, I would like to thank my advisor, Professor Magnus Egerstedt, for his support, guidance, encouragement, and enthusiasm. He introduced me to many interesting research areas while always giving me the freedom to pursue my interests. I would also like to thank all my professors and teachers along the way, for they have all had a positive influence on me. I would like to thank Professors Bonnie H. Ferri, Chin-Hui Lee, Spyros Reveliotis, and Anthony Yezzi for serving on my defense committee, and for their constructive comments and suggestions.

I would also like to thank all of my fellow students I have had the pleasure to interact with over the years. Graduate school provided me with a unique opportunity to meet and work with an incredibly diverse group of people. I have enjoyed my time with them and gained from their varying perspective on all sort of matters from work to life. At the risk of leaving out a few names, I would like to thank Henrik Axelsson, Shun-Ichi Azuma, Mohamed Babaali, Mauro Boccadoro, Florent Delmotte, Dennis Ding, Eric Innis, Meng Ji, Patrick Martin, Matt Powers, Brian Smith, Ganesh Sundaramoorthi, Dave Wooden, Deryck Yeung, ...

Having a great work atmosphere is important, but this must be complemented with an enjoyable life away work. I have to thank my friends and family for giving me a great escape from research. There is no way for me to list everyone here, but I would particularly like to thank Vihang Desai, Priya Mehta, Shohel Mollah, Anup Patel, Minal Patel, Raj Rao, ...

No acknowledgement can be complete without thanking my parents, Rajendra and Nila. They were my first teachers in life and provided me with endless support,

guidance, and advice throughout my life. They have always put my happiness, needs, and desires before their own, and I cannot overstate my gratitude to them for the way they raised me.

Finally, it goes without saying that I would not be where I am without my better half, my love, Maya. She is my constant source of inspiration, support, and encouragement. Thank you for everything, sweetheart!

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

SUMMARY xii

CHAPTER 1 INTRODUCTION AND BACKGROUND 1
1.1 Introduction 1
1.2 Background 3
1.2.1 Motion Description Languages 5
1.2.2 Optimal Control of Hybrid Systems 7
1.2.3 Mobile Robot Navigation 9

CHAPTER 2 MULTI-MODAL CONTROL PROGRAMS 12
2.1 Reinforcement Learning 14
2.1.1 Standard Reinforcement Learning 15
2.1.2 Learning Control Programs for Discrete-Time Systems 16
2.1.3 Maze Example 17
2.2 Learning Control Programs for Continuous-Time Systems 19
2.2.1 Maze Revisited 25
2.3 Refining the Learning Process 26
2.4 Robustness Analysis 28
2.5 Conclusions 29

CHAPTER 3 ADAPTIVE MULTI-MODAL CONTROL 31
3.1 Continuous-Time LTI Systems 32
3.2 General Framework for Adaptive Multi-Modal Control 34
3.2.1 Numerics 38
3.3 Examples 39
3.3.1 LTI Example 40
3.3.2 Robotics Example 42
3.4 Conclusions 44

CHAPTER 4 A UNIFIED VARIATIONAL FRAMEWORK 46
4.1 Time-Driven Interrupts 49
4.2 Event-Driven Interrupts 53
4.3 Applications 60
4.3.1 Mobile Robot Navigation 60
4.3.2 Optimal Membership Functions 64
4.4 Conclusions 68

CHAPTER 5 LEARNING FROM EXAMPLE 69
5.1 Problem Formulation 71
5.2 Variational Approach 74
5.2.1 Numerical Algorithms 75
5.2.2 Example 78
5.2.3 Navigation Using the Magellan Pro 81
5.3 Learning Applied to Ground Robots (LAGR) Project 84
5.3.1 LAGR Control Architecture 86
5.3.2 Learning Behaviors From Example 88
5.3.3 Experimental Results 92
5.4 Conclusions 93

CHAPTER 6 MULTI-MODAL, MULTI-DIMENSIONAL SYSTEMS 95
6.1 A Motivating Example 96
6.2 Optimal Control Framework 101
6.2.1 Problem Formulation 101
6.2.2 Optimality Conditions 103
6.2.3 Numerical Algorithms 107
6.3 Optimal Control of an Ice Skater 110
6.4 Conclusions 112

CHAPTER 7 CONCLUSIONS AND EXTENSIONS 113
7.1 Conclusions 113
7.2 Extensions 115

REFERENCES 118

LIST OF TABLES

Table 1 An algorithm for simultaneously exploring the state space and learning the optimal control program. 22

Table 2 A gradient descent algorithm. 39

Table 3 Cost comparison between the optimal mode string generated using the original mode set (Σ), the augmented mode set using approach outlined in Chapter 3 (Σ_{n1}), and the augmented mode set using the methods outlined in this chapter (Σ_{n2}). 63

Table 4 The inner algorithm for Learning from Example. 76

Table 5 The outer algorithm for Learning from Example. 77

Table 6 A hill climbing algorithm. 91

Table 7 A descent algorithm for M^3D systems. 108

Table 8 An algorithm to update u during each iteration. 109

LIST OF FIGURES

Figure 1 An example of a robot navigating using a behavior-based approach. 10

Figure 2 Depicted is the progression from X and U being smooth manifolds (a) to the case when both the state space and the input set are finite (c) through the introduction of multi-modal control procedures and Lebesque sampling. 13

Figure 3 Robot navigating through a maze using a standard reinforcement-learning model (left) and using modes with interrupts as the control set (right). 18

Figure 4 Simulation results demonstrating the learning of optimal mode strings in a continuous-time system. 25

Figure 5 (a) The experimental setup of the maze. (b) The path of the robot together with the range sensor readings (IR-based) obtained throughout the final run. Note how the odometric drift makes the maze look somewhat distorted. 26

Figure 6 Depicted is the original trajectory $x(t)$ and the approximation trajectory $z(t)$. 35

Figure 7 Comparison of the CBH and COV methods for (a) $T = 2$, (b) $T = 1$. 41

Figure 8 The evolution of $\vec{\alpha}$ for (a) T = 2 and (b) T = 1. 41

Figure 9 The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the (a) original set of modes, (b) augmented set of modes. 44

Figure 10 Depicted is the original trajectory $x(t)$ and the approximation trajectory $z(t)$. 48

Figure 11 The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the original set of modes. . . . 61

Figure 12 The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the augmented set of modes using method outlined in Chapter 3. 62

Figure 13 The optimal path to drive a unicycle from x_0 to x_g using the augmented set of modes using method outlined in Chapter 3 (dashes) and the method outlined in this chapter (thick). 63

Figure 14 Architecture of a Takagi and Sugeno fuzzy controller. 64

Figure 15 The standard piecewise linear or triangular (solid) and the differentiable exponential (dashed) membership functions. 66

Figure 16 Optimization results: the trajectory for the initial guess of α (dashed) along with the final optimal trajectory. 67

Figure 17 (a) The experimental setup, (b) The resulting trajectories plotted using the odometry readings, while the obstacle are inferred from the sensor readings. 68

Figure 18 The observed trajectory from a training run of a unicycle navigating from x_0 to x_g. 79

Figure 19 The optimal cost as a function of the number of modes given by the outer algorithm. 80

Figure 20 Depicted is the approximation trajectory ($\tilde{x}$) obtained by using three modes (dashed) and two modes (dotted) along with the original observed trajectory. 80

Figure 21 (a) The experimental setup, (b) the observed trajectory from a training run from x_0 to x_g. 81

Figure 22 Depicted is the approximation trajectory ($\tilde{x}$) obtained by using the learned behavior (dashed) along with the original observed trajectory. 82

Figure 23 Depicted is the cost J as a function of the control parameters α_g and α_o. The cost surface is color scaled from low cost (blue) to high cost (red). 83

Figure 24 The LAGR Robot. 84

Figure 25 Standard Hybrid Control System Block Diagram. 86

Figure 26 A graphical representation of the voting scheme employed to navigate the robot. The x-axis of each plot represents an ego-centric angular distribution of possible paths around the robot from $-\pi$ to $+\pi$, with 0 being in front of the robot. The y-axis represents the relative preference of each path, according to the respective controller. Vetoes are drawn as large negative values. The last plot represents the sum of the votes provided by all the controllers. The largest non-vetoed value is chosen for action by the robot. 87

Figure 27 Sample images from the test run navigating using the learned behaviors. 93

Figure 28 Skating trajectories using the proposed M^3D model. 97

Figure 29 Depicted is the force applied during the SL mode 98

Figure 30 State Transition 100

Figure 31 Depicted here is a situation where the standard update method leads to a conflict in dimensions of the control u_i. 108

Figure 32 (a) The evolution of the cost as a function of the iteration, (b) the active mode as a function of time for the optimal switching times. . 111

Figure 33 Depicted is the optimal trajectory staring in SR & GL mode and switching between the GL, SL & GR, GR modes. 112

SUMMARY

The objective of this dissertation is to incorporate the concept of optimality to multi-modal control and apply the theoretical results to obtain successful navigation strategies for autonomous mobile robots. The main idea in multi-modal control is to breakup a complex control task into simpler tasks. In particular, number of control modes are constructed, each with respect to a particular task, and these modes are combined according to some supervisory control logic in order to complete the overall control task. This way of modularizing the control task lends itself particularly well to the control of autonomous mobile robot, as evidenced by the success of behavior-based robotics. Many challenging and interesting research issues arise when employing multi-modal control. This thesis aims to address these issues within an optimal control framework.

To this end, the contributions of this dissertation are as follows: We first addressed the problem of inferring global behaviors from a collection of local rules (i.e., feedback control laws). Given a collection of modes, an algorithm, that characterizes the expressiveness of the multi-modal system and learns control programs that complete a desired task while minimizing a prescribed performance criterion, is presented. Next, we addressed the issue of adaptively varying the multi-modal control system to further improve performance. A variational framework for adaptive multi-modal control is developed, where a given collection of modes is adaptively improved by adding new modes to the set. This augmentation of the mode set increases the expressiveness and the performance of the system as well as reduces the complexity of the control programs.

Adaptive multi-modal control led to an interesting application to the the Learning From Example problem, where new controllers are learned from training examples.

First, a variational framework is used to learn new modes as needed to approximate a given training trajectories. Next, this framework was applied to the DARPA sponsored Learning Applied to Ground Robots (LAGR) project. The LAGR project motivated a need for new solutions not relying on differentiability assumptions (as the variational approach does), which was addressed by posing the learning problem as an combinatorial optimization problem, and an algorithm for solving this problem using a hill climbing method is presented.

Next, we addressed the optimal control of multi-modal systems with infinite dimensional constraints. These constraints are formulated as multi-modal, multi-dimensional (M^3D) systems, where the dimensions of the state and control spaces change between modes to account for the constraints, to ease the computational burdens associated with traditional methods. The optimality conditions for this formulation are derived and an algorithmic framework for the optimal control of M^3D systems is presented.

Finally, we used multi-modal control strategies to develop effective navigation strategies for autonomous mobile robots. The theoretical results presented in this thesis are verified by conducting simulated experiments using Matlab and actual experiments in a lab setting using the Magellan Pro mobile robot platform. Moreover, human operated training runs are used to develop effective navigation strategies following the constructivist framework for the learning from example problem. These results were successfully verified on the LAGR robot to learn effective strategies for the LAGR competition.

In closing, the main strength of multi-modal control lies in breaking up complex control task into simpler tasks. This divide-and-conquer approach helps modularize the control system. This has the same effect on complex control systems that object-oriented programming has for large-scale computer programs, namely it allows greater simplicity, flexibility, and adaptability.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

To manage the complexity associated with many modern control applications, multi-modal control has emerged as a viable approach in which a number of control modes are constructed and combined according to some supervisory control logic, e.g., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. The idea is that the design of each individual mode, designed with respect to a particular control task, data source, operating point, or system configuration, constitutes a significantly more manageable task than the design of one single, multi-objective control law. Successful examples of this divide-and-conquer approach include flight mode control in avionics [13] and the behavior-based control of autonomous robots [14, 15]. In addition to simplifying the design process, the multi-modal approach also makes it possible to add new functionality to the system without significant increase in complexity.

Two major design tasks are involved when employing multi-modal control. The first task pertains to the design of the individual modes, which consist of a feedback control law and may also include a condition for its termination. The second task concerns concatenating these modes in order to achieve the desired objective. Given that such a mode string is the design objective, the control task thus involves mapping symbols (tokenized mode descriptions) to signals rather than signals (continuous control values) to signals as done in classic control theory. For example, a sample control string for commanding a robot to fetch a ball can be "find the ball," "grab the ball," "bring the ball back" (as opposed to specifying a control value at each time instant). A number of modelling paradigms facilitating this construction have been proposed, including Hybrid Automata [16, 17], Maneuver Automata [5, 6], Control Quanta [7, 8], and Motion Description Languages [3, 12, 18].

The overall objective of this thesis is to incorporate the concept of optimality to both of the aforementioned tasks in multi-modal control and apply the results to robotics applications. To this end, the main topics addressed in this thesis are

- *From local rules to global behaviors:* Given a collection of modes, what can be said about the global behavior of the multi-modal system in terms of expressiveness and task completion? Additionally if task completion is possible, can we optimize some performance criterion while ensuring task completion?

- *Adaptive multi-modal control:* Given a collection of modes, can this collection be adaptively varied in order to achieve better overall performance?

- *Learning from example:* Can we use multi-modal control concepts to learn effective control strategies guided by training examples?

- *Multi-modal, multi-dimensional (M^3D) systems:* Given a multi-modal system with infinite dimensional state constraints, can we simplify the analysis/control of such systems by adopting a non-standard model in which the dimensions of the state and input spaces change between different modes to account for the constraints?

- *Robotics applications:* Can we use the multi-modal control concepts developed above to generate effective navigation strategies for autonomous mobile robots operating in an unstructured environment?

The organization of the thesis is as follows: Section 1.2 introduces some background and a brief review of work pertaining to our research. Chapter 2 addresses the question of inferring global behaviors from local rules. In particular, given a collection of modes, an algorithm that estimates the reachable set is presented, thus characterizing the expressiveness of this mode set. While estimating the reachable set, the algorithm simultaneously learns control programs that complete the desired

task while optimizing a specified performance criterion. Chapter 3 introduces the motivation for adaptive multi-modal control. It is shown how systematically adding new modes can improve the overall performance of the multi-modal system, and a general framework for adaptive multi-modal control is introduced. Chapter 4 continues the development of Chapter 3 and presents a unified variational framework for adaptive multi-modal control. Chapter 5 presents an application of adaptive multi-modal concepts to the Learning From Example problem, where effective control strategies are learned from training examples. The training examples can be generated through human operation or inspired by biological systems. Chapter 6 addresses the problem of constrained hybrid systems, where different modes experience different state constraints. These systems are modelled in a very unique manner as M^3D systems to account for the infinite dimensional constraints, and an algorithmic framework for optimal control of such systems is presented. The multi-modal control concepts developed throughout the thesis are applied to the problem of mobile robot navigation, and examples are presented in various chapters when suitable. Finally, summary of contributions and extensions are discussed in Chapter 7.

1.2 Background

A *multi-modal* system is a system whose dynamical model switches among finitely many possibilities. These switches can be in response to an occurrence of a specific event or a controlled decision. As such they belong to the wider class of *hybrid systems*. Hybrid systems are dynamic systems whose behavior is governed by interacting continuous and discrete dynamics. The continuous-time dynamics of these systems are generally given by differential equations corresponding to the physical properties of the system. The discrete-event dynamics generally correspond to some switching between states or modes and can be supervised by switches, digital circuitry, and/or computer software. Such systems arise when dealing with continuous systems with

phased operations (e.g., diodes), continuous systems controlled by discrete inputs (e.g., switches), complex systems involving the coordination of different processes (e.g., flight control, mobile robots), etc. The origins of hybrid systems can be traced back to Witsenhausen in 1966 [19], although earlier variations, including bang-bang control, sliding mode control, and digital control, date further back.

The increasing use of computer-aided control and embedded control systems has made hybrid systems more common. Correspondingly, this has led to growth in research related to hybrid systems. The disciplines involved in this research effort are control theory, computer science, mathematics, mechanics, and others. The research directions spawned from this growth include modelling and simulation [4, 18, 20, 21], analysis and verification [16, 22, 23, 24, 25], and control of hybrid systems [9, 13, 26, 27, 28, 29]. Note that this list of references is merely a small sample and by no means represents an exhaustive list. Since our main interest lies in the optimal control of multi-modal systems and robotics applications, we will devote more attention to these topics. Readers interested in the other areas of research are encouraged to follow the references given above.

To successfully study hybrid systems, we have to select a proper framework for dealing with them. A number of modelling paradigms facilitating the control and analysis of hybrid systems have been proposed over the years, including the hybrid automata approach ([17, 21, 22]), the systems approach ([20, 24]), and the language-based approach ([3, 5, 7, 18, 30]). Throughout our development, we will follow the *Motion Description Language* (MDL) framework, which was first introduced by Brockett in 1988 and later extended by Manikonda *et. al.* This is discussed more thoroughly in the next section, followed by sections on the optimal control of hybrid systems and behavior-based robotics.

1.2.1 Motion Description Languages

Given a finite set, or *alphabet*, $\mathcal{A}$, by $\mathcal{A}^\star$ we understand the set of all strings of finite length over $\mathcal{A}$. There exist a binary operation on this set corresponding to the concatenation of string, denoted by $a_1 \cdot a_2$ (i.e., if $a_1, a_2 \in \mathcal{A}^\star$, then $a_1 \cdot a_2 \in \mathcal{A}^\star$). Relative to this operation, $\mathcal{A}^\star$ is a semigroup, and if we include the empty string in $\mathcal{A}^\star$, it becomes a monoid (i.e., a semigroup with an identity). A *formal language* is a subset of a free monoid over a finite alphabet.

A *motion alphabet* is a set, possibly infinite, of symbols representing different control actions that define segments of motion. This notion of using language primitives for motion control was first introduced by Brockett in an attempt to formalize the computer control of movement [3], referred to as *Motion Description Language* (MDL). An MDL is given by a set of symbolic strings that represent idealized motions; thus, an MDL is a subset of a free monoid over a given motion alphabet.

In particular, an MDL device (i.e., a computer-controlled mechanism employing the MDL framework) is transmitted a triple $(u(\cdot), \kappa(\cdot), T)$, where $u(\cdot)$ is the open loop control law, $\kappa(\cdot)$ is the closed loop control law, and T defines the epoch over which this control pair is to be used. These triples are referred to as *modal segments*, as they define the mode of control over a segment of time. Suppose that the MDL device under consideration can be described as

$$\dot{x}(t) = f(x(t)) + G(x(t))(u(t) + \kappa(y(t))).$$

Then upon receiving the input string $(u_1, \kappa_1, T_1)(u_2, \kappa_2, T_2) \cdots (u_r, \kappa_r, T_r)$, the MDL device executes a motion that closely approximates the trajectory of x defined by

$$\begin{aligned}
\dot{x} &= f(x) + G(x)(u_1 + \kappa_1(y)); \quad && 0 \le t < T_1 \\
\dot{x} &= f(x) + G(x)(u_2 + \kappa_2(y)); \quad && T_1 \le t < T_1 + T_2 \\
&\vdots && \vdots \\
\dot{x} &= f(x) + G(x)(u_r + \kappa_r(y)); \quad && \sum_{i=1}^{r-1} T_i \le t < \sum_{i=1}^{r} T_i.
\end{aligned}$$

To implement such a system that interprets a family of modal segments, it is necessary to index the modes in a finite way. It is shown that this restriction to a finite family of affine modal segments does not limit the expressiveness of the mechanism. In other words, these modal segments can be used to approximate any state trajectory produced by continuous control signals with arbitrary precision. It should be noted, however, that the limiting case of ϵ-precision may require that the cycle period T goes to zero and the length of the mode sequence goes to infinity. This demonstrates the classic trade-off encountered when dealing with symbol-driven control, namely, the trade-off between the *complexity* of the mode string and the *expressiveness* of the system.

We have already defined expressiveness informally as the ability of a symbol-driven system to generate a desired trajectory with adequate robustness. The complexity of a mode string $\bar{\sigma}$ (note here that σ can be a particular modal segment described above) can be characterized by the number of bits needed to encode it. In this case, the complexity of mode string $\bar{\sigma}$, whose elements are drawn from a mode set Σ, is given by $|\bar{\sigma}| \log_2(card(\Sigma))$, where $|\cdot|$ is the length of the mode string and $card(\Sigma)$ is the cardinality of the mode set. For more information regarding complexity, see [1, 2], where Egerstedt shows that the use of feedback in the MDL framework can reduce the specification complexity of control programs. In particular, the analysis of the reduction in complexity is done using an automaton model.

Motivated by behavior-based robotics and the need for event triggered switches between modes, Brockett's original framework was modified by Manikonda *et. al.* to produce an extended version known as MDLe (extended MDL). In this context, a mode is specified as a triple $\sigma = (\kappa(\cdot), \xi(\cdot), T)$, where $\kappa(\cdot)$ is the feedback control law (includes both the open and closed loop controls), $\xi(\cdot)$ is the interrupt function, and T is the time over which the mode is active. In particular, ξ is a mapping from the observations of the system to the set $\{0, 1\}$; we say that an interrupt is triggered

when a particular observation is mapped to a 1. In this construction, a sequence of modes is operated on as earlier, but the transition to the next mode in the sequence can be driven by either an event (triggering of the interrupt ξ) or time (T seconds elapse). For more information regarding this framework, see [12, 18, 30].

In concluding this section, we note that issues regarding expressiveness and complexity are naturally encountered in our work. In particular, reducing the complexity of the mode string serves as the performance index for the learning algorithm, while increasing expressiveness (thus possibly increasing performance) is the main motivation for adaptive multi-modal control.

1.2.2 Optimal Control of Hybrid Systems

The concept of optimality regarding hybrid systems was discussed along with the introduction of these systems in the visionary work of Witsenhausen [19]. He formulated the problem of finding a continuous control that minimizes a cost function while satisfying the specified terminal conditions and provided the necessary conditions for optimality. There has been a plethora of work pertaining to the optimal control of hybrid systems since then. We cannot possibly provide an exhaustive survey here, but rather, we present some well known references covering a wide variety of optimal control problems.

These problems vary depending on the model of the system and, consequently, the optimization parameters. The system can be modelled with either linear or nonlinear dynamics, with a continuous control variable or no control variable (i.e., autonomous), with autonomous switching (i.e., switch in the discrete dynamics in response to some uncontrolled event) or controlled switching, fixed-schedule (i.e., the switching sequence or the sequence in which different process are visited is assumed fixed) or variable-schedule, etc. Naturally, the optimization parameters change according to the choice of the particular model. For instance, a variable-schedule model would include the sequencing variable as a parameter to optimize over, while a fixed-schedule

model would not include such a parameter. These references also vary in their approach to deriving the optimal control. Of the many viable approaches, we identify two that are pertinent to this thesis. The first approach, referred to as the variational approach, involves using Hamilton-Jacobi theory to access the necessary conditions for optimality. The second approach, referred to as the learning-based approach, uses dynamic programming principles to attain an optimal solution. Selected references are presented next.

Good comprehensive references for modelling and optimal control of hybrid systems include [4, 20]. In particular, Brockett introduces four different models for hybrid systems that combine differential equations and discrete phenomenon and discusses the control aspects of these systems. Similary, Branicky *et. al.* also cites four different models arising in real-world models. Branicky *et. al.*, moreover, proposes a unifying model for hybrid systems and develops hybrid controllers for hybrid systems in an optimal control framework.

In the case of autonomous systems, the control variable often includes the switching time vector (assuming a time-driven system, i.e. when the switches are controlled time-instants) [27, 31, 32, 33, 34, 35, 36, 37, 38]. In particular, Guia *et. al.* present a solution to the switching-time problem for continuous-time linear dynamics, while Egerstedt *et. al.* assume general nonlinear dynamics for autonomous systems. Xu and Antsaklis also assume general dynamics, but their control consists of a continuous input as well the switching times. These references assume a fixed-schedule, thus the control only involves the switching instants (and perhaps a continuous control variable). The following references pertain to the more general optimal control problem, where the control includes the switching instants and the sequencing variable. In particular, Bemporad *et. al.* present a solution for switched linear systems using mixed-integer programming. Shaikh and Caines present a set of necessary conditions for optimality and using these condition propose a class of general hybrid maximum

principle. Axelsson *et. al.* attack the scheduling problem by inserting new modes into a fixed-scheduled system and evaluate the benefit of this insertion using variational methods. The above references regard the optimization of time-driven switched systems, while Boccadoro considers the case of optimizing an event-driven switched system, where the switches are triggered by surfaces [39].

While most of the above references use Hamilton-Jacobi theory to derive the necessary conditions for optimality by differentiating the cost function, the following references consider learning-based approaches for optimizing hybrid systems [11, 26, 40, 41, 42]. In particular, Bradtke considers the problem of Linear Quadratic Regulation (LQR) using policy iteration. Hedlund and Rantzer approach the control of hybrid systems by discretizing the Bellman equation and optimizing the discrete version. In contrast to the discretization approach, Crawford and Morgansen employ learning methods by using a functional-approximation approach to approximate the value function.

1.2.3 Mobile Robot Navigation

In the literature on robot navigation, two distinctly different approaches have emerged. The first approach, which we denote as the *reactive* approach (following the terminology of Arkin in [14]), consists of designing a collection of behaviors, or modes of operations, such as "avoid-obstacle" or "approach-goal." These different behaviors are defined through a particular control law, dedicated to performing a specific task, and the robot switches between different behaviors as obstacles, landmarks, etc. are encountered in the environment, see for example Figure 1. This way of structuring the navigation system has the major advantage that it simplifies the design task. Each controller is designed with only a limited set of objectives under consideration and no elaborate world maps are needed (see for example [14, 15, 43, 44]). Unfortunately, very little can be said analytically about such systems. Another common criticism of this approach is that behaviors are typically functions of only immediate sensor

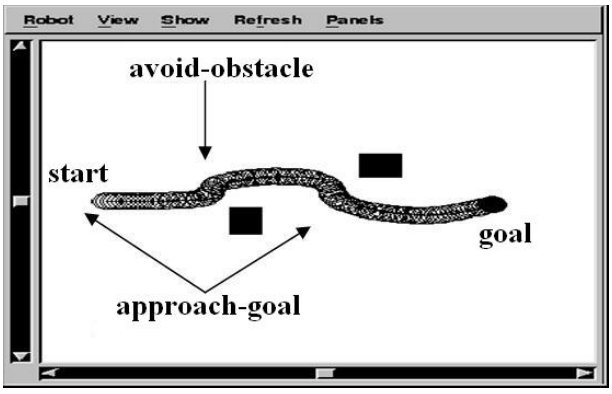


Figure 1. An example of a robot navigating using a behavior-based approach.

information (i.e., without memory), as such they may become stuck in a local minima (i.e., a cul-de-sac). However, this may be avoided by using more intelligent behaviors such as the following obstacles behavior presented by Hopcroft in [45].

We contrast this approach with the second approach under consideration here, namely the *deliberative* approach. Here, the motion is carefully planned out in advance and care can be taken to minimize energy consumption and so on [46, 47, 48, 49, 50, 51]. This plan-based approach has proved useful in structured environments, e.g., in industrial settings, while unstructured environments pose a challenge. This is due to the fact that there is normally a hefty computational burden associated with path planning and optimal control. And, even if one is willing to pay this cost once, as soon as unmodelled obstacles are encountered, the cost will be incurred again.

Although we only distinguished these two approaches, many *hybrid* strategies for combining these two approaches have been offered (see [14] for further discussion). The main idea here is to combine the benefits of both approaches while possibly avoiding the pitfalls of both. In our development, we stay within the reactive navigation architecture but argue that optimality might still be relevant. In particular, optimality is attained by planning over a set of reactive behaviors. In that sense, one can argue that our approach falls into the category of hybrid strategies. The work of Frazzoli *et. al.*([5, 6]) and Bicchi *et. al.* ([7, 8]) is particularly closely related to our

development, since both approach motion planning using symbolized or quantized control systems.

CHAPTER 2
MULTI-MODAL CONTROL PROGRAMS

In this chapter, we address questions regarding the global behaviors of multi-modal systems assuming a collection of modes (i.e., local rules) have already been defined. As hinted to in the introduction, we assume that each mode consists of a feedback control law and a condition for its termination (called an interrupt). In particular, given a collection of such modes, we want to derive a control strategy (i.e., a sequence of modes) to complete a desired task while minimizing some performance criterion. Assuming such a mode sequence is the desired output, note that control task involves mapping symbols to signals, rather than the classical control approach of mapping signals to signals. We show that adopting this view of mapping symbols to signals allows us to use standard reinforcement learning techniques on previously computationally intractable problems, namely for continuous-time control systems, where the states and control signals take on values in uncountably large sets. It should be noted that reinforcement learning is readily applicable when the state and the input spaces are finite sets, and the system is event-driven (e.g., finite state machines or Markov decision processes). See for example [52, 53, 54, 55].

Here is a brief overview of how adopting an MDL view of mapping symbols to signals provides a natural quantization of the state and the input spaces, thus allowing the use of reinforcement learning techniques on systems with continuous state and control spaces. If we start by considering a finite number of feedback laws and interrupts, a finite quantization of the control space is readily obtained. Note that the control set itself is not quantized but rather that the quantization acts at a functional level. This observation takes care of the problem of quantizing the control inputs. Moreover, by adopting a Lebesque sampling strategy where a new state is sampled only when the interrupts trigger, the continuous time problem is transformed into

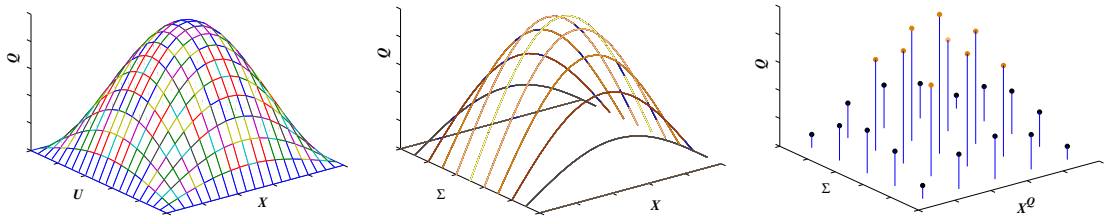


Figure 2. Depicted is the progression from X and U being smooth manifolds (a) to the case when both the state space and the input set are finite (c) through the introduction of multi-modal control procedures and Lebesque sampling.

an event-driven problem. The final piece of the puzzle is the observation that, given an initial state x_0 and a finite length multi-modal program, only a finite number of states are reachable. These ideas are illustrated in Figure 2, where the first figure corresponds to a case where the state space $X \sim \mathbb{R}^n$ and the input space $U \sim \mathbb{R}^m$. Depicted as a function of x and u is the so-called Q-function that characterizes the utility of using control input u at state x. In the next figure, U is replaced by Σ, which corresponds to a finite set of control-interrupt pairs. Without discretizing U, a finite control space is obtained by defining a finite set of available control modes. The final figure shows a situation where both the state space and the input space are finite. The input space is again given by Σ, while X^Q is the quantized state space obtained through an exploration of the states that are reachable from x_0 (in less than N steps) at the distinct times when the interrupts may trigger.

To go from a continuous time control system to a finite state machine is certainly not a new idea. In particular, discretizations of the space-time domain are routinely used for establishing reachability properties. However, such discretizations do not reflect the underlying dynamics in any meaningful way. Alternatives are given in [8], where tokenized control symbols result in reachable lattices, and in [28], where LTL specifications are defined for a quantized system while guaranteeing that the specifications still hold for the original system. The idea of structured state space

explorations was pursued in [56], where the reachable part of the state space was implicitly discretized using rapidly-exploring random trees. Additional results on motion description languages and tokenized control strategies can be found in [7, 57, 2]. Moreover, it is not necessary to let the state space and input space be finite in order to apply learning techniques [55]. For example, a set of basis functions can be defined for supporting the Q-function such as sigmoids, wavelets, or Gaussian kernel functions. However, the computational burden associated with these methods is often prohibitive.

In this chapter, we will make these preliminary, informal observations rigorous. The outline of the chapter is as follows: In Section 2.1, we will discuss reinforcement learning for discrete event-driven systems and see how these techniques can be modified in order to incorporate multi-modal feedback strategies. In Section 2.2, we switch our attention to continuous-time control systems, where the state and control spaces are $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. A robotics example illustrating the potential usefulness of the proposed approach is presented. Additional improvements and refinement issues are treated in Section 2.3, and a brief robustness analysis is discussed in Section 2.4.

2.1 Reinforcement Learning

For systems operating in unknown environments and/or with unknown dynamics, reinforcement learning provides the means for systematic trial-and-error interactions with the environment. Although the aim of this chapter is to apply learning techniques to multi-modal hybrid systems to learn control programs for continuous-time systems, first, we briefly cover the standard reinforcement learning model and learning control programs for discrete-time systems.

2.1.1 Standard Reinforcement Learning

In the standard reinforcement-learning model, at each step (discrete time), the agent chooses an action, $u \in U_F$, based on the current state, $x \in X_F$, of the environment, where U_F and X_F are finite sets (hence the subscript F). The corresponding result is given by $x_{k+1} = \delta(x_k, u_k)$, where $\delta : X_F \times U_F \to X_F$ is the state transition function that encodes the system dynamics. Moreover, a cost $c : X_F \times U_F \to \mathbb{R}$ is associated with taking action u at state x. The agent should choose actions to minimize the overall cost. Given a policy $\pi : X_F \to U_F$, the discounted cost that we wish to minimize is given by

$$V^{\pi}(x_0) = \sum_{k=0}^{\infty} \gamma^k c(x_k, \pi(x_k)), \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor and $x_{k+1} = \delta(x_k, \pi(x_k))$, $k = 0, 1, \ldots$

We use $V^*(x)$ to denote the minimum discounted cost incurred if the agent starts in state x and executes the optimal policy, denoted by π^*. In other words, the optimal value function is defined through the Bellman equation

$$V^*(x) = \min_{u \in U_F} \big[c(x, u) + \gamma V^*(\delta(x, u))\big], \forall x \in X_F. \tag{2}$$

Equation (2) simply states that the optimal value is obtained by taking the action that minimizes the instantaneous cost plus the remaining discounted cost. Once V^* is known, the optimal policy, π^*, follows directly through

$$\pi^*(x) = \min_{u \in U_F} \big[c(x, u) + \gamma V^*(\delta(x, u))\big], \tag{3}$$

which shows why knowing V^* is equivalent to knowing the optimal policy.

If we now let $Q^*(x, u)$ be the discounted cost for taking action u in state x and then continuing to act optimally, we observe that $V^*(x) = \min_u Q^*(x, u)$, and therefore

$$Q^*(x, u) = c(x, u) + \gamma \min_{u' \in U_F} Q^*(\delta(x, u), u'). \tag{4}$$

To find Q^*, we start by assigning a uniform value to every state-action pair, and then randomly selecting state-action pairs (x, u) and updating the Q-table using the

following Q-learning law

$$Q_k(x,u) := Q_{k-1}(x,u) + \alpha_k\Big(c(x,u) + \gamma \min_{u' \in U_F}\Big\{Q_{k-1}(\delta(x,u),u') - Q_{k-1}(x,u)\Big\}\Big). \quad (5)$$

If each action is selected at each state an infinite number of times on an infinite run and α_k, the learning rate, is decayed appropriately, the Q values will converge to Q^* with probability 1. By appropriate decay of α_k we mean that $\sum_k \alpha_k = \infty$, while $\sum_k \alpha_k^2 < \infty$; hence, decreasing the learning rate over time (e.g., $\alpha_k = 1/k$) will guarantee convergence. For more details regarding reinforcement learning, see [52, 53, 54, 55].

2.1.2 Learning Control Programs for Discrete-Time Systems

We now define a new input space that corresponds to tokenized descriptions of feedback laws and interrupts, as prescribed within the MDLe framework. Instead of interacting with the environment at each step, the agent takes actions based on a feedback law κ, which is a function of the state x. The agent furthermore continues to act on the feedback control law κ until an interrupt ξ triggers, at which point a scalar cost is incurred.

Formally, let X_F and U_F be finite sets, as defined earlier, and let $\Sigma = K \times \Xi$, where $K \subseteq {U_F}^{X_F}$ (the set of all maps from X_F to U_F) and $\Xi \subseteq \{0,1\}^{X_F}$. Moreover, let $\tilde{\delta} : X_F \times \Sigma \to X_F$ be the state transition mapping, $\tilde{x}_{k+1} = \tilde{\delta}(\tilde{x}_k, (\kappa_k, \xi_k))$, obtained through the following free-running, feedback mechanism [2]: Let $\tilde{x}_0 = x_0$ and evolve x according to $x_{k+1} = \delta(x_k, \kappa_0(x_k))$ until the interrupt triggers, i.e., $\xi_0(x_{k_0}) = 1$ for some index k_0. Now, let $\tilde{x}_1 = x(k_0)$ and repeat the process, i.e., $x_{k+1} = \delta(x_k, \kappa_1(x_k))$ until $\xi_1(x_{k_1}) = 1$. Now, let $\tilde{x}_2 = x(k_1)$, and so on. Also, let $\zeta : X_F \times \Sigma \to \mathbb{R}$ be the cost associated with the transition.

We want to apply reinforcement learning to this model. To accomplish this we must make a few modifications. First, note that $card(\Sigma)$ is potentially much larger than $card(U_F)$, where $card(\cdot)$ denotes the cardinality. This directly affects the number

of entries in our Q-table. If all possible feedback laws and interrupts were available, the cardinality of the new input space would be $[2card(U_F)]^{card(X_F)}$ with obvious implications for the numerical tractability of the problem.

Second, to find Q^*, we start again by assigning a uniform value to every state-action pair, and then iteratively updating the Q values by randomly selecting a state-action pair with the action comprising of one of the possible feedback laws in K and interrupts in Ξ. The consequent Q-learning law is

$$Q_k(x,(\kappa,\xi)) := Q_{k-1}(x,(\kappa,\xi)) + \alpha_k\Big(\zeta(x,(\kappa,\xi)) + \\ +\gamma \min_{(\kappa',\xi')}\Big\{Q_{k-1}(\tilde{\delta}(x,(\kappa,\xi)),(\kappa',\xi')) - Q_{k-1}(x,(\kappa,\xi))\Big\}\Big). \tag{6}$$

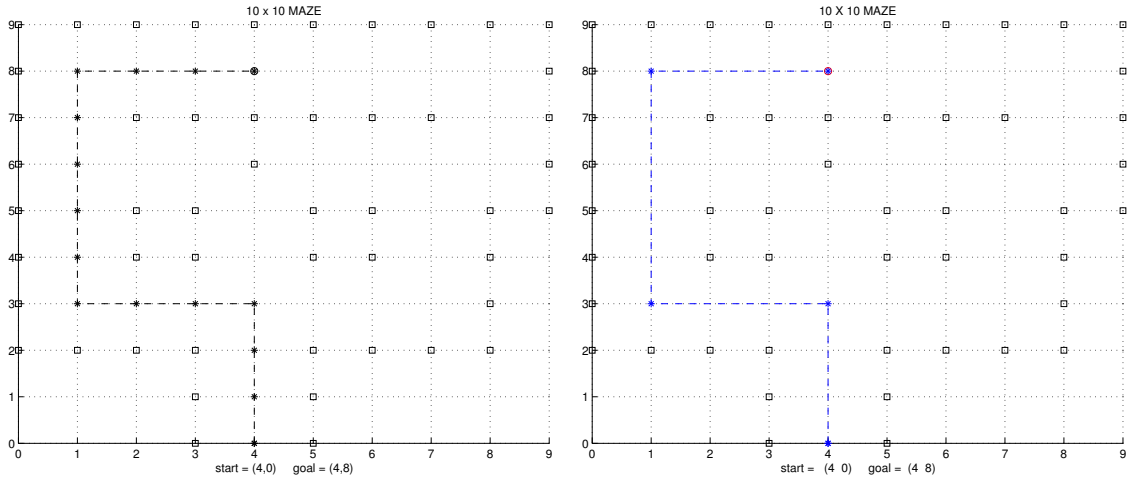
Since Ξ and K are finite, the set of all possible modes Σ is finite as well. Hence, the convergence results still hold as long as each mode is selected for each state an infinite number of times and α_k decays appropriately.

2.1.3 Maze Example

Consider the problem of an agent navigating a $M \times M$ planar grid (we will let $M = 10$) with obstacles. For any of the M^2 possible positions, the agent can move either north (N), south (S), east (E), west (W), or not at all (ϵ). Each such action, except of course ϵ, advances the agent one step, and it is understood that there is a boundary along the perimeter of the grid that the agent can not cross. Moreover, the agent can advance through obstacles even though a hefty cost is incurred whenever this happens. Starting from an arbitrary location, the agent needs to find the shortest path to a specified goal, while avoiding obstacles.

We can restate this problem as a reinforcement learning problem, where the agent must learn the optimal policy given the model of the environment. Formally, we have

- $x = (x_1, x_2)$, where $x_1, x_2 \in \{0, 1, 2, \ldots, M-1\}$;
- $u \in \{N, S, E, W, \epsilon\}$;



(a) (b)

Figure 3. Robot navigating through a maze using a standard reinforcement-learning model (left) and using modes with interrupts as the control set (right).

- $\delta(x,u) = \begin{cases} (x_1, \min\{x_2+1, M-1\}) & \text{if } u = N \\ (x_1, \max\{x_2-1, 0\}) & \text{if } u = S \\ (\min\{x_1+1, M-1\}, x_2) & \text{if } u = E \\ (\max\{x_1-1, 0\}, x_2) & \text{if } u = W \\ (x_1, x_2) & \text{if } u = \epsilon \end{cases}$

- $c(x,u) = \begin{cases} 0 & \text{if } \delta(x,u) = x_{goal} \\ 100 & \text{if } \delta(x,u) \in \mathcal{O} \\ 1 & \text{otherwise} \end{cases}$

Here, x_{goal} is the goal state, while $\mathcal{O} \subset X$ is the set of obstacles. Using standard Q-learning, as previously described, the agent quickly learns the shortest path to the goal and the resulting simulation is shown in Figure 3(a).

In this example, each input corresponds to one step in the maze. However, one could ask the question about the shortest mode string that makes the agent reach the goal, following the development in [2]. Unfortunately, the total number of feedback laws is $card(K) = card(U_F)^{card(X_F)}$, i.e., in this example we have 5^{100} possible control

modes, which is a numerically intractably large number. Hence, we have to reduce the size of K, and our particular choice is the set of constant feedback laws, i.e. $K = \{\kappa_N, \kappa_S, \kappa_E, \kappa_W, \kappa_\epsilon\}$, where $\kappa_N(x) = N,\ \forall x \in X_F$, and so on. Similarly, we need to limit the size of the interrupt set, and we simply let Ξ be set of interrupts that trigger after m steps, $m = 1, 2, \ldots, N$ (we denote these interrupts by $\Xi = \{\xi_1, \ldots, \xi_N\}$). In this case $card(\Xi) = N$, and for the maze problem, we let $N = 9$ (since $M = 10$). Thus, we need $9 \times 5 \times 100 = 4500$ entries in the Q-table. Note that, in order to keep track of the number of steps, the state space has to be augmented in a straightforward manner.

In order to find Q^*, and consequently the optimal policy, we start by assigning a uniform value to every state-action pair (recall we have 4500 possible such pairs). We then randomly select a state-action pair and update its Q-value according to the previously discussed, modified Q-learning law. The result of the simulation is shown in Figure 3(b). Note that this may not always be the shortest path in terms of length (even though it happens to be the shortest in this particular case), but it is the optimal path in terms of the length of the mode string.

2.2 Learning Control Programs for Continuous-Time Systems

Now that the discrete-time case with finite state and input spaces is covered, we shift focus to the problem of learning multi-modal control programs for continuous-time systems. Suppose we have the following system:

$$\dot{x} = f(x, u), \quad \text{where } x \in X = \mathbb{R}^n, u \in U = \mathbb{R}^m, \text{ and } x(t_0) = x_0 \text{ is given.} \qquad (7)$$

If at time t_0, the system receives the input string $\sigma = (\kappa_1, \xi_1), \ldots, (\kappa_q, \xi_q)$, where $\kappa_i : X \to U$ is the feedback control law, and $\xi_i : X \to \{0, 1\}$ is the interrupt, then x

evolves according to

$$\begin{aligned} \dot{x} &= f(x, \kappa_1(x)); \quad t_0 \leq t < \tau_1 \\ &\vdots \qquad\qquad\quad \vdots \\ \dot{x} &= f(x, \kappa_q(x)); \quad \tau_{q-1} \leq t < \tau_q, \end{aligned}$$

where τ_i denotes the time when the interrupt ξ_i triggers (i.e., changes from 0 to 1).

We are interested in finding a sequence of control-interrupt pairs that minimizes a given cost for such a system. For example, we might be interested in driving the system to a certain part of the state space (e.g., to the origin) and penalize the final deviation from this target set. Previous work on reinforcement learning for continuous-time control systems can broadly be divided into two different camps. The first camp represents the idea of a direct discretization of the temporal axis as well as the state and input spaces (e.g., [26, 40, 58]). The main criticism of this approach is that if the discretization is overly coarse, the control optimizing the discretized problem may not be very good when applied to the original problem. Of course, this complication can be moderated somewhat by making the discretization finer. Unfortunately, in this case, the size of the problem very quickly becomes intractable.

The second approach is based on temporal discretization (sampling) in combination with the use of appropriate basis functions to represent the Q-table (e.g. [11, 42, 55]). Even though this is a theoretically appealing approach, it lacks in numerical tractability. In contrast to both of these approaches, we propose to let the temporal quantization be driven by the interrupts directly (i.e., not by a uniform sampling) and let the control space have finite cardinality through the interpretation of a control symbol as a tokenized control-interrupt pair. In other words, by considering a finite number of feedback laws $\kappa_i : X \to U,\ i = 1, \ldots, M$, together with interrupts $\xi_j,\ j = 1, \ldots, N$, the control space (viewed at a functional level) is finite even though the actual control signals take on values in $\mathbb{R}^m$. Another effect of the finite mode-set

assumption is that it provides a natural quantization of the state space. Moreover, if we bound the length of the mode sequences, this quantization results in a finite set of reachable states.

Given an input $\sigma = (\kappa, \xi) \in \Sigma$, where $\Sigma \subseteq U^X \times \{0,1\}^X$, the flow is given by

$$\phi(x_0, \sigma, t) = x_0 + \int_0^t f(x(s), \kappa(x(s)))ds. \tag{8}$$

If there exists a finite time $T \geq 0$ such that $\xi(\phi(x_0, \sigma, T)) = 1$, then we let the interrupt time be given by

$$\tau(\sigma, x_0) = min\{t \geq 0 \mid \xi(\phi(x_0, \sigma, t)) = 1\}. \tag{9}$$

If no such finite time T exists, then we say that $\tau(\sigma, x_0) = \tau_\infty$ for some distinguishable symbol τ_∞. Furthermore, we let the final point on the trajectory generated by σ be

$$\chi(\sigma, x_0) = \phi(x_0, \sigma, \tau(\sigma, x_0))$$

if $\tau(\sigma, x_0) \neq \tau_\infty$ and use the notation $\chi(\sigma, x_0) = \chi_\infty$ otherwise. Moreover, let $\chi(\sigma, \chi_\infty) = \chi_\infty, \forall \sigma \in \Sigma$.

This construction allows us to define the Lebesque sampled finite state machine $(X_N^Q, \Sigma, \tilde{\delta}, \tilde{x_0})$, where N is the longest allowable mode string, and where the state transition is given by

$$\tilde{x}_0 = x_0$$
$$\tilde{x}_{k+1} = \tilde{\delta}(\tilde{x}_k, \sigma_k) = \chi(\sigma_k, \tilde{x}_k), k = 0, 1, \ldots$$

The state space X_N^Q is given by the set of all states that are reachable from $\tilde{x}_0$ using mode strings of length less than or equal to N.

Now that we have a finite state machine describing the dynamics, we can directly apply the previously discussed reinforcement learning algorithm, with an appropriate cost function, to obtain the optimal control program. However, to preserve computing

Table 1. An algorithm for simultaneously exploring the state space and learning the optimal control program.

$\mathcal{X} := \{\tilde{x}_0, \tilde{\delta}(\tilde{x}_0, \sigma)\},\ \forall \sigma \in \Sigma$
$step(\tilde{x}_0) := 0$
$step(\tilde{\delta}(\tilde{x}_0, \sigma)) := 1,\ \forall \sigma \in \Sigma$
$p := 1$
$Q_p(\tilde{x}, \sigma) := const\ \forall \tilde{x} \in \mathcal{X}, \sigma \in \Sigma$
`repeat`
  $p := p + 1$
  $\tilde{x} := rand(\chi \in \mathcal{X} \mid step(\chi) < N)$
  $\sigma := rand(\Sigma)$
  $\tilde{x}' := \tilde{\delta}(\tilde{x}, \sigma)$
  `if` $\tilde{x}' \notin \mathcal{X}$ `then`
    $step(\tilde{x}') := step(\tilde{x}) + 1$
    $\mathcal{X} := \mathcal{X} \cup \{\tilde{x}'\}$
    $Q(\tilde{x}', \sigma) := const\ \forall \sigma \in \Sigma$
  `else`
    $step(\tilde{x}') := \min(step(\tilde{x}'), step(\tilde{x}) + 1)$
  `end if`
  $Q_p(\tilde{x}, \sigma) := Q_{p-1}(\tilde{x}, \sigma)$
    $+ \alpha_p\Big(\zeta(\tilde{x}, \sigma) + \gamma \min_{\sigma' \in \Sigma}\Big\{Q_{p-1}(\tilde{x}', \sigma') - Q_{p-1}(\tilde{x}, \sigma)\Big\}\Big)$
`until` $mod(p, L) = 0$ *and* $|Q_p(\tilde{x}, \sigma) - Q_{p-L}(\tilde{x}, \sigma)| < \epsilon,\ \forall\ \tilde{x} \in \mathcal{X}, \sigma \in \Sigma$
$X_N^Q = \mathcal{X}$

resources, we run this in parallel with the state exploration. The general algorithm for accomplishing this is given in Table 1.

Unlike the earlier Q-learning algorithm, the state space is initially unknown for this case, and we thus begin learning/exploring from the states we know (namely $\tilde{x}_0$ and all the states reachable in one step). At each iteration of the learning process, we select a state randomly from the set of known states and select a mode randomly from the set of modes. In the algorithm, the function $step(\tilde{x})$ represents the length of the shortest control program used so far to reach state $\tilde{x}$ from the initial state $\tilde{x}_0$. This is to ensure that we only explore states that are reachable from $\tilde{x}_0$ using mode strings of length less than or equal to N, i.e., $\mathcal{X} \subseteq X_N^Q$. We then calculate the next state and determine if it is a member of our known state space (In practice, it may be necessary to check if the next state belongs to a neighborhood of a previously visited state). If not, we add this state to the known state space and make the corresponding change in the Q-table. We continue to explore and update the state space and our Q-table (or value function) in this manner until the Q-table is stationary. Note that in the algorithm, $\epsilon > 0$ is a small positive scalar and L is a large number needed to ensure that a sufficient number of state-action pairs are visited.

In summary, the algorithm above effectively estimates the reachable set while applying reinforcement learning over this estimated set to obtain the optimal mode sequence. Some robustness concerns regarding the presented method are addressed in Section 2.4.

2.2.0.1 Example

As an example, consider the following system:

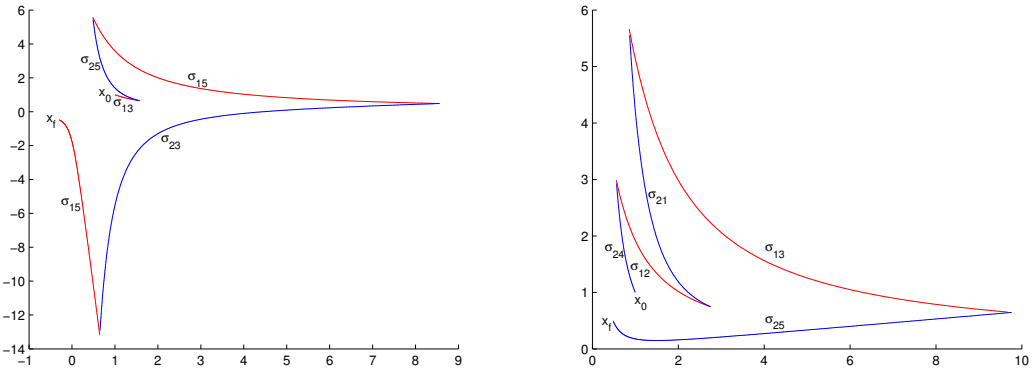
$$\dot{x} = u, \qquad x \in \mathbb{R}^2, \qquad x_o = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \text{with modes } \sigma_{ij} = (\kappa_i, \xi_{ij}).$$

Suppose κ_i and ξ_{ij} are defined as follows:

$$\kappa_1(x) = \begin{pmatrix} 1 & 0.1 \\ 0 & -1 \end{pmatrix} x, \qquad \kappa_2(x) = \begin{pmatrix} -1 & 0 \\ -0.2 & 2 \end{pmatrix} x,$$

$$\xi_{1j} = \begin{cases} 1 & \text{if } x_2^2 < M \cdot \delta^j \\ 0 & \text{otherwise} \end{cases}, \qquad \xi_{2j} = \begin{cases} 1 & \text{if } x_1^2 < M \cdot \delta^j \\ 0 & \text{otherwise} \end{cases} \qquad for \quad j = 1, 2, \ldots, 5.$$

Note that the system is unstable in either mode. We want to learn if there exists a mode sequence that will stabilize the system. Although it may not be possible to drive the system to $x = 0$, we want to know if there is a string of modes that can bring the system in a neighborhood around the equilibrium point $x = 0$. To make this feasible, we limited the system to two modes and five interrupts in each mode. The reachable set of states in this case has length $2\sum_{i=0}^{N} 5^i$, where N is the maximum number of steps (or string length). As can be seen, the state space increases exponentially with respect to the length of the control program. To make the learning process manageable, we limit the size of the string length to five (i.e., $N = 5$). Since we want to stabilize the system in the minimum number of steps (which must be less than or equal to 5), it is natural to assign a cost for each switch and penalize on the final position. We will also add a cost for the trajectory since we want to minimize the control efforts. Just to see a variation in the results, we conducted two experiments. The resulting plots from the simulation (with $M = 1$ and $\delta = 0.75$) are shown in Figure 4. In the first experiment, we assume that the system always starts using κ_1. The resulting optimal mode string is $\bar{\sigma}^* = \sigma_{13}\sigma_{25}\sigma_{15}\sigma_{23}\sigma_{15}$. In the second experiment, we removed this assumption, and the optimal mode string is $\bar{\sigma}^* = \sigma_{24}\sigma_{12}\sigma_{21}\sigma_{13}\sigma_{25}$. This fact is apparent when we compare the trajectories from the two simulations since the total length of the second trajectory (b) is shorter than the first one (a).



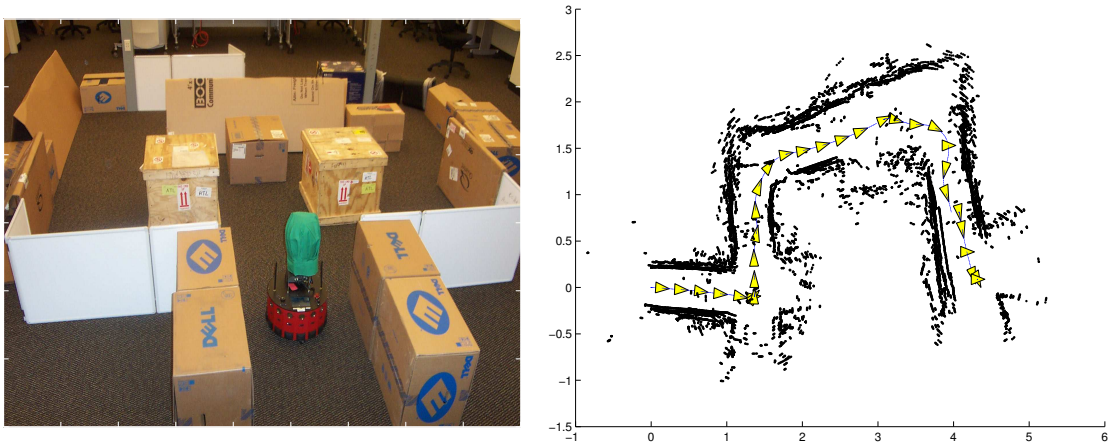
(a) (b)

Figure 4. Simulation results demonstrating the learning of optimal mode strings in a continuous-time system.

2.2.1 Maze Revisited

We now apply this strategy for obtaining finite state machine descriptions of continuous time multi-modal control systems to the previously discussed maze problem. The experiment will be conducted on the Magellan Pro Mobile Robot platform from iRobot. The Magellan Pro platform will be used for various robotics applications throughout this thesis. The robot is driven by two active wheels independently driven by two dc motors and one caster wheel. The robot features 16 bump sensors, 16 infrared sensors, 16 ultrasonic sensors, and a color camera. For our experiments, we will primarily use the infrared and ultrasonic sensors providing a sensing-range of approximately 2 meters. The control architecture allows us to send linear and angular velocity commands as control variables. Moreover, the robot runs on a carpeted floor in the lab, thus allowing us to ignore wheel-slippage. Hence, the dynamics of the robot can be accurately captured by using a unicycle model, i.e.,

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega, \end{aligned} \tag{10}$$



(a) (b)

Figure 5. (a) The experimental setup of the maze. (b) The path of the robot together with the range sensor readings (IR-based) obtained throughout the final run. Note how the odometric drift makes the maze look somewhat distorted.

where (x, y) are the Cartesian coordinates of the center of the robot, and ϕ is its orientation with respect to the x-axis. The linear velocity v and angular velocity ω are the control variables.

For the purpose of this experiment, we still use the mode set $\{N, S, E, W, \epsilon\}$, but define it for a planar integrator (i.e., a unicycle) instead of a finite state machine. Moreover, we let the interrupts, which previously counted the number of steps taken, correspond to a certain distance travelled. We apply this scheme to the problem of making a robot negotiate a maze, and Figure 5 (a) shows the experimental setup of the maze and the Magellan Pro robot used negotiate the maze. Figure 5 (b) shows final path obtained through the learning algorithm.

2.3 Refining the Learning Process

In this section, we discuss some methods for enhancing the learning process. In particular, for problems with large state and input spaces (basically all interesting problems), the convergence is typically slow when using a purely random exploration strategy. However, it is well-known that one can use knowledge about the problem

(i.e., heuristics) in order to speed up the learning process. The idea is to start out the learning process completely at random, but as the system gains "experience" the state space exploration becomes less and less random. In other words, we bias the selection of the state-action pairs to explore and update based on current values of the Q-table.

In order to formalize this, we need to introduce some notation. Let $P(x, u)$ denote the probability of selecting state-action pair (x, u) from $X_F \times U_F$, with

$$\sum_{x \in X} \sum_{u \in U} P(x, u) = 1.$$

Initially, we begin with

$$P_0(x, u) = \frac{1}{card(X)card(U)}. \tag{11}$$

In other words, every state-action pair has an equal likelihood of being selected. As we gain experience, we can change these probabilities to bias the selection in favor of state-action pairs with lower Q-values (potentially "good" state-action pairs). There may be many appropriate methods for biasing these probabilities, and one simple approach is to let the probability of selection state-action pair (x, u) be given by

$$P_k(x, u) = \frac{Q_{k-1}(x, u)^{-1}}{\sum_{x' \in X_F} \sum_{u' \in U_F} Q_{k-1}(x', u')^{-1}}. \tag{12}$$

Given such a biased probability distribution, we do not want to use it prematurely, for this may lead us to not learn the optimal policy. Instead, we want to introduce a confidence value, $c \in [0, 1]$, which is based on the time step k and the past Q-values. With a lower value of c, the exploration strategy should be more random, while higher value of c suggest using a more biased exploration strategy. Note that we still want to leave some amount of randomness in the selection process in order to ensure that the entire state and input space is explored. The degree of bias in the selection process and the necessary experience will vary from problem to problem.

Based on our knowledge of the problem, we can also start pruning the state-space as we gain experience. This means that we could exclude states that we are certain

(possibly with high probability) are not part of the optimal trajectory. This reduction in the size of the state-space enables the learning process to converge faster since the plausible state-action pairs can be selected more often. However, great caution and high degree of accuracy must be used when pruning the state-space to ensure that the optimal policy is still learned since incorrectly pruning a potentially useful state may mean that only a sub-optimal policy is learned.

2.4 Robustness Analysis

Note that the entire argument presented in this chapter concerning the finite state space model hinges on the fact that we start from a fixed initial state. In this section, we will conduct a sensitivity analysis to show that if the mode string $\hat{\sigma}$ is optimal when starting at x_0, it is in fact still optimal for $\tilde{x}_0 = x_0 + \Delta x_0$, for some small perturbation Δx_0. It is sufficient to show that if x_0 is perturbed a little, then $\tilde{x}_f$, the point obtained after executing $\hat{\sigma}$ from $\tilde{x}_0$, lies within a small neighborhood of x_f, i.e., we need to show that $\Delta x_f = x_f - \tilde{x}_f$ is small.

In order to simplify the notation, we let the interrupt surfaces be encoded by smooth functions $g_i(x) = 0$, i.e. $\xi_i(x) = 1$ when $g_i(x) = 0$ and $\xi_i(x) = 0$ otherwise. Also, the trajectory of x is given by $x(t) = \Phi_1(t, t_0)$ until $g_1(x) = 0$. Then it is given by $x(t) = \Phi_2(t, \tau_1)$ until $g_2(x) = 0$, and so on. Here Φ_i is the state-transition function associated with $\dot{x} = f(x, \kappa_i(x))$, and τ_i is the time that interrupt ξ_i triggers, i.e., $g_i(x(\tau_i)) = 0$. Moreover we will denote this point $x_{h_i} = x(\tau_i)$. So for $t \in [0, \tau_1)$, we get

$$\begin{aligned} \dot{\tilde{x}} &= f_1(\tilde{x}, u) = f_1(x + \Delta x_0, u) \\ &= f_1(x, u) + \frac{\partial f_1}{\partial x}\Delta x_0 + o(\Delta x). \end{aligned} \tag{13}$$

Hence,

$$\Delta \dot{x} = \frac{\partial f_1}{\partial x}\Delta x_0 + o(\Delta x), \tag{14}$$

meaning that for $t \in [0, \tau_1)$, $\Delta x(t) = \Phi_1(t, t_0)\Delta x_0 + o(\Delta x)$. To examine the trajectory after the interrupt, we have to calculate the change in the interrupt time τ_1 and the position at this time, namely x_{h_1}. Again, using a first order approximation, we get

$$\begin{aligned}\tilde{x}(\tau_1 + \Delta\tau_1) &= x(\tau_1 + \Delta\tau_1) + \Delta x(\tau_1 + \Delta\tau_1) \\ &= x(\tau_1) + f_1(x(\tau_1))\Delta\tau_1 + \Delta x(\tau_1) + o(\Delta\tau_1). \end{aligned} \tag{15}$$

Here $t = \tau_1 + \Delta\tau_1$ is the time that the trajectory of $\tilde{x}$ hits the interrupt surface, so we must have

$$g_1(\tilde{x}(\tau_1 + \Delta\tau_1)) = 0,$$

which implies that

$$g_1(x(\tau_1)) + \frac{\partial g_1}{\partial x}(x(\tau_1))\big[f_1(x(\tau_1))\Delta\tau_1)\big] + \frac{\partial g_1}{\partial x}(x(\tau_1))\Delta x(\tau_1) + o(\Delta\tau_1) = 0. \tag{16}$$

Letting $L_{f_1}g_1(x(\tau)) := \frac{\partial g_1}{\partial x}(x(\tau_1))f_1(x(\tau_1))$, which is the Lie derivative of g_1 with respect to x along flow f_1, and assuming that this quantity is non-zero, we get

$$\Delta\tau_1 = -\frac{\frac{\partial g_1}{\partial x}(x(\tau_1))\Phi_1(\tau_1, t_0)\Delta x_0}{L_{f_1}g_1(x(\tau_1))}, \tag{17}$$

where we have ignored higher order terms. Hence,

$$\begin{aligned}\Delta x_{h_1} &= \tilde{x}(\tau_1 + \Delta\tau_1) \\ &= \Big[I - \frac{f_1 \frac{\partial g_1}{\partial x}(x(\tau_1))}{L_{f_1}g_1(x(\tau_1))}\Big]\Phi_1(\tau_1, t_0)\Delta x_0. \end{aligned} \tag{18}$$

Now, based on the assumption that $L_{f_1}g_1(x(\tau_1)) \neq 0$ (i.e., the interrupt triggers non-tangentially), Δx_{h_1} is small. Similarly, we get that Δx_{h_2} is small under the assumption that $L_{f_2}g_2(x(\tau_2)) \neq 0$. Continuing in this manner, we deduce that Δx_f will be small as long as $L_{f_i}g_i(x(\tau_i)) \neq 0$, for $i = 1, \ldots, M$, and the result follows.

2.5 Conclusions

In this chapter, we presented a method for going from continuous-time control systems to finite state machines in a structured manner. In particular, by only considering a

finite number of modes, i.e., control-interrupt pairs, the input space is finite and the continuous-time dynamics can be replaced by a Lebesque sampled, discrete-time system. Moreover, by limiting the length of the mode string, the reachable state space (at the interrupt times) is finite as well. This construction means that previously unavailable computational methods, such as reinforcement learning, are now applicable in a straight forward manner. In summary, we presented an algorithm that estimates the reachable set (thus, encoding the expressiveness of the system) and learns control programs that optimize a prescribed performance index while ensuring task completion (when possible).

It should be mentioned that this method is based on assumptions about sufficient knowledge of the system dynamics, initial conditions, and the environment. The optimal control program should be interpreted as a high-level plan over a set of available modes. In this case, the disturbances in the dynamics can be handled by a low-level controller. In Section 2.4, we conducted a robustness analysis to show that the learning algorithm is robust to small errors in the initial condition; however, the sensitivity of the algorithm with respect to errors in the environment is still unexplored. To add robustness to errors in the environment, it may be beneficial to extend this algorithm to dynamically update the control program whenever unmodelled obstacles are encountered. The abundance of literature on reinforcement learning in unknown environments (see [52, 53, 55, 58]) can facilitate this extension.

CHAPTER 3
ADAPTIVE MULTI-MODAL CONTROL

Now that we have a method in which strings of control modes (i.e., control programs) can be produced given a collection of modes, one natural question to ask is whether we can adaptively vary the mode set to further improve performance. As mentioned earlier, one of the strengths of multi-modal control is that it allows us to add new functionality (or modes) to the system without adding significant increase in complexity. Hence, instead of changing the existing modes to improve performance, we intend to introduce new modes to the mode set in a structured manner to improve performance. In this chapter, we will develop the basic framework for adaptive multi-modal control, which will be expanded in Chapter 4.

First, let us introduce some motivation for how adaptively changing the mode set can improve performance. As hinted to in the introduction to motion description languages (see Chapter 1), the key issue in any symbol driven system is the tradeoff between complexity and expressiveness. Adaptive multi-modal control is an attempt to increase the expressiveness of the system (thus possibly increasing the performance of the system) while decrease the complexity of the control programs. If we let Σ be the set of available modes and let the mode string $\bar{\sigma} = \sigma_1\sigma_2\cdots\sigma_q$ solve a particular control task (e.g., one that drives the system to the origin), then we can define the complexity of this control program as the number of bits needed for its encoding, as was the case in [1]. In other words, the complexity of the control program $\bar{\sigma}$, whose elements are drawn from the mode set Σ, is given by

$$|\bar{\sigma}| \log_2(card(\Sigma)).$$

Remember, we are interested in designing new modes in a highly structured manner to increase expressiveness while reducing complexity. Moreover, the mode string should be readily updated to account for these new modes without incurring any

hefty computational costs. Our proposed solution is based on the observation that if a given mode string fragment $\sigma_1 \cdots \sigma_r$ occurs (possibly repeatedly) in the control program, it might be possible and beneficial to replace this string with one single mode σ_{1r} that results in (roughly) the same behavior. If p occurrences of $\sigma_1 \cdots \sigma_r$ are replaced by σ_{1r} in the original control program ($\bar{\sigma}_{old}$), then the complexity of new mode string ($\bar{\sigma}_{new}$) is now

$$(|\bar{\sigma}_{old}| - p(r-1))card(\Sigma_{old} + 1) < |\bar{\sigma}_{old}|card(\Sigma_{old})$$

as long as $p \geq 1$ and $r > 1$. In other words, the complexity would be reduced if such a σ_{1r} could be found. Since the new mode will be designed to produce roughly the same behavior, rather than the same behavior, the modified multi-modal system will be able to produce a larger set of trajectories than the original multi-modal system (i.e., the expressiveness is increased).

In this chapter, we will develop a general framework and algorithms for adaptive multi-modal control by augmenting the mode set with modes that replace recurring mode string fragments. The chapter is organized as follows: In Section 3.1, we look at a specific, motivating example of the continuous-time linear time-invariant (LTI) systems. In Section 3.2, we introduce the general framework for adaptive multi-modal control, where the problem is posed as an optimal control problem and solved using calculus of variations. Section 3.3 presents examples to illustrate the viability of the proposed methods, followed by conclusions in Section 3.4.

3.1 Continuous-Time LTI Systems

Before deriving a general framework and algorithms for adaptive multi-modal control, we will look at a specific, motivating example of the continuous-time linear time-invariant (LTI) systems. Consider the following autonomous linear system:

$$\dot{x}(t) = \begin{cases} A_1 x(t) & \text{if } t \in [0, \frac{T}{2}] \\ A_2 x(t) & \text{if } t \in [\frac{T}{2}, T] \end{cases} . \tag{19}$$

If $x \in \mathbb{R}^n$ and $x(0) = x_0$ is given, then the evolution of x is given as follows:

$$\begin{aligned} x(0) &= x_0 \\ x(\tfrac{T}{2}) &= e^{A_1 \frac{T}{2}} x_0 \\ x(T) &= e^{A_2 \frac{T}{2}} x(\tfrac{T}{2}) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0. \end{aligned}$$

Now, suppose we want to find a new A such that

$$x(T) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0 \approx e^{AT} x_0. \tag{20}$$

One way of obtaining A is through the use of the well-known Campbell-Baker-Hausdorff (CBH) formula, which can be stated as follows:

Campbell-Baker-Hausdorff (CBH) Formula For any two matrices X, Y sufficiently close to 0 , there exists a matrix $Z \in L(X, Y)$ such that $e^Z = e^X e^Y$. Moreover, Z can be explicitly expressed in the Dynkin form as: $Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] + \frac{1}{12}[Y, [Y, X]] + \ldots$, where $[X, Y] = XY - YX$ is the matrix commutator.

Since the CBH formula gives an infinite series, we have to be concerned about convergence when applying the formula. The convergence of the CBH formula has been well studied [59, 60], and it is shown that the Dynkin series converges for matrices X, Y if there is a Lie norm for which

$$\| X \|_{Lie} + \| Y \|_{Lie} \leq \log(2). \tag{21}$$

Here, $\| \cdot \|_{Lie}$ denotes the Lie norm, which is a norm on matrices compatible with Lie multiplication, i.e.,

$$\| [X, Y] \|_{Lie} \leq \| X \|_{Lie} \| Y \|_{Lie} . \tag{22}$$

Clearly, if T is sufficiently small, then $\| A_i \frac{T}{2} \|_{Lie}$ will meet the bound above (21) for $i = 1, 2$. In this case we should be able to approximate this result by using a finite number of elements from the Lie algebra.

Using the CBH formula it is clear that

$$\begin{aligned} A &= \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \tfrac{T^2}{8}[A_1, [A_1, A_2]] + \dots \\ &\equiv \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \Delta(T^2), \end{aligned} \tag{23}$$

where $\Delta(T^2)$ is the remaining part of the series, which is polynomial in T of degree greater than T^2. Now, let us denote $\tilde{A} = \frac{1}{2}A_1 + \frac{1}{2}A_2 + \frac{T}{4}[A_1, A_2]$; we will show that $\| e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \|$ is bounded by $o(T^2)$. Hence, $x(T) \approx e^{\tilde{A}T}$ for a small enough T. First, note the following expression derived in [61]:

$$e^{A+\Delta} - e^{A} = \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau + o(\| \Delta \|). \tag{24}$$

Hence, by manipulating (24) we obtain

$$\begin{aligned} \| e^{A+\Delta} - e^{A} \| &\leq \| \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau \| + o(\| \Delta \|) \\ &\leq \int_0^1 \| e^{(1-\tau)A} \Delta e^{\tau A} \| \, d\tau + o(\| \Delta \|) \\ &\leq \int_0^1 e^{\|A\|} \| \Delta \| \, e^{\|A\|} d\tau + o(\| \Delta \|) \\ &= e^{2\|A\|} \| \Delta \| + o(\| \Delta \|) \end{aligned} \tag{25}$$

So in our case (25) is reduced to

$$\| e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \| \leq e^{2\|\tilde{A}\|} o(T^2). \tag{26}$$

We have thus shown that A given by the CBH formula can be approximated by $\tilde{A}$ for a small enough T. Of course, we can approximate A by using higher-order Lie brackets to obtain a better approximation if desired. Shortly, we will compare this approach with the general approach presented in the next section, which relies on the calculus of variations.

3.2 General Framework for Adaptive Multi-Modal Control

Recall, we are interested in developing a general framework for adaptive multi-modal control by augmenting the mode set with modes that replace recurring mode string

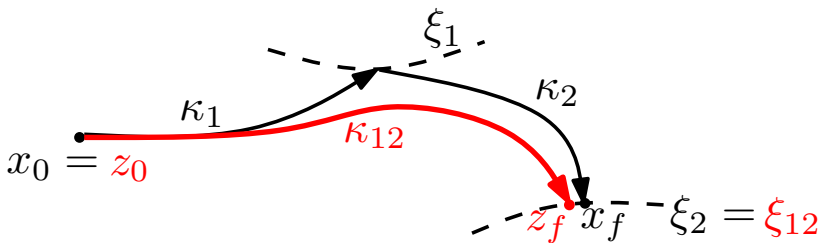


Figure 6. Depicted is the original trajectory $x(t)$ and the approximation trajectory $z(t)$.

fragments. The CBH formula gives us an explicit formula for finding a matrix A that behaves similarly to A_1 followed by A_2. Note that this formula is an infinite series, but this example gives us an idea for a general construction of a "meta-mode" σ_{1r} that replaces $\sigma_1 \cdots \sigma_r$. Namely, we would like this new mode to be constructed as a function of the modes it is replacing. Constructing σ_{1r} involves designing the feedback law κ_{1r} and the interrupt ξ_{1r}. First, we start by letting $\xi_{1r} = \xi_r$ since we want σ_{1r} to behave similarly to $\sigma_1 \cdots \sigma_r$. Now for designing κ_{1r}, we will use an approximation function $z(t)$, which would approximate the trajectory of $x(t)$ given using the mode string fragment $\sigma_1 \cdots \sigma_r$ that we are trying to replace. This idea of using an approximation function is depicted in Figure 6 for the particular example of replacing $\sigma_1\sigma_2$ with σ_{12}. In the general case, we define the approximation trajectory z as follows:

$$\dot{z} = f(z, \kappa_{1r}) \text{ until } \xi_{1r}(x) = \xi_r(x) = 1 \text{ with } z(0) = x(0). \tag{27}$$

Now, the problem of augmenting the mode set is reduced to finding the feedback mapping κ_{1r} that best approximates $x(t)$. Moreover, we insist on this feedback mapping being a function of the feedback laws corresponding to the modes being replaced in the recurring mode sequence. Unfortunately, we cannot let the κ_{1r} be a general function of the existing feedback laws, since this problem would be intractable. Thus, in the section, we constrain the feedback law κ_{1r} to be a linear combination of *basis functions* g_i:

$$\kappa_{1r} = \sum_{i=1}^{N} \alpha_i g_i(z(t)), \text{ with } \alpha_i \in \mathbb{R}. \tag{28}$$

The idea here is to let the basis functions $g_i : \mathbb{R}^n \to \mathbb{R}^m$ be some differentiable function of the existing feedback laws $g_i = \zeta(\kappa_1, \ldots, \kappa_r)$. For the remainder of this development, we do not specify $\zeta(\cdot)$ exactly, but one possibility is to let ζ map to the lie algebra of $\kappa_1, \ldots, \kappa_r$ (as done in the case of continuous-time LTI systems). The problem of finding κ_{1r} is now reduced to choosing $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ so that the performance criterion

$$J(\vec{\alpha}) = \int_0^T L(x(t), z(t))dt + \psi(x(T), z(T)) \tag{29}$$

is minimized, where $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $\psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ are twice differentiable in their second argument. Note here that the instantaneous cost L determines how close we want the approximation trajectory to track the original trajectory, while the terminal cost ψ penalizes the final deviation.

This problem can be solved using a variational approach, where the cost (29) is appended with the constraint given by (27) and (28) via a co-state (or lagrange multiplier) $\lambda(t)$. Next, we perturb the control vector $\vec{\alpha}$ and compute the Gateaux (or directional) derivative of J in the direction of the perturbation to gain access to the optimality conditions. The key concept to remember in this development is that the co-state should be chosen to simplify our computation. More specifically, we will choose the co-state so that we avoid computing the variation in the state trajectories. This computation is detailed next.

Starting by adding the constraint with a co-state $\lambda(t)$ to (29), we obtain

$$\tilde{J}(\vec{\alpha}) = \int_0^T \Big[L(x(t), z(t)) + \lambda(t)\Big(f\big(z(t), \sum_{i=1}^N \alpha_i g_i(z(t))\big) - \dot{z}(t)\Big)\Big]dt + \\ + \psi(x(T), z(T)). \tag{30}$$

Note above that $\tilde{J}(\vec{\alpha})$ denotes the unperturbed cost. Now, we perturb (30) in such a way that $\vec{\alpha} \to \vec{\alpha} + \epsilon\vec{\theta}_k$, where $\vec{\theta}_k = [0, \ldots, \theta_k, \ldots, 0]^T$ (note the k^{th} entry is θ_k and all other entries are 0), and $\epsilon << 1$, then $z \to z + \epsilon\eta$ is the resulting variation

in $z(t)$. Note that above, we dropped the argument t when referring to $z(t)$ and will continue this convention in the following development for compactness, with the implicit understanding that x, z, and λ are functions of t. Now, the perturbed cost is given by

$$\begin{aligned}\tilde{J}(\vec{\alpha}+\epsilon\vec{\theta}_k) = \int_0^T \Big[L(x,z+\epsilon\eta)) + \lambda\Big(f\big(z+\epsilon\eta, \alpha_1 g_1(z+\epsilon\eta) + \alpha_2 g_2(z+\epsilon\eta) + \\ + \ldots + (\alpha_k+\epsilon\theta_k)g_k(z+\epsilon\eta) + \ldots + \alpha_N g_N(z+\epsilon\eta)\big) - \dot{z} - \epsilon\dot{\eta}\Big)\Big]dt + \\ + \psi(x(T),(z+\epsilon\eta)(T)). \qquad (31)\end{aligned}$$

Hence, the Gateaux (also referred to as directional) derivative of $\tilde{J}$ in the direction of $\vec{\theta}_k$ is

$$\begin{aligned}\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) &= \lim_{\epsilon\to 0}\frac{\tilde{J}(\vec{\alpha}+\epsilon\vec{\theta}_k)-\tilde{J}(\vec{\alpha})}{\epsilon} \\ &= \int_0^T \Big[\frac{\partial L}{\partial z}\eta + \lambda\Big(\frac{\partial f}{\partial z}\eta + \sum_{i=1}^N \alpha_i \frac{\partial f}{\partial u}\frac{\partial g_i}{\partial z}\eta + \frac{\partial f}{\partial u}g_k\theta_k - \dot{\eta}\Big)\Big]dt + \frac{\partial \psi}{\partial z}\eta(T). \qquad (32)\end{aligned}$$

Now, by integrating $\lambda\dot{\eta}$ in (32) by parts and rearranging terms, we obtain

$$\begin{aligned}\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \int_0^T \Big[\frac{\partial L}{\partial z} + \lambda\frac{\partial f}{\partial z} + \lambda\sum_{i=1}^N \alpha_i\frac{\partial f}{\partial u}\frac{\partial g_i}{\partial z} + \dot{\lambda}\Big]\eta dt + \\ + \theta_k\int_0^T \lambda\frac{\partial f}{\partial u}g_k(z)dt - \Big[\lambda\eta\Big]_0^T + \frac{\partial\psi}{\partial z}\eta(T) \qquad (33)\end{aligned}$$

Note that $\eta(0)=0$ since $z(0)=x(0)=x_0$. Recall, we want to choose the co-state λ to avoid the computation of the variation η. Thus, we let $\lambda(t)$ be given by

$$\lambda(T) = \frac{\partial\psi}{\partial z}(x(T),z(T)), \qquad (34)$$

$$\dot{\lambda}(t) = -\frac{\partial L}{\partial z}(x,z) - \lambda(t)\Big(\frac{\partial f}{\partial z}(z,\kappa_{1r}) + \sum_{i=1}^N \alpha_i\frac{\partial f}{\partial u}(z,\kappa_{1r})\frac{\partial g_i}{\partial z}(z)\Big). \qquad (35)$$

With this choice of the co-state $\lambda(t)$, which can be solved by integrating (35) backwards with initial condition (34), we obtain

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \Big[\int_0^T \lambda\frac{\partial f}{\partial u}(z,\kappa_{1r})g_k(z)dt\Big]\theta_k \qquad (36)$$

Finally, note that (36) gives access to the partial derivative $\frac{\partial \tilde{J}}{\partial \alpha_k}$ since we know that

$$\nabla_{\vec{\theta}} \tilde{J}(\vec{\alpha}) = \frac{\partial \tilde{J}}{\partial \alpha_1}\theta_1 + \cdots + \frac{\partial \tilde{J}}{\partial \alpha_N}\theta_N, \tag{37}$$

where $\vec{\theta} = [\theta_1, \ldots, \theta_N]^T$. Hence using (36), (37), and the fact that α_k (for $k = 1, \ldots, N$) are independent of each other, we deduce that

$$\frac{dJ}{d\alpha_k} = \int_0^T \lambda \frac{\partial f}{\partial u}\big(z, \kappa_{1r}(z)\big) g_k(z) dt. \tag{38}$$

We summarize these results in the following theorem:

Theorem 3.2.1 *Given a function $x(t) \in \mathbb{R}^n$ and a set of twice differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^m$ for $i = 1, 2, \ldots, N$, with $z(t) \in \mathbb{R}^n$ given by (27) and (28), an extremum to the performance index*

$$J(\vec{\alpha}) = \int_0^T L(x(t), z(t))dt + \psi(x(T), z(T))$$

is attained when the control vector $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ is chosen such that

$$\frac{dJ}{d\alpha_k} = \int_0^T \lambda(t) \frac{\partial f}{\partial u}\big(z(t), \kappa_{1r}(z(t))\big) g_k(z(t)) dt = 0 \quad \text{for } k = 1, 2, \ldots, N,$$

where the co-state $\lambda(t)$ is chosen as follows:

$$\lambda(T) = \frac{\partial \psi}{\partial z}(x(T), z(T)),$$
$$\dot{\lambda}(t) = -\frac{\partial L}{\partial z}(x, z) - \lambda(t)\Big(\frac{\partial f}{\partial z}(z, \kappa_{1r}) + \sum_{i=1}^{N} \alpha_i \frac{\partial f}{\partial u}(z, \kappa_{1r}) \frac{\partial g_i}{\partial z}(z)\Big).$$

3.2.1 Numerics

In the previous section, we derived the necessary conditions that any extremum of the performance index J must satisfy. The theorem, however, gives us very little insight about how to attain an extremum to the performance index. In this section, we present a numerical algorithm that utilizes the optimality conditions derived earlier to converge to a stationary solution (i.e., control parameters that produce an extremum to the performance index J). This algorithm (shown in Table 2) employs a gradient

Table 2. A gradient descent algorithm.

- Initialize with a guess of the control variables $\vec{\alpha}^{(0)}$ and let $p = 0$.
- `while` $p < 1$ `or` $|J^{(p)} - J^{(p-1)}| < \epsilon$
  1. Compute the approximation function $z(t)$ forward in time from 0 to T using (27) and (28) and cost $J^{(p)}$ (29).
  2. Compute the co-state $\lambda(t)$ backward in time from T to 0 using (129) and (135).
  3. Compute the gradient $\nabla J(\vec{\alpha}^{(n)}) = \left[\frac{dJ}{d\alpha_1^{(n)}}, \ldots, \frac{dJ}{d\alpha_N^{(n)}}\right]^T$ using (38).
  4. Update the control variables as follows:
  $$\vec{\alpha}^{(n+1)} = \vec{\alpha}^{(n)} - \gamma^{(n)} \nabla J(\vec{\alpha}^{(n)})$$
  5. $p = p + 1$
- `end while`

descent method, in which, the control parameters are updated in the negative gradient direction until a stationary solution has been reached.

Note that the choice of the step-size $\gamma^{(n)}$ can be critical for the method to converge. An efficient method among others is the use of Armijo's algorithm presented in [62]. Because of the non-convex nature of the cost function J, this gradient descent algorithm will only converge to a local minimum. Hence the attainment of a "good" local minimum can be quite dependent on the choice of a "good" initial guess for the control variables. However, the method presented here may still offer significant reductions in the performance index. The association of such a local method with heuristic strategies to find a global minimum is not investigated here.

3.3 Examples

In this section, we consider some examples that illustrate the viability of our approach. We first consider a specific example of a continuous-time LTI system, where the

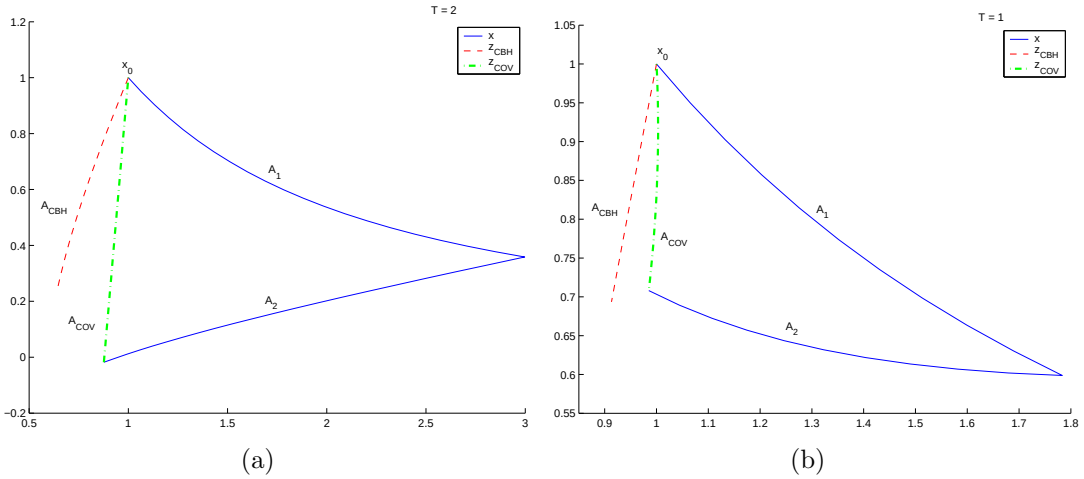
variational method is compared to the CBH solution presented earlier. Then, we look at an example of a unicycle navigating through a cluttered environment to a specified goal location. This latter example illustrates the learning of optimal mode strings and shows how adaptive multi-modal control can further improve performance.

3.3.1 LTI Example

In this section, we present a simple example of a linear, time-invariant (LTI) system to illustrate the operation of the gradient descent algorithm derived earlier. Let

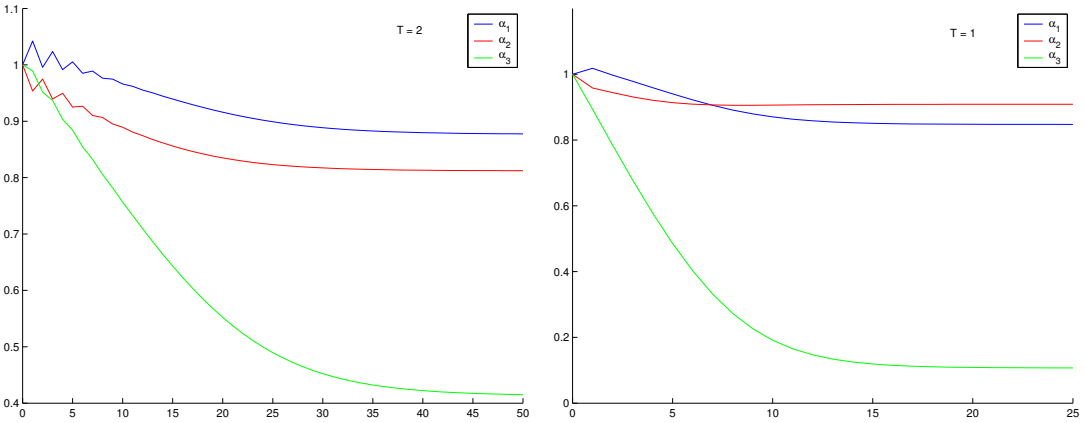
$$\dot{x}(t) = \begin{cases} \begin{pmatrix} 1 & 0.3 \\ 0 & -1 \end{pmatrix} x(t); & \text{if } 0 \leq t < \frac{T}{2} \\ \begin{pmatrix} -1.2 & 0.1 \\ -0.3 & 1 \end{pmatrix} x(t); & \text{if } \frac{T}{2} \leq t < T \end{cases} . \tag{39}$$

Suppose $x_0 = (1,1)^T$, and suppose we want to derive a new matrix A_{new} that transfers the system from x_0 to $x(T) \approx e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0$ in time T without the switch at time $\frac{T}{2}$. To this end, we let the instantaneous cost $L(x,z) = 0$ and the terminal cost $\psi(x(T), z(T)) = \| x(T) - z(T) \|^2$. Figure 7 (a) shows the trajectories obtained using the first-order approximation of the CBH formula and the corresponding calculus of variation approximation using $\hat{A}_{new} = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_2 [A_1, A_2]$ with $T = 2$. Note that the COV approach obtained a virtually perfect match, while the CBH approximation was not very accurate. In this case $\vec{\alpha}^* = (0.8763, 0.8112, 0.4134)^T$, hence $\hat{A}_{new} = 0.8763 A_1 + 0.8112 A_2 + 0.4134 [A_1, A_2]$ as opposed to $\tilde{A}_{new} = 0.5 A_1 + 0.5 A_2 + 0.5 [A_1, A_2]$, given by the CBH formula. The CBH formula expectedly provides a better approximation when $T = 1$ (i.e., for a smaller T), as shown in Figure 7 (b). More specifically, for $T = 2$, $\| x(T) - z_{CBH}(T) \| = 0.3459$, while $\| x(T) - z_{COV}(T) \| = 1.81 \cdot 10^{-5}$. In the case of $T = 1$, $\| x(T) - z_{CBH}(T) \| = 0.0736$, while $\| x(T) - z_{COV}(T) \| = 5.84 \cdot 10^{-4}$. The evolution of $\vec{\alpha}$ in the steepest descent algorithm for both cases ($T = 2, 1$) is shown in Figure 8. Here, $\gamma^{(n)} = 0.05$ for $T = 2$ and



(a) (b)

Figure 7. Comparison of the CBH and COV methods for (a) $T = 2$, (b) $T = 1$.



(a) (b)

Figure 8. The evolution of $\vec{\alpha}$ for (a) T = 2 and (b) T = 1.

$\gamma^{(n)} = 0.2$ for $T = 1$ is the constant step-size for all iterations n, and it should be noted that the algorithm converges quickly. Observe that the calculus of variations result depends on the initial condition x_0 and hence $\vec{\alpha}^*$ will vary as x_0 varies; however, the CBH formula provides global results that are independent of the initial condition x_0.

3.3.2 Robotics Example

In this section, we apply adaptive multi-modal control to the problem of mobile robot navigation. We start by assuming that some preliminary modes have already been designed, namely, "approach-goal" and "avoid-obstacles," and apply the proposed reinforcement learning algorithm to produce the optimal mode string that drives a unicycle through cluttered environment. The performance criterion is to minimize the total distance travelled while minimizing the specification complexity of the control program. Next we show that the performance can be further improved by adding a new mode that is a combination of the existing modes, as outlined in the previous section.

Formally, the dynamics for the unicycle are

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega. \end{aligned} \tag{40}$$

In the system above, (x, y) are the Cartesian coordinates of the center of the unicycle and ϕ is its orientation with respect to the x-axis. Assume that v is constant and ω is the control variable. Given that the system initially has two behaviors, namely, "approach-goal" and "avoid-obstacle," the feedback mappings associated with each behavior are

$$\kappa_g(x, y, \phi) = \omega_g = C_g(\phi_g - \phi), \tag{41}$$

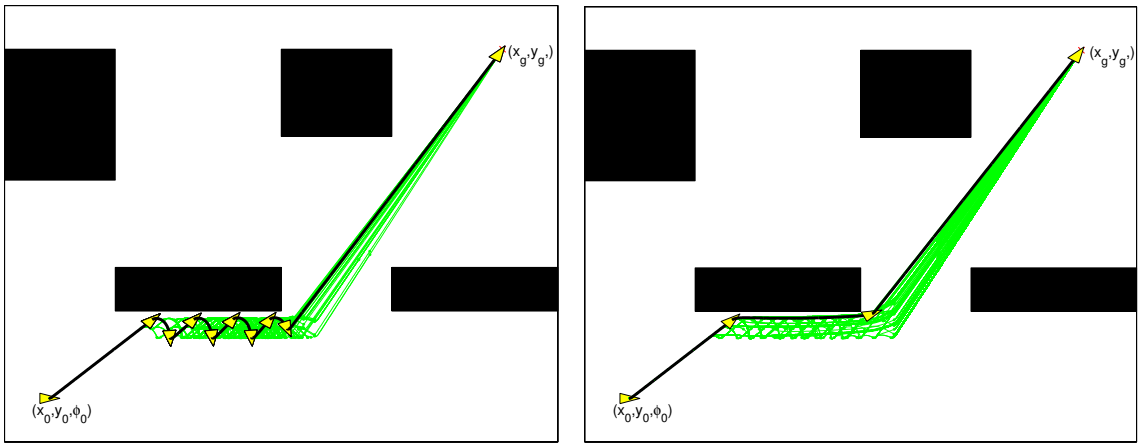
$$\kappa_o(x, y, \phi) = \omega_o = C_o(\pi + \phi_o - \phi). \tag{42}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle, respectively. Both of these angles are measured with respect to the x-axis and can be expressed as

$$\phi_g = \arctan(\frac{y_g - y}{x_g - x}) \text{ and } \phi_o = \arctan(\frac{y_{ob} - y}{x_{ob} - x}), \tag{43}$$

where (x_g, y_g) and (x_{ob}, y_{ob}) are the Cartesian coordinates of the goal and the nearest obstacle, respectively. We also have a set of three interrupts, $\xi_{1,2,3}(x)$, that trigger at three different distances away from the nearest obstacle (x_{ob}, y_{ob}), and all three interrupts always trigger at the goal (x_g, y_g). Hence the total number of available modes is six, i.e., $card(\Sigma) = 6$. The problem then is to plan a path from an initial state (x_0, y_0, ϕ_0) to an open ball around (x_g, y_g) given the set of modes above while minimizing the string length of the control program (i.e., number of switches) along with the total distance travelled.

Given this set of modes, we begin by exploring the reachable space and then performing reinforcement learning to find the optimal path, as described earlier. The resulting optimal path is shown in Figure 9 (a). The optimal control sequence in this case is $\bar{\sigma}^* = (\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_1)(\kappa_g, \xi_1)$. So, clearly, $(\kappa_o, \xi_3)(\kappa_g, \xi_1)$ is repeated often in the control program. Thus, it may be beneficial to replace it with a single mode (κ_n, ξ_1), where we let $\kappa_n = \alpha_g \kappa_g + \alpha_o \kappa_o$. In this case, we let the instantaneous cost $L(x(t), z(t)) = 0.05 \parallel x(t) - z(t) \parallel^2$ and the terminal cost $\psi(x(T), z(T)) = 10 \parallel x(T) - z(T) \parallel^2$. Note by selecting the instantaneous cost in this manner, we indirectly ensure feasibility of the approximation trajectory $z(t)$ since we are penalizing the deviation from the $x(t)$ which is feasible. It is possible to explicitly ensure feasibility by imposing hard constraints on the approximation trajectory; however, this makes the optimal control problem much more difficult. Using the variational techniques presented earlier, it is found that $\alpha_g^* = 0.211$ and $\alpha_o^* = 0.801$. Now we recalculate the optimal path with the new feedback mapping $\kappa_n(x)$ and again the three existing interrupts for its termination added to the mode set. The resulting path is shown in Figure 9 (b) and the optimal control sequence is given by $\tilde{\sigma}^* = (\kappa_g, \xi_1)(\kappa_n, \xi_3)(\kappa_g, \xi_1)$. The augmentation of the motion alphabet results in great improvement in terms of the optimal mode sequence and the resulting optimal trajectory. Although we only designed the new feedback map to "merge"



(a) (b)

Figure 9. The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the (a) original set of modes, (b) augmented set of modes.

two modes, the overall effect of adding the new modes reduced the size of the control program from $|\sigma^*| = 7$ to $|\tilde{\sigma}^*| = 3$. Moreover, the complexity of the control program is reduced from $7 \cdot log_2(6) = 18.0947$ to $3 \cdot log_2(9) = 9.5098$, while the expressiveness is clearly increased since the system can generate more trajectories. This example illustrates the viability of the proposed method.

3.4 Conclusions

In this chapter, we formulated the problem of adaptive multi-modal control. Our approach focused on enhancing the mode set by adding new modes that replace recurring mode string fragments, rather than changing the existing modes. We showed that this mode augmentation can increase the expressiveness of the multi-modal system (thus possibly resulting in improved performance), while reducing the complexity of the control program, assuming such replacement modes can be found. We presented a framework for constructing new modes so that frequently recurring mode combinations can be combined into single "meta-modes." In particular, the "meta-modes" are obtained through a linear combination of the known modes (or any generalizing functions, such as the P. Hall basis). The solution utilizes calculus of variations to

obtain optimality condition, and numerical examples illustrate the usefulness of the presented approach.

CHAPTER 4
A UNIFIED VARIATIONAL FRAMEWORK

In Chapter 3, we motivated the advantages of adaptive multi-modal control and presented a general framework for augmenting the mode set in order to increase performance. In this chapter, we look at several refinements to this initial framework. Although we obtained good performance in certain applications using the initial approach, there may be some room for improvement. In this development, we incorporate the following changes the original framework:

- Why use a linear combination? Can we use a more general feedback law to improve performance?
- Can we utilize the interrupt ξ_{1r} to further improve performance?
- Can we guarantee that σ_{1r} will replace every occurrence of the mode string fragment $\sigma_1 \cdots \sigma_r$ in the mode string $\bar{\sigma}$?

The first point can be addressed by making the new feedback law κ_{1r} more general using weighing functions in the linear combination instead of using scaler weights. In this construction, the feedback law is given by

$$\kappa_{1r} = \sum_{i=1}^{N} \mu_i(x, \alpha_i) g_i(z(t)), \tag{44}$$

where $\mu_i : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$. In (44), the weight of each basis function g_i is determined by a weighing function μ_i, which is parameterized by vector $\alpha_i \in \mathbb{R}^k$. We refer to these weighing functions as *membership functions* as they closely resemble membership functions in fuzzy-logic control [63, 64]. Here the control vector is the concatenation of the shaping vectors α_i for each of the N membership functions, hence $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T \in \mathbb{R}^{Nk}$.

In the original framework, we decided to let $\xi_{1r} = \xi_r$ in order to simplify the problem. We rationalized that since we wanted the approximation function $z(t)$ to be close to $x(t)$, they can both be triggered by the same interrupt. Although this is, in fact, true, it may be possible to further increase performance by utilizing the interrupt function ξ_{1r}. Instead of using the interrupt ξ_r, it may be beneficial to construct ξ_{1r} by incrementally adapting the interrupt ξ_r. We will break up this discussion into two distinct cases: time-driven interrupts and event-driven interrupts. The adjustment in the case of time-driven interrupts is straight-forward, as this involves optimizing switching times. In the case of event-driven interrupts, we will assume that the interrupts are parameterized by some control vector. In particular, we let interrupt ξ_i be shaped by control parameter $\beta_i \in \mathbb{R}^k$, i.e., $\xi_i : \mathbb{R}^n \times \mathbb{R}^k \to \{0, 1\}$. Now we can design ξ_{1r} by adjusting β_r; hence, the optimization problem involves an additional control parameter $\beta \in \mathbb{R}^k$.

Since the original method only optimized over one occurrence of the mode string fragment, we cannot say anything analytically about the approximation of the new mode for other occurrences of the mode string fragment. However, we wish to find σ_{1r} such that every occurrence of mode string fragment $\sigma_1 \cdots \sigma_r$ can be replaced with σ_{1r}. This can be achieved if we define the approximation function $z(t)$ over the entire mode string $\bar{\sigma}_n$, where $\bar{\sigma}_n$ is the new mode string with σ_{1r} replacing every occurrence of $\sigma_1 \cdots \sigma_r$. Now the control vector $\vec{\alpha}$ that minimizes (29) can be found using a variational approach as done earlier, but the new mode designed using $\vec{\alpha}^*$ can readily replace all occurrence of mode string fragment $\bar{\sigma}_{1r}$. Note that this method only provides a local solution, but it will allow us to replace $\sigma_1 \cdots \sigma_r$ globally in the mode string $\bar{\sigma}$. It should be noted, already at this point, that this construction increases the complexity of the optimal control problem significantly since the approximation trajectory has mode switches. Thus, perturbing the control vector will induce a variation in the approximation trajectory as well as the switching time instants.

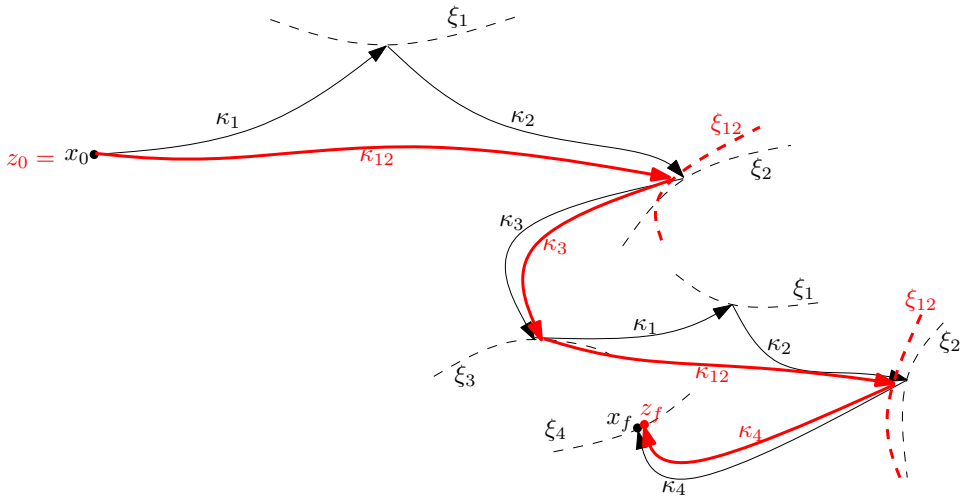


Figure 10. Depicted is the original trajectory $x(t)$ and the approximation trajectory $z(t)$.

In light of the preceding discussion, we propose a new construction for σ_{1r} that incorporates the ideas discussed above. Before detailing this construction, let's look at a specific example to make these informal observations more concrete. Suppose we have a mode string $\bar{\sigma} = \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_4$, and note that $\sigma_1\sigma_2$ is a recurring mode string fragment in $\bar{\sigma}$. Using the initial construction, the new mode is computed by using an approximation function $z(t)$ as shown in Figure 6. With the changes discussed above, the new mode would be constructed using an approximation trajectory $z(t)$ as shown in Figure 10. In particular, the approximation trajectory is given by

$$\dot{z} = \begin{cases} f(z, \kappa_{12}(z)) & \text{until } \xi_{12}(z) = 1 \\ f(z, \kappa_3(z)) & \text{until } \xi_3(z) = 1 \\ f(z, \kappa_{12}(z)) & \text{until } \xi_{12}(z) = 1 \\ f(z, \kappa_4(z)) & \text{until } \xi_4(z) = 1 \end{cases}, \tag{45}$$

with $z(0) = x(0)$. In equation (45) above, κ_{12} is shaped by the control vector $\vec{\alpha}$ as shown in equation (44) and ξ_{12} is ξ_2 reshaped by β_{12}.

With the approximation trajectory defined in this manner, we can cast the mode augmentation problem as an optimal control problem as done earlier. The outline of this chapter is as follows: In Section 4.1, we consider the case of time-driven interrupts.

Thus, the control parameters include the shaping vector $\vec{\alpha}$ for the new feedback law and the temporal interrupts (i.e., switching-times). In Section 4.2, the general case of event-driven interrupts is considered. In this case, the control parameters include the shaping vector $\vec{\alpha}$ for the new feedback law and the shaping vector β_{1r} for the new interrupt. A detailed navigation example illustrating the viability of the proposed methods is presented in Section 4.3, followed by conclusion in Section 4.4.

4.1 Time-Driven Interrupts

As done in Section 3.2, we cast the mode augmentation problem as an optimal control problem using the construction outlined in the previous section for the the case when the interrupts are time-driven. In this derivation, the problem involves finding the optimal set of control parameters $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T \in \mathbb{R}^{Nk}$ to shape the feedback law κ_{1r} and the optimal switching times $\tau_1, \ldots, \tau_M$, which correspond to the temporal interrupts.

It will be advantageous to introduce an identifier $p(i)$, taking values in a finite set, denoting the mode of operation during the time interval $[\tau_{i-1}, \tau_i)$. Now the approximation trajectory $z(t)$ is given by

$$\dot{z}(t) = \begin{cases} f(z, \kappa_{p(1)}(z)) & \text{when } t \in [\tau_0, \tau_1) \\ f(z, \kappa_{p(2)}(z)) & \text{when } t \in [\tau_1, \tau_2) \\ \quad \vdots & \quad \vdots \\ f(z, \kappa_{p(M)}(z)) & \text{when } t \in [\tau_{M-1}, \tau_M] \end{cases}, \tag{46}$$

with $z(0) = x(0)$. Thus for our example, $p(1) = 12$, $p(2) = 3$, and so on. Observe that $f(z, \kappa_{p(i)}(z))$ is a function of z and the control vector $\vec{\alpha}$ when $p(i) = 1r$ and just a function of z otherwise. Thus for ease of notation, we introduce a new indexing function $\tilde{f}_i(z, \vec{\alpha})$ defined as follows:

$$\tilde{f}_i(z, \vec{\alpha}) = \begin{cases} f(z, \sum_{j=1}^{N} \mu_j(z, \alpha_j) g_j(z)) & \text{if } p(i) = 1r \\ f(z, \kappa_{p(i)}) & \text{otherwise.} \end{cases} \tag{47}$$

The optimal control problem, thus, involves finding the control vector $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ and the switching-times vector $\vec{\tau} = [\tau_1, \ldots, \tau_M]^T$ such that the performance index

$$J = \int_{\tau_0}^{\tau_M} L(x(t), z(t))dt + \psi(x_f, z(\tau_M)) \tag{48}$$

is minimized, where x_f is the desired final position. In the performance index above, note that $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $\psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ are required to be twice differentiable in their second argument. Also for what follows, we assume that the basis functions g_i and membership functions μ_i are twice differentiable.

We will derive the necessary conditions for optimality using variational arguments. The unperturbed cost, denoted by $\tilde{J}_0$, is attained by adding the constraint with a co-state $\lambda(t)$ to the cost (48). For ease of notation, we start by defining the Hamiltonian as

$$H_i(x, z, \lambda_i, \vec{\alpha}_i) = L(x, z) + \lambda_i \tilde{f}_i(z, \vec{\alpha}). \tag{49}$$

Now, the augmented (but unaltered from an evaluation point of view) unperturbed cost is given by

$$\tilde{J}_0 = \sum_{i=1}^{M} \int_{\tau_{i-1}}^{\tau_i} \Big[H_i(x, z, \lambda_i, \vec{\alpha}) - \lambda_i \dot{z} \Big] dt + \psi(x_f, z(\tau_M)). \tag{50}$$

Now, we perturb (50) in such a way that $\vec{\alpha} \to \vec{\alpha} + \epsilon \vec{\gamma}_{l_r}$, where $\vec{\gamma}_{l_r} = [0, \ldots, \gamma_{l_r}, \ldots, 0]^T$ (note the $(kl + r)^{th}$ entry is γ_{l_r} and all other entries are 0, i.e., we are perturbing the r^{th} entry of shaping vector α_l) and $\tau_i = \tau_i + \epsilon\theta_i$ for $i = 1, \ldots, M$, and $\epsilon << 1$, then $z \to z + \epsilon\eta$ is the resulting variation in $z(t)$. Note that $\tau_0 = 0$ is assumed fixed. The perturbed cost, denoted by $\tilde{J}_\epsilon$, is given by

$$\begin{aligned} \tilde{J}_\epsilon = \sum_{i=1}^{M} \int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i} \Big[H_i(x, z + \epsilon\eta, \lambda_i, \vec{\alpha} + \epsilon\vec{\gamma}_{l_r}) - \lambda_i \dot{z} - \epsilon\lambda_i \dot{\eta} \Big] dt + \\ + \psi(x_f, (z + \epsilon\eta)(\tau_M + \epsilon\theta_M)). \end{aligned} \tag{51}$$

Using a first order approximation, we get

$$\begin{aligned} \tilde{J}_\epsilon = \sum_{i=1}^{M} \int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i} \Big[H_i(x, z, \lambda_i, \vec{\alpha}) + \frac{\partial H_i}{\partial z}\epsilon\eta + \frac{\partial H_i}{\partial \alpha_{l_r}}\epsilon\gamma_{l_r} - \lambda_i \dot{z} - \epsilon\lambda_i \dot{\eta} \Big] dt + \\ + \Big[\psi(x_f, z) + \frac{\partial \psi}{\partial z}\epsilon[\dot{z}\theta_M + \eta] \Big]_{t=\tau_M}. \end{aligned} \tag{52}$$

Hence, the first order variation in $\tilde{J}$ in the direction of variation in the control parameters is

$$\begin{aligned}\delta\tilde{J} &= \lim_{\epsilon\to 0}\frac{\tilde{J}_\epsilon - \tilde{J}_0}{\epsilon}\\ &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z}\eta + \frac{\partial H_i}{\partial \alpha_{l_r}}\gamma_{l_r} - \lambda_i\dot{\eta}\Big]dt +\\ &\quad + \sum_{i=1}^{M-1}\theta_i\Big[\lambda_{i+1}(\tau_i+)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big)\Big] +\\ &\quad + \Big[\frac{\partial\psi}{\partial z}\tilde{f}_M\theta_M + L(x,z)\theta_M + \frac{\partial\psi}{\partial z}\eta\Big]_{t=\tau_M}. \end{aligned} \tag{53}$$

The integral terms in (53), denoted by $\delta\chi$, can be further reduced by integrating $\lambda_i\dot{\eta}$ by parts to obtain

$$\begin{aligned}\delta\chi &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z} + \dot{\lambda}_i\Big]\eta dt +\\ &\quad + \sum_{i=1}^{M}\gamma_{l_r}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\frac{\partial H}{\partial\alpha_{l_r}}dt - \sum_{i=1}^{M}\Big[\lambda_i\eta\Big]_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}. \end{aligned} \tag{54}$$

Recall that $\theta_0 = 0$ since $\tau_0 = 0$ is fixed, and note that $\eta(0) = 0$ since $z(0) = x(0) = x_0$. Using the fact that $\eta(t)$ is continuous, (54) is reduced to

$$\begin{aligned}\delta\chi &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z} + \dot{\lambda}_i\Big]\eta dt + \sum_{i=1}^{M}\gamma_{l_r}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\frac{\partial\tilde{f}_i}{\partial\alpha_{l_r}}dt -\\ &\quad - \sum_{i=1}^{M-1}\big[\lambda_i(\tau_i-) - \lambda_{i+1}(\tau_i+)\big]\eta(\tau_i) - \lambda_M(\tau_M-)\eta(\tau_M-). \end{aligned} \tag{55}$$

Recall, we want to select the co-state $\lambda(t)$ so that we avoid having to compute variation $\eta(t)$. Substituting $\delta\chi$ back into $\delta\tilde{J}$, we see that we can use single continuous co-state $\lambda(t)$ given by

$$\lambda(\tau_M) = \frac{\partial\psi}{\partial z}, \tag{56}$$

$$\dot{\lambda}(t) = -\frac{\partial H_i}{\partial z},\ \text{when } t\in(\tau_{i-1},\tau_i), \tag{57}$$

$$\lambda(\tau_i-) = \lambda(\tau_i+), for i = 1,\ldots,M-1. \tag{58}$$

With this choice of the co-state $\lambda(t)$, which can be solved by integrating (57) backwards with boundary conditions (56) and (58), we obtain

$$\delta\tilde{J} = \sum_{i=1}^{M} \gamma_{l_r} \int_{\tau_{i-1}}^{\tau_i} \frac{\partial \tilde{f}_i}{\partial \alpha_{l_r}} dt + \sum_{i=1}^{M-1} \theta_i \Big[\lambda(\tau_i) \big(f_i(\tau_i-) - f_{i+1}(\tau_i+) \big) \Big] + \\ + \theta_M \Big[L(x,z) + \frac{\partial \psi}{\partial z} \tilde{f}_M \Big]_{t=\tau_M}. \tag{59}$$

Since the θ_is and α_{l_r}s are independent, the necessary conditions for optimality (i.e., $\delta\tilde{J} = 0$) are

$$\frac{\partial J}{\partial \tau_M} = L(x(\tau_M-), z(\tau_M-)) + \frac{\partial \psi}{\partial z} \tilde{f}_M(\tau_M-) \equiv 0, \tag{60}$$

$$\frac{\partial J}{\partial \tau_i} = \lambda(\tau_i) \big[\tilde{f}_i(\tau_i-) - \tilde{f}_{i+1}(\tau_{i+1}+) \big] \equiv 0 \\ \text{for } i = 1, \ldots, M-1, \text{ and} \tag{61}$$

$$\frac{\partial J}{\partial \alpha_{l_r}} = \sum_{i=1}^{M} \int_{\tau_{i-1}}^{\tau_i} \lambda \frac{\partial \tilde{f}_i}{\partial \alpha_{l_r}} \, dt \equiv 0 \\ \text{for } l = 1, \ldots, N, \text{ and } r = 1, \ldots, k. \tag{62}$$

Relating this back to our original problem formulation, the partial derivative $\frac{\partial \tilde{f}_i}{\partial z}$ in (57) is

$$\frac{\partial \tilde{f}_i}{\partial z} = \begin{cases} \frac{\partial f}{\partial z} + \frac{\partial f}{\partial u} \Big[\sum_{j=1}^{N} \Big[g_j \frac{\partial \mu_j}{\partial z} + \mu_j \frac{\partial g_j}{\partial z} \Big] \Big] & \text{if } p(i) = 1r \\ \frac{\partial f}{\partial z} & \text{otherwise.} \end{cases} \tag{63}$$

The partial derivative with respect to the shaping vector is

$$\frac{\partial \tilde{f}_i}{\partial \alpha_{l_r}} = \frac{\partial f}{\partial u} \frac{\partial \mu_l}{\partial \alpha_{l_r}} g_l \tag{64}$$

if p(i) = 1r, and 0 otherwise. Hence, (62) can be further reduced to

$$\frac{\partial J}{\partial \alpha_{l_r}} = \sum_{\{i \mid p(i)=1r\}} \int_{\tau_{i-1}}^{\tau_i} \lambda \frac{\partial f}{\partial u} \frac{\partial \mu_l}{\partial \alpha_{l_r}} g_l \, dt \equiv 0, \tag{65}$$

for $l = 1, \ldots, N$, and $r = 1, \ldots, k$.

These results are summarized in a theorem below:

Theorem 4.1.1 *Given a function $x(t) \in \mathbb{R}^n$ and a set of twice differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^m$ and $\mu_i : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ for $i = 1, 2, \ldots, N$, with $z(t) \in \mathbb{R}^n$ given by (44) and (46), an extremum to the performance index*

$$J = \int_0^T L(x(t), z(t))dt + \psi(x_f, z(T))$$

is attained when the control vector $\vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T \in \mathbb{R}^{Nk}$ and switching-times vector $\vec{\tau} = [\tau_1, \tau_2, \ldots, \tau_M]$ are chosen as follows:

Euler-Lagrange Equations:

$$\dot{\lambda}(t) = -\frac{\partial H_i}{\partial z} = -\frac{\partial L}{\partial z}(x, z) - \lambda(t)\frac{\partial \tilde{f}_i}{\partial z}(z, \alpha_i) \text{ when } t \in (\tau_{i-1}, \tau_i),$$

Boundary Conditions:

$$\lambda(\tau_M) = \frac{\partial \psi}{\partial z},$$

$$\lambda(\tau_i-) = \lambda(\tau_i+) \text{ for } i = 1, \ldots, M-1,$$

Optimality Conditions:

$$\begin{aligned}
\frac{\partial J}{\partial \tau_M} &= L(x(\tau_M-), z(\tau_M-)) + \frac{\partial \psi}{\partial z}\tilde{f}_M(\tau_M-) \equiv 0 \\
\frac{\partial J}{\partial \tau_i} &= \lambda(\tau_i)\left[\tilde{f}_i(\tau_i-) - \tilde{f}_{i+1}(\tau_{i+1}+)\right] \equiv 0 \\
&\qquad\qquad \text{for } i = 1, \ldots, M-1, \\
\frac{\partial J}{\partial \alpha_{l_r}} &= \sum_{\{i \mid p(i)=1r\}} \int_{\tau_{i-1}}^{\tau_i} \lambda \frac{\partial f}{\partial u}\frac{\partial \mu_l}{\partial \alpha_{l_r}} g_l \, dt \equiv 0 \\
&\qquad\qquad \text{for } l = 1, \ldots, N, \text{ and } r = 1, \ldots, k.
\end{aligned}$$

4.2 Event-Driven Interrupts

In this section, we consider the more general case of event-driven systems. In addition to the usual dependence on the state, we assume that interrupts are parameterized using a control parameter $\beta \in \mathbb{R}^k$, i.e., $\xi : \mathbb{R}^n \times \mathbb{R}^k \to \{0, 1\}$. Using an identifier $p(i)$

again, the approximation trajectory $z(t)$ is given by

$$\dot{z}(t) = \begin{cases} f(z, \kappa_{p(1)}(z)) & \text{until } \xi_{p(1)}(z, \beta_{p(1)}) = 1 \\ \vdots & \vdots \\ f(z, \kappa_{p(M)}(z)) & \text{until } \xi_{p(M)}(z, \beta_{p(M)}) = 1 \end{cases}, \tag{66}$$

with $z(0) = x(0)$. Observe that $f(z, \kappa_{p(i)}(z))$ is a function of z and the control vector $\vec{\alpha}$ when $p(i) = 1r$ and just a function of z otherwise. Thus for ease of notation, as done before, we introduce a new indexing function $\tilde{f}_i(z, \vec{\alpha})$ defined as

$$\tilde{f}_i(z, \vec{\alpha}) = \begin{cases} f(z, \sum_{j=1}^{N} \mu_j(z, \alpha_j) g_j(z)) & \text{if } p(i) = 1r \\ f(z, \kappa_{p(i)}) & \text{otherwise.} \end{cases} \tag{67}$$

Similarly, since we are only reshaping ξ_{1r}, we can treat $\beta_{p(i)}$ as a fixed constant when $p(i) \neq 1r$. Hence, our control parameter for shaping the interrupt is $\vec{\beta} = \beta_{1r} \in \mathbb{R}^k$. Again, for ease of notation, we introduce $\tilde{\xi}_i(z, \vec{\beta})$ defined as follows:

$$\tilde{\xi}_i(z, \vec{\beta}) = \begin{cases} \xi_{1r}(z, \vec{\beta}) & \text{if } p(i) = 1r \\ \xi_{p(i)}(z, \beta_{p(i)}) & \text{otherwise.} \end{cases} \tag{68}$$

Moreover, we assume that $\tau_0 = 0$ is fixed, and the other switching instants are given by the interrupts as

$$\tau_i = \{t > \tau_{i-1} \mid \xi_{p(i)}(z(t), \beta_{p(i)}) = 1\}, \tag{69}$$

for $i = 1, \ldots, M$.

Now the mode augmentation problem is reduced to finding the control vectors $\vec{\alpha}$ and $\vec{\beta}$ such that the cost

$$J = \int_{\tau_0}^{\tau_M} L(x(t), z(t))dt + \psi(x_f, z(\tau_M)) \tag{70}$$

is minimized, where $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $\psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ are twice differentiable in their second argument. As done earlier, we also assume that the basis functions g_i and membership functions μ_i are twice differentiable. In addition to these assumptions,

we must make one more assumption to ensure that the approximation function $z(t)$ does not approach any of the switching surfaces tangentially. Namely, we assume that

$$L_{\tilde{f}_i}\frac{\partial\tilde{\xi}_i}{\partial z} = \frac{\partial\tilde{\xi}_i}{\partial z}\tilde{f}_i \neq 0$$

for $i = 1, \ldots, M$. In other words, we assume that the Lie derivative of $\tilde{\xi}_i$ with respect to z along the flow $\tilde{f}_i$ does not equal 0.

Defining the Hamiltonian as

$$H_i(x, z, \lambda_i, \vec{\alpha}_i) = L(x, z) + \lambda_i\tilde{f}_i(z, \vec{\alpha}), \tag{71}$$

the augmented (but unaltered from an evaluation point of view) unperturbed cost is given by

$$\tilde{J}_0 = \sum_{i=1}^{M}\int_{\tau_{i-1}}^{\tau_i}\Big[H_i(x, z, \lambda_i, \vec{\alpha}) - \lambda_i\dot{z}\Big]dt + \sum_{i=1}^{M}\nu_i\big(\tilde{\xi}_i(z(\tau_i), \vec{\beta})\big) + \psi(x_f, z(\tau_M)). \tag{72}$$

Note here that the continuous co-state $\lambda_i(t)$ corresponds to the constraints of the continuous dynamics during the time interval (τ_{i-1}, τ_i), while the co-state ν corresponds to the discrete switching dynamics at time instant $t = \tau_i$. Now we perturb (72) in such a way that $\vec{\alpha} \to \vec{\alpha} + \epsilon\vec{\gamma}_{l_r}$, where $\vec{\gamma}_{l_r} = [0, \ldots, \gamma_{l_r}, \ldots, 0]^T$ (note the $(kl + r)^{th}$ entry is γ_{l_r} and all other entries are 0, i.e., we are perturbing the r^{th} entry of shaping vector α_l) and $\vec{\beta} = \vec{\beta} + \epsilon\vec{\delta}$. With $\epsilon << 1$, $z \to z + \epsilon\eta$ is the resulting variation in $z(t)$ and $\tau_i \to \tau_i + \epsilon\theta_i$ for $i = 1, \ldots, M$ is the resulting variation in the switching instants. The perturbed cost, denoted by $\tilde{J}_\epsilon$, is given by

$$\begin{aligned}\tilde{J}_\epsilon = \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i}&\Big[H_i(x, z+\epsilon\eta, \lambda_i, \vec{\alpha}+\epsilon\vec{\gamma}_{l_r}) - \lambda_i\dot{z} - \epsilon\lambda_i\dot{\eta}\Big]dt + \\ &+\sum_{i=1}^{M}\nu_i\big(\tilde{\xi}_i((z+\epsilon\eta)(\tau_i+\epsilon\theta_i), \vec{\beta}+\epsilon\vec{\delta})\big) + \\ &+\psi(x(\tau_M+\epsilon\theta_M), (z+\epsilon\eta)(\tau_M+\epsilon\theta_M)).\end{aligned} \tag{73}$$

Using a first order approximation, we get

$$\tilde{J}_\epsilon = \sum_{i=1}^{M} \int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i} \Big[H_i(x,z,\lambda_i,\vec{\alpha}) + \frac{\partial H_i}{\partial z}\epsilon\eta + \frac{\partial H_i}{\partial \alpha_{l_r}}\epsilon\gamma_{l_r} - \lambda_i\dot{z} - \epsilon\lambda_i\dot{\eta}\Big]dt + \\ + \sum_{i=1}^{M} \nu_i\Big[\tilde{\xi}_i(z,\vec{\beta}) + \frac{\partial\tilde{\xi}_i}{\partial z}\dot{z}\epsilon\theta_i + \frac{\partial\tilde{\xi}_i}{\partial z}\epsilon\eta + \frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}}\epsilon\delta\Big]_{t=\tau_i} + \\ + \Big[\psi(x_f,z) + \frac{\partial\psi}{\partial z}\epsilon[\dot{z}\theta_M + \eta]\Big]_{t=\tau_M}. \tag{74}$$

Hence, the first order variation in $\tilde{J}$ is

$$\begin{aligned}\delta\tilde{J} &= \lim_{\epsilon\to 0}\frac{\tilde{J}_\epsilon - \tilde{J}_0}{\epsilon} \\ &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z}\eta + \frac{\partial H_i}{\partial\alpha_{l_r}}\gamma_{l_r} - \lambda_i\dot{\eta}\Big]dt + \\ &\quad + \sum_{i=1}^{M-1}\theta_i\Big[\lambda_{i+1}(\tau_i+)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big)\Big] + \\ &\quad + \sum_{i=1}^{M}\nu_i\Big[\frac{\partial\tilde{\xi}_i}{\partial z}\tilde{f}_i\theta_i + \frac{\partial\tilde{\xi}_i}{\partial z}\eta + \frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}}\delta\Big]_{t=\tau_i} + \\ &\quad + \Big[L(x,z)\theta_M + \frac{\partial\psi}{\partial z}\tilde{f}_{M+1}\theta_M + \frac{\partial\psi}{\partial z}\eta\Big]_{t=\tau_M}.\end{aligned} \tag{75}$$

The integral terms in (75), denoted by $\delta\chi$, can be further reduced by integrating $\lambda_i\dot{\eta}$ by parts to obtain

$$\begin{aligned}\delta\chi &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z} + \dot{\lambda}_i\Big]\eta dt + \\ &\quad + \sum_{i=1}^{M}\gamma_{l_r}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\frac{\partial H}{\partial\alpha_{l_r}}dt - \sum_{i=1}^{M}\Big[\lambda_i\eta\Big]_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}.\end{aligned} \tag{76}$$

Note that $\theta_0 = 0$ since $\tau_0 = 0$ is fixed, and $\eta(0) = 0$ since $z(0) = x(0) = x_0$. Using the fact that $\eta(t)$ is continuous, (76) is reduced to

$$\begin{aligned}\delta\chi &= \sum_{i=1}^{M}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial z} + \dot{\lambda}_i\Big]\eta dt + \sum_{i=1}^{M}\gamma_{l_r}\int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i}\frac{\partial\tilde{f}_i}{\partial\alpha_{l_r}}dt - \\ &\quad - \sum_{i=1}^{M-1}\big[\lambda_i(\tau_i-) - \lambda_{i+1}(\tau_i+)\big]\eta(\tau_i) - \lambda_M(\tau_M-)\eta(\tau_M-).\end{aligned} \tag{77}$$

Substituting $\delta\chi$ back into $\delta\tilde{J}$, we want to select the co-states λ_i and ν_i (for $i = 1, \ldots, M$) so that we avoid having to calculate the variations η and θ_i (for $i = 1, \ldots, M$). For the evolution of $\lambda(t)$, we get the expected (standard) result:

$$\dot{\lambda}_i = -\frac{\partial H_i}{\partial z} = -\frac{\partial L}{\partial z} - \lambda_i \frac{\partial \tilde{f}_i}{\partial z}. \tag{78}$$

The boundary conditions are, however, very different. It turns out that the co-state $\lambda(t)$ is discontinuous at the switching instants. To see this, lets first look at the variation $\eta(\tau_M-)$:

$$\begin{aligned} \eta(\tau_M-)\Big[-\lambda_M(\tau_M-) + \nu_M \frac{\partial \tilde{\xi}_M}{\partial z}(\tau_M-) + \frac{\partial \psi}{\partial z}(\tau_M-)\Big] &\equiv 0 \\ \implies \lambda_M(\tau_M) = \frac{\partial \psi}{\partial z}(\tau_M-) + \nu_M \frac{\partial \tilde{\xi}}{\partial z}(\tau_M-). & \end{aligned} \tag{79}$$

Similarly looking at the variation $\eta(\tau_i-)$ for $i = 1, \ldots, M-1$, the boundary conditions at the switching instants τ_i are

$$\lambda_i(\tau_i-) = \lambda_{i+1}(\tau_i+) + \nu_i \frac{\partial \tilde{\xi}_i}{\partial z}(\tau_i-) \tag{80}$$

for $i = 1, \ldots, M-1$.

Using Equations (78)-(80), $\lambda(t)$ can be solved by integrating backwards in time. However, the boundary conditions at τ_i depend on the costate ν_i. To see how we should select ν_i, let's first examine the variation θ_i for $i = 1, \ldots, M-1$:

$$\begin{aligned} \theta_i\Big[\lambda_{i+1}(\tau_i+)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big) + \nu_i \frac{\partial \tilde{\xi}_i}{\partial z}\tilde{f}_i(\tau_i-)\Big] &\equiv 0 \\ \implies \nu_i = -\frac{\lambda_{i+1}(\tau_i+)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big)}{L_{\tilde{f}_i}\frac{\partial \tilde{\xi}_i}{\partial z}(\tau_i-)}, & \end{aligned} \tag{81}$$

where $L_{\tilde{f}_i}\frac{\partial \tilde{\xi}_i}{\partial z} = \frac{\partial \tilde{\xi}_i}{\partial z}\tilde{f}_i$ denotes the Lie derivative of $\tilde{\xi}_i$ with respect to z along the flow $\tilde{f}_i$. Similarly looking at θ_M, we select ν_M as

$$\nu_M = -\Big[\frac{L(x_f, z) + L_{\tilde{f}_M}\frac{\partial \psi}{\partial z}}{L_{\tilde{f}_M}\frac{\partial \tilde{\xi}_M}{\partial z}}\Big]_{t=\tau_M-}. \tag{82}$$

Note that all of the expressions derived above are well defined as long as $L_{\tilde{f}_i}\frac{\partial\tilde{\xi}_i}{\partial z}(\tau_i) \neq 0$ for $i = 1, \ldots, M$. Recall that this condition simply means that the trajectory of z does not approach the interrupt (or switching) surface tangentially.

With this choice of the co-states, (75) is reduced to

$$\delta\tilde{J} = \sum_{i=1}^{M} \gamma_{l_r} \int_{\tau_{i-1}}^{\tau_i} \frac{\partial\tilde{f}_i}{\partial\alpha_{l_r}} dt + \sum_{i=1}^{M} \nu_i \frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}}\vec{\delta}. \tag{83}$$

Since α_{l_r} (for $l = 1, \ldots, N$ and $r = 1, \ldots, k$) and $\vec{\beta}$ are independent, the necessary conditions are optimality (i.e., $\delta J = 0$) are

$$\frac{\partial J}{\partial\vec{\beta}} = \sum_{i=1}^{M} \nu_i \frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}}(\tau_i-) \equiv 0, \text{ and} \tag{84}$$

$$\frac{\partial J}{\partial\alpha_{l_r}} = \sum_{i=1}^{M} \int_{\tau_{i-1}}^{\tau_i} \lambda_i \frac{\partial\tilde{f}_i}{\partial\alpha_{l_r}}\, dt \equiv 0$$
$$\text{for } l = 1, \ldots, N, \text{ and } r = 1, \ldots, k. \tag{85}$$

Relating this back to our original problem formulation, the partial derivative $\frac{\partial\tilde{f}_i}{\partial z}$ is

$$\frac{\partial\tilde{f}_i}{\partial z} = \begin{cases} \frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\Big[\sum_{j=1}^{N}\left[g_j\frac{\partial\mu_j}{\partial z} + \mu_j\frac{\partial g_j}{\partial z}\right]\Big] & \text{if } p(i) = 1r \\ \frac{\partial f}{\partial z} & \text{otherwise.} \end{cases} \tag{86}$$

The partial derivative of $\tilde{f}_i$ with respect to the shaping vector is

$$\frac{\partial\tilde{f}_i}{\partial\alpha_{l_r}} = \frac{\partial f}{\partial u}\frac{\partial\mu_l}{\partial\alpha_{l_r}}g_l \tag{87}$$

if p(i) = 1r, and 0 otherwise. Also, the partial derivative of $\tilde{\xi}$ are

$$\frac{\partial\tilde{\xi}_i}{\partial z} = \frac{\partial\xi_{p(i)}}{\partial z}, \tag{88}$$

$$\frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}} = \frac{\partial\xi_{1r}}{\partial\beta_{1r}} \text{ if } p(i) = 1r, \text{ and} \tag{89}$$

$$\frac{\partial\tilde{\xi}_i}{\partial\vec{\beta}} = 0 \text{ if } p(i) \neq 1r. \tag{90}$$

Hence, the necessary conditions for optimality can be further reduced to

$$\frac{\partial J}{\partial \vec{\beta}} = \sum_{\{i \mid p(i)=1r\}} \nu_i \frac{\partial \xi_{1r}}{\partial \beta_{1r}}(\tau_i-) \equiv 0, \text{ and} \tag{91}$$

$$\frac{\partial J}{\partial \alpha_{l_r}} = \sum_{\{i \mid p(i)=1r\}} \int_{\tau_{i-1}}^{\tau_i} \lambda_i \frac{\partial f}{\partial u}\frac{\partial \mu_l}{\partial \alpha_{l_r}} g_l \, dt \equiv 0,$$
$$\text{for } l = 1, \ldots, N, \text{ and } r = 1, \ldots, k. \tag{92}$$

These results are summarized in a theorem below:

Theorem 4.2.1 *Given a function $x(t) \in \mathbb{R}^n$ and a set of twice differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^m$ and $\mu_i : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ for $i = 1, 2, \ldots, N$, with $z(t) \in \mathbb{R}^n$ given by (44) and (66), an extremum to the performance index*

$$J = \int_0^T L(x(t), z(t))dt + \psi(x_f, z(T))$$

is attained when the control vectors $\vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T \in \mathbb{R}^{Nk}$ and $\vec{\beta} = \in \mathbb{R}^k$ are chosen as follows:

Euler-Lagrange Equations:

$$\dot{\lambda}(t) = -\frac{\partial H_i}{\partial z} = -\frac{\partial L}{\partial z}(x, z) - \lambda(t)\frac{\partial \tilde{f}_i}{\partial z}(z, \alpha_i) \text{ when } t \in (\tau_{i-1}, \tau_i),$$
$$\nu_i = -\frac{\lambda_{i+1}(\tau_i+)\big(f_i(\tau_i-) - f_{i+1}(\tau_i+)\big)}{L_{\tilde{f}_i}\frac{\partial \tilde{\xi}_i}{\partial z}(\tau_i-)} \text{ for } i = 1, \ldots, M-1,$$
$$\nu_M = -\Big[\frac{L(x_f, z) + L_{\tilde{f}_M}\frac{\partial \psi}{\partial z}}{L_{\tilde{f}_M}\frac{\partial \tilde{\xi}_M}{\partial z}}\Big]_{t=\tau_M-},$$

Boundary Conditions:

$$\lambda_M(\tau_M) = \frac{\partial \psi}{\partial z}(\tau_M-) + \nu_M \frac{\partial \tilde{\xi}}{\partial z}(\tau_M-),$$
$$\lambda_i(\tau_i-) = \lambda_{i+1}(\tau_i+) + \nu_i \frac{\partial \tilde{\xi}_i}{\partial z}(\tau_i-) \text{ for } i = 1, \ldots, M-1,$$

Optimality Conditions:

$$\frac{\partial J}{\partial \vec{\beta}} = \sum_{\{i \mid p(i)=1r\}} \nu_i \frac{\partial \xi_{1r}}{\partial \beta_{1r}}(\tau_i-) \equiv 0 \text{ and}$$
$$\frac{\partial J}{\partial \alpha_{l_r}} = \sum_{\{i \mid p(i)=1r\}} \int_{\tau_{i-1}}^{\tau_i} \lambda_i \frac{\partial f}{\partial u}\frac{\partial \mu_l}{\partial \alpha_{l_r}} g_l \, dt \equiv 0$$
$$\text{for } l = 1, \ldots, N, \text{ and } r = 1, \ldots, k.$$

4.3 Applications

4.3.1 Mobile Robot Navigation

In this section, we consider the problem of navigating a unicycle, as discussed in Section 3.3.2. We will compare the navigation strategies derived using the adaptive multi-modal control framework presented in Section 3.2 to the strategies derived using the methods presented in this chapter. For convenience, we will reintroduce the model and control laws presented earlier.

The kinematic model of a unicycle is

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega. \end{aligned} \tag{93}$$

In the system above, (x, y) are the Cartesian coordinates of the center of the unicycle and ϕ is its orientation with respect to the x-axis. Assume that v is constant and ω is the control variable. Given that the system initially has two behaviors, namely, "go-to-goal" and "avoid-obstacle," the feedback mappings associated with each behavior are as follows:

$$\kappa_g(x, y, \phi) = \omega_g = C_g(\phi_g - \phi), \tag{94}$$

$$\kappa_o(x, y, \phi) = \omega_o = C_o(\pi + \phi_o - \phi). \tag{95}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle, respectively. Both of these angles are measured with respect to the x-axis and can be expressed as

$$\phi_g = \arctan(\frac{y_g - y}{x_g - x}) \text{ and } \phi_o = \arctan(\frac{y_{ob} - y}{x_{ob} - x}),$$

where (x_g, y_g) and (x_{ob}, y_{ob}) are the Cartesian coordinates of the goal and the nearest obstacle, respectively. We also have a set of three interrupts, $\xi_{1,2,3}(x)$, that trigger at three different distances away from the nearest obstacle (x_{ob}, y_{ob}), and all three

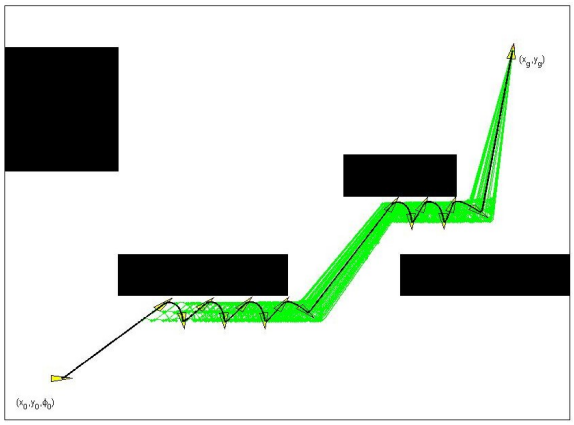


Figure 11. The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the original set of modes.

interrupts always trigger at the goal (x_g, y_g). Hence the total number of available modes is six, i.e., $card(\Sigma) = 6$. The problem then is to plan a path from an initial state (x_0, y_0, ϕ_0) to an open ball around (x_g, y_g) given the set of modes above while minimizing the performance criterion

$$J = J_{spec} + J_{path} = |\bar{\sigma}| \log_2(card(\Sigma)) + \int_0^T v^2 dt + ||\vec{x}_g - \vec{x}(T)||^2.$$

Here T is the time that $\bar{\sigma}$ terminates, $\vec{x} = (x, y)^T$, so J_{path} is the length of the robot trajectory plus a final terminal cost penalizing deviation from the desired goal.

Given this set of modes, we begin by exploring the reachable space using RRTs as outlined in Section 2.2. The simulation environment, the estimated reachable set, and the optimal path are shown in Figure 11. The optimal control sequence is

$$\bar{\sigma}^* = (\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_2)$$
$$(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1).$$

The specification complexity of this control program is $15 \log_2(6) = 38.77$ bits. Thus the total cost $J = 38.77 + 83.07 = 121.84$.

Looking at the control program $\bar{\sigma}$, we see that mode string fragment $(\kappa_o, \xi_3)(\kappa_g, \xi_1)$ is repeated often in the optimal control program; hence, it may be beneficial to replace it with a single mode (κ_n, ξ_1), where we let $\kappa_n = \alpha_g \kappa_g + \alpha_o \kappa_o$. Using the

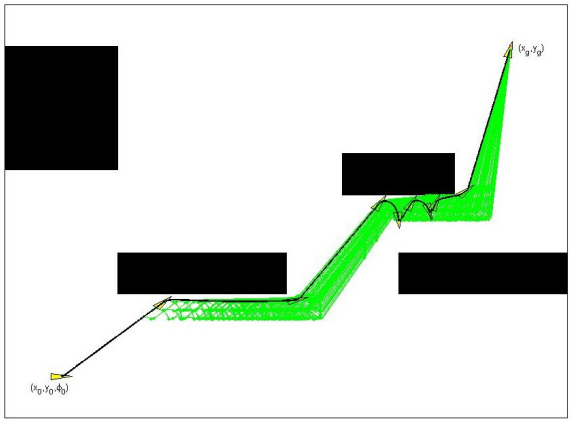


Figure 12. The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the augmented set of modes using method outlined in Chapter 3.

framework presented in Section 3.2, we calculate the optimal weights to replace the first occurrence of this mode string fragment. Again, we let the instantaneous cost $L(x(t), z(t)) = 0.05 \parallel x(t) - z(t) \parallel^2$ and the terminal cost $\psi(x(T), z(T)) = 10 \parallel x(T) - z(T) \parallel^2$.It is found that $\alpha_g^* = 0.21$ and $\alpha_o^* = 0.80$. Now, we recalculate the optimal path with the new feedback mapping $\kappa_n(x)$ and the existing interrupt ξ_1 for its termination added to the mode set. The resulting path is shown in Figure 12, and the optimal control sequence is

$$\bar{\sigma}_{n1}^* = (\kappa_g, \xi_1)(\kappa_n, \xi_1)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_2)(\kappa_g, \xi_1)(\kappa_n, \xi_1)(\kappa_g, \xi_1).$$

The specification complexity in this case is $9log_2(7) = 25.27$ bits, and the total cost is reduced to $J = 25.27 + 68.60 = 93.87$. We see that the new mode (κ_n, ξ_1) readily replaces the first few occurrences of the mode string fragment $(\kappa_o, \xi_3)(\kappa_g, \xi_1)$ in the mode string. However, the new mode is not very affective in circumventing the obstacle closer to the goal (i.e., the new mode does not effectively replace later occurrences of the mode string fragment). It seems that there is still room for improvement here.

Indeed, using the methods outlined in this chapter, we should be able to get a better approximation for all occurrences of the mode string fragment. Moreover, the guard shaping technique will allow us to tune the interrupt to further improve

Table 3. Cost comparison between the optimal mode string generated using the original mode set (Σ), the augmented mode set using approach outlined in Chapter 3 (Σ_{n1}), and the augmented mode set using the methods outlined in this chapter (Σ_{n2}).

Mode Set	Optimal Control Program	J_{spec}	J
Σ	$\bar{\sigma}^*$	38.77	121.84
Σ_{n1}	$\bar{\sigma}^*_{n1}$	25.27	93.87
Σ_{n2}	$\bar{\sigma}^*_{n2}$	14.04	82.23

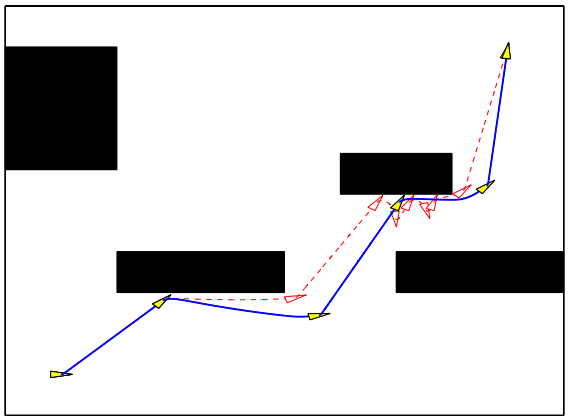


Figure 13. The optimal path to drive a unicycle from x_0 to x_g using the augmented set of modes using method outlined in Chapter 3 (dashes) and the method outlined in this chapter (thick).

performance. We still use the same construction for the new feedback law, i.e., $\kappa_n = \alpha_g\kappa_g + \alpha_o\kappa_o$. Now, we define a new interrupt function $\xi_n(z,\beta) = ||z - z_{obs}||^2 - \beta$. Using the methods presented in this paper, we find that $\alpha_g^* = 0.14$, $\alpha_o^* = 0.81$, and $\beta^* = 3.1$. Using this new augmented set of modes, the optimal path is shown in Figure 13. The optimal control sequence is

$$\bar{\sigma}^*_{n2} = (\kappa_g, \xi_1)(\kappa_n, \xi_n)(\kappa_g, \xi_1)(\kappa_n, \xi_n)(\kappa_g, \xi_1).$$

Looking at the figure, we see the improvement from using the unified framework presented in this chapter. The new mode obtained using this new framework provides a good approximation for all occurrences of the mode fragment $(\kappa_o, \xi_3)(\kappa_g, \xi_1)$ in $\bar{\sigma}$. The specification complexity of the new mode string $\bar{\sigma}^*_{n2}$ is $5log_2(7) = 14.04$ bits, and the total cost is further reduced to $J = 14.04 + 68.19 = 82.23$. These results are further summarized in Table 3.

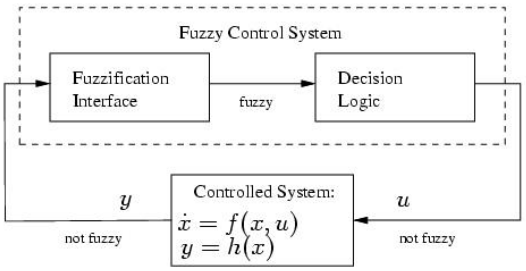


Figure 14. Architecture of a Takagi and Sugeno fuzzy controller.

4.3.2 Optimal Membership Functions

In the robotics application discussed in the previous section, we employed a multi-modal (or hybrid) control strategy, meaning that the controller switches between different modes of control. An alternative to this approach would be to let the different controllers (i.e., control modes) run concurrently, where the overall control output to the robot is some combination of these controllers. This approach is quite common in fuzzy-logic control, where different controllers are combined using membership functions [64]. Although there are many different types of fuzzy controllers, we will focus on the popular Takagi and Sugeno (TS) fuzzy controller [63]. In the TS approach, the inputs are specified as fuzzy sets (fuzzification) as done in the standard fuzzy-logic control approach, but the decision logic provides a crisp output so there is no defuzzification process. The architecture of a TS fuzzy controller is shown in Figure 14. There is typically an ensemble of control laws, i.e., $u = \kappa_i(y)$ for $i = 1, \ldots, N$, whose members become applicable based on the classification of the output y of the system. The task of the decision logic is to determine the degree of applicability μ_i to each of the control laws based on fuzzy sets or membership functions, which are typically triangular or piecewise continuous functions. Of course, it is also possible to use other types of membership functions. The final control value is then computed as

$$u = \frac{\sum_{i=1}^{N} \mu_i(y)\kappa_i(y)}{\sum_{i=1}^{N} \mu_i(y)}. \tag{96}$$

The advantage of using this approach is that it results in smooth overall performance since there are no hard switches, but it is often hard to say anything analytically about the optimality of such systems. The success of fuzzy control systems depends on a number of parameters including the fuzzy membership function, yet they are often selected subjectively and then tuned manually to improve performance. Some work has been done on tuning these parameters using genetic algorithms as well as other methods [65, 66, 67]. We will show that these fuzzy sets can be optimized using the variational approach presented earlier. This method relies on the dynamics of the controlled system, and requires that the state transition functions and the membership functions are twice differentiable. In particular, we will use this technique to derive optimal membership functions for a fuzzy-logic control based navigation strategy.

We will derive a fuzzy-logic controller with optimal membership functions to control a unicycle with two control laws that correspond to the "go-to-goal" and "avoid-obstacles" behavior. The dynamics of the unicycle and the behaviors are defined in the previous section (refer to Equations (93)-(95)). Thus, the overall control output w of the fuzzy controller is given by

$$w = \frac{\mu_g(x)\omega_g + \mu_o(x)\omega_o}{\mu_g(x) + \mu_o(x)}. \tag{97}$$

The two control laws are weighted by μ_g and μ_o computed according to the corresponding membership function.

Now that the individual control laws are defined, we must next decide on the appropriate membership functions, which determine the degree of applicability of each control law. This is typically done by using piecewise linear or triangular membership functions as shown in Figure 15, but we need these functions to be twice differentiable. Thus, we specify them using exponential functions (e.g., $e^{-\alpha\|x-x_o\|^2}$), and optimize them with respect to the tuning parameter α. In particular, the membership functions

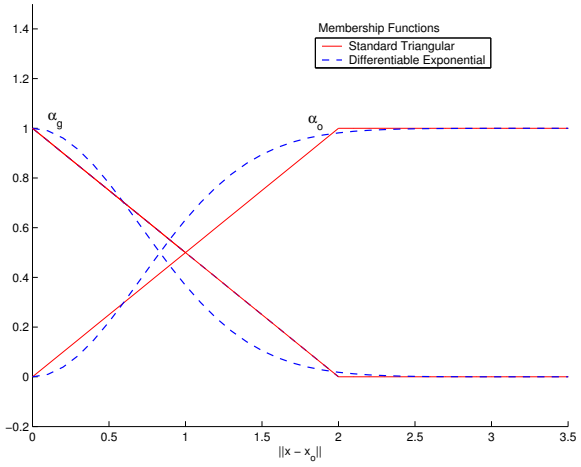


Figure 15. The standard piecewise linear or triangular (solid) and the differentiable exponential (dashed) membership functions.

are defined as

$$\mu_g(x, \alpha_g) = 1 - e^{-\alpha_g \|x - x_o\|^2} \text{ and } \mu_o(x, \alpha_o) = e^{-\alpha_o \|x - x_o\|^2}. \tag{98}$$

Note that by defining the membership functions using exponentials, we can get functions that look similar to the standard triangular function while ensuring differentiability.

Having defined the membership functions in this manner, we can use the variational solution derived earlier to select the tuning parameters α so that the performance index

$$J = \int_{\tau_0}^{\tau_M} L(x(t), z(t))dt + \psi(x_f, z(\tau_M)) \tag{99}$$

is minimized. For our particular problem, we let

$$L(x(t), z(t)) = ae^{-b\|z - z_0\|^2} + c \parallel z - z_g \parallel^2, \tag{100}$$

where $z = [x, y]^T$ and $\dot{z}$ is given by (93) and (97). Also, let $\psi(x(T), z(T)) = 0$ since the control (97) ensures that the unicycle will reach the goal given that the goal is sufficiently far away from an obstacle. The result of the gradient descent algorithm is shown in Figure 16. For the simulation, $z_0 = [-1.5, 0]^T$, $\phi_0 = 0$, $z_o = [0, 0]^T$,

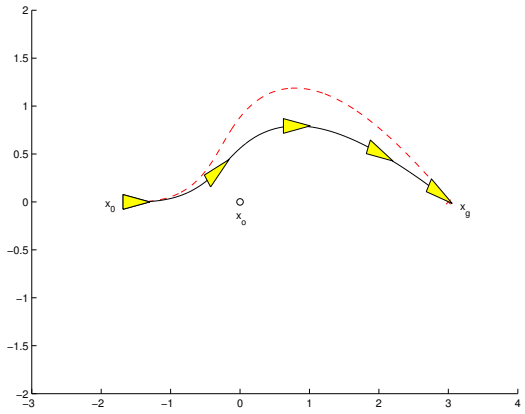
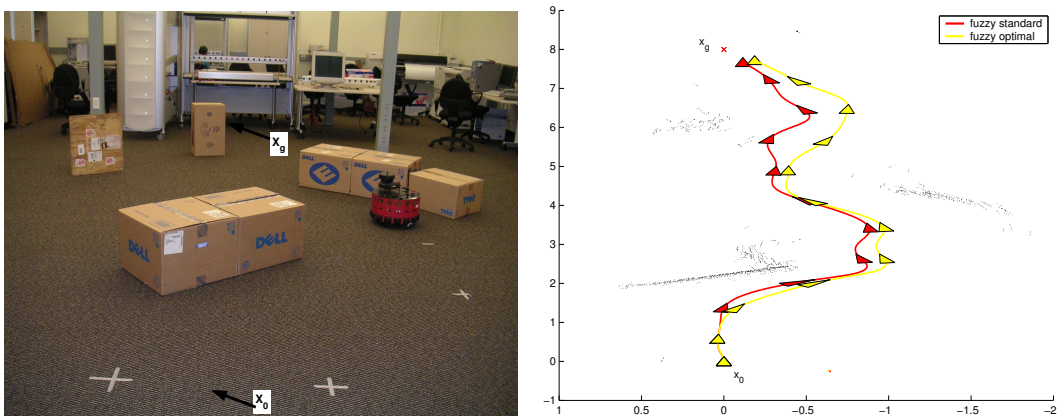


Figure 16. Optimization results: the trajectory for the initial guess of α (dashed) along with the final optimal trajectory.

$z_g = [3, 0]^T$, $a = 2$, $b = 10$, and $c = 0.01$. The algorithm converges to $\alpha_g^* = 5.5584$ and $\alpha_o^* = 1.2068$.

With the optimal membership functions designed for this known environment, the resulting fuzzy logic control strategy can easily be transitioned onto a real robotic platform navigating in an unknown environment. Note that since the optimization is performed over a well-defined environment, the resulting navigation strategy will no longer be optimal in an unknown environment but rather corresponds to a sub-optimal performance enhancing strategy. To illustrate this point, we compare the performance of the optimal fuzzy controller to the standard fuzzy controller with triangular functions on the Magellan Pro platform with the setup shown in Figure 17. The resulting trajectories are plotted using the odometry and sensor readings from the robot. The standard fuzzy controller resulted in a trajectory with cost $J_{std} = 8.8603$, while the optimized fuzzy controller lowered the cost to $J_{opt} = 7.7641$. These results show that the approach presented in this paper offers a novel systematic approach for fine-tuning fuzzy controllers by optimizing the corresponding membership functions, thus resulting in improved performance.



(a) (b)

Figure 17. (a) The experimental setup, (b) The resulting trajectories plotted using the odometry readings, while the obstacle are inferred from the sensor readings.

4.4 Conclusions

In this chapter, we continued the development on adaptive multi-modal control started in Chapter 3, where the utility of adaptive multi-modal control was presented. The strength of our method lies in constructing new modes that can effectively replace many occurrences of mode string fragments. In this chapter, we completed the effort begun in the Chapter 3 by developing a unified variational framework for replacing recurring mode string fragments with new "meta-modes." The construction of the new modes involves designing a feedback control law, which is designed as a combination of previously established control laws, and the design of an interrupt. We explicitly addressed the design of the interrupt for both time-driven and event-driven systems, and designed the new interrupts by incrementally adapting previously established interrupts. In particular, the problem was cast as an optimal control problem and solved using variational arguments. Moreover, the viability of the presented methods is illustrated through a detailed navigation example. It should be noted that since we are using variational arguments, the results found in this chapter are local results. However, the idea here is that these locally optimal replacements modes make the multi-modal system more expressive, which promises better overall performance.

CHAPTER 5
LEARNING FROM EXAMPLE

Consider a situation in which a human operator is driving a robot to a specified goal location through an unknown environment. One would typically expect the human operator to try to find safe paths to the goal while avoiding hazardous regions. Moreover, it is conceivable that the robot (through human control) reacts to distinctive features in the environment in a particular way, such as "stay at least 1 meter away from the wall." A natural objective is to have the robot mimic the actions of the human operator in a completely autonomous manner. However, the problem is not to simply store the path that the human-operated robot took and then reproduce it, but rather to *learn* at a behavioral level the control laws, i.e., closed-loop mappings from sensory input data to control signals, needed to reproduce this motion. We will refer to this problem as the *Learning From Example* problem, and variants of it have received considerable attention in the robotics community [68, 69, 70, 71, 73, 74, 75].

Many different strategies for the Learning From Example problem have emerged over the years. Generally speaking, these strategies can be placed in two major research camps. One camp approaches the problem from a perception point of view and attempts to learn the relevant features (e.g., paths, walls, etc). Then these features are classified as traversable or non-traversable based on the learning cues from the observed behavior. In other words, what is learned is a feature classification that can serve as guidance for the robot to plan through the terrain [70, 71, 76]. We will refer to this as the *perception-centric* approach. This approach can complemented with the control-based view, where behaviors that closely resemble the motion demonstrated by the human operator are learned. Typically, the relevant features are assumed to be *a priori* known, and what is learned is a policy mapping features to control signals [77, 78, 79]. This *control-centric* approach will be taken in this development.

A common feature among previous control-centric methods is that they try to learn behavioral mappings using a "blank-slate" view, i.e., they attempt to learn these mappings without reference to previously established capabilities [11, 68, 80, 81, 82]. However, a natural modification of this approach would consist of a systematic improvement of the existing capabilities. Consider for example the scenario of learning to ride a motorcycle. Assuming we already know how to ride a bicycle, we will not completely throw out this knowledge when learning to ride a motorcycle. In fact, we use our experience from riding a bicycle to leverage the learning of riding a motorcycle. Indeed, *constructivism* views learning as a process in which the learner actively constructs or builds new ideas or concepts based upon current and past knowledge [83, 84].

We take this point-of-view throughout our development and approach the Learning From Example problem by constructing new behaviors from previously established ones. In particular, we assume that we have access to a set of behaviors (possibly of limited expressiveness). It is natural to consider such previous capabilities as designed with a particular task in mind such as "avoiding-obstacles" or "following-wall." However, no such interpretation is necessary.

Within this context, constructivist learning can be viewed as learning new behaviors as some function of the known behaviors, where the learning is guided by training examples. In particular, given a training trajectory, the learning task consists of finding an appropriate sequence of new behaviors in order to approximate the training trajectory. It should already be apparent that this view is consistent with the approach we took in adaptive multi-modal control (Chapter 3), where new modes were introduced as functions of existing modes. However, instead constructing a single "meta-mode" to replace recurring mode string fragments, here we are interested in finding a sequence of new behaviors. Again, the new behaviors will be defined as

linear combinations of the existing behaviors, as done in adaptive multi-modal control. Thus, the main task involves finding the weights of each existing behavior in the linear combination, and in determining how many such new behavioral combinations are required to approximate the training trajectory. We will let the transitions between new behaviors be temporally driven, even though event-driven transitions may be better suited for mobile robot navigation. But, as this work represents a initial study of constructivist learning from an optimal control point of view, we leave this issue to future endeavors. Since the switching between new learned behaviors will be temporally driven, the problem involves optimizing over the switching instants and the individual behavioral weights, as was the case in Section 4.1.

The outline of the Chapter is as follows: In Section 5.1, we formalize the Learning From Example problem as an optimal control problem. In Section 5.2, we utilize variational arguments to solve this problem and present some examples. In Section 5.3, we look at a more specific example of the DARPA [1] sponsored Learning Applied to Ground Robots (LAGR) project. Here, the main problem is to incorporate the learning process seamlessly within the existing system architecture. The existing architecture does not allow the use of the variational techniques; thus, we discuss an alternative solution to the optimal control problem posed in Section 5.1. Finally, the conclusions are presented in Section 5.4.

5.1 Problem Formulation

We start by formally introducing the Learning From Example problem from a constructivist point of view and casting it as an optimal control problem. More specifically, we start by assuming a prior collection of behaviors and try to approximate the training trajectory from a combination of these existing behaviors.

Formally, let X denote the state space, Y denote the observation space, and U be

[1] DARPA - Defense Advanced Research Projects Agency.

the control space. Suppose the system dynamics are

$$\dot{x}(t) = f(x(t), u(t)), \tag{101}$$

$$y(t) = h(x(t)). \tag{102}$$

A behavior is a mapping from observations to control values, i.e., $\kappa : Y \to U$. Hence, if we drive the robot according to behavior κ_1 until time τ_1, κ_2 until time τ_2, and so on, the evolution of the system is

$$\dot{x} = \begin{cases} f(x, \kappa_1(y)) & \text{when } t \in [\tau_0, \tau_1) \\ f(x, \kappa_2(y)) & \text{when } t \in [\tau_1, \tau_2) \\ \vdots & \vdots \end{cases} \tag{103}$$

Note that an event-driven version of this model can be defined, where the switching times are driven by interrupts $\xi_i : Y \to \{0, 1\}$. In this case, the switching time τ_i would be given by

$$\tau_i = \min_{t > \tau_{i-1}} \{t \ : \ \xi_i(y(t)) = 1\}. \tag{104}$$

Observe that this is a generalization of the motion description language framework presented in Section 1.2.1, where state observation was assumed.

Now, suppose we have a collection of behaviors $K = \{\kappa_1, \kappa_2, \ldots, \kappa_N\}$. In the constructivist framework presented here, the new learned behavior will be defined through a combination of the behaviors in K. One option is to let the new behavior κ_n be given as a linear combination of the existing behaviors, i.e.,

$$\kappa_n(y) = \sum_{i=1}^{N} \alpha_i \kappa_i(y),$$

where α_i is a scaling vector. However, in order to learn a richer class of behaviors, we propose the following construction of the new behavior:

$$\kappa_n(y) = \sum_{i=1}^{N} \mu_i(y, \alpha_i) \kappa_i(y),$$

where $\mu_i : Y \times \mathbb{R}^k \to \mathbb{R}$ is a weighing function that is parameterized by control vector $\alpha_i \in \mathbb{R}^k$, as introduced in Chapter 4. It is easy to see that this more general specification of the new control mode can accommodate the linear combination solution. Note that other combinations may also be allowed, but we will use this construction for the solution presented in Section 5.2.

Now, given an observed trajectory from the human operated training example, which we denote as $y : [0, T] \to Y$, we are interested in learning a sequence of new behaviors that will approximate the training trajectory. Assume that we know the initial state $x_0 = x(0)$. Further assume that the human operator used M modes (note that we will not enforce this assumption in the presented method), then define an approximation trajectory $\tilde{x}(t)$ as follows:

$$\dot{\tilde{x}}(t) = \begin{cases} f(\tilde{x}, \kappa_{n1}(\tilde{y})) & \text{when } t \in [\tau_0, \tau_1) \\ f(\tilde{x}, \kappa_{n2}(\tilde{y})) & \text{when } t \in [\tau_1, \tau_2) \\ \quad \vdots & \quad \vdots \\ f(\tilde{x}, \kappa_{nM}(\tilde{y})) & \text{when } t \in [\tau_{M-1}, \tau_M] \end{cases} \tag{105}$$

with $\tilde{x}(0) = x(0)$. Here the observations $\tilde{y}(t) = h(\tilde{x}(t))$, and the new behavior κ_{nj} is given by

$$\kappa_{nj}(\tilde{y}) = \sum_{i=1}^{N} \mu_i(\tilde{y}, \alpha_i^j)\kappa_i(\tilde{y}), \tag{106}$$

where α_i^j is the control vector parameterizing membership function μ_i. The Learning From Example problem can be posed as an optimization problem: choose the control variables α_i^j for $j = 1, \ldots, M$ and $i = 1, \ldots, N$, and the switching times τ_i for $i = 1, \ldots, M-1$ such that the performance criterion

$$J = \int_{\tau_0}^{\tau_M} L(y(t), \tilde{y}(t))dt + \psi(y(\tau_M), \tilde{y}(\tau_M)) \tag{107}$$

is minimized, where $L : Y \times Y \to \mathbb{R}$ is the instantaneous cost, and $\psi : Y \times Y \to \mathbb{R}$ is the terminal cost. Also, in order to utilize the variational methods presented later,

we assume that L and ψ are twice differentiable in their second argument. Note here that $\tau_0 = 0$ and $\tau_M = T$ are assumed fixed.

Of course, we do not know the exact number of modes used by the human operator. Hence, we will provide an algorithm for determining the number of modes necessary to approximate the observed trajectories. We call this algorithm the *outer algorithm*, while the *inner algorithm* will find the optimal control parameters α_i^j (for $j = 1, \ldots, M$ and $i = 1, \ldots, N$) and the optimal switching times τ_i (for $i = 1, \ldots, M-1$) given the number of switches. This problem will be solved using variational arguments in Section 5.2, while an alternative solution (motivated by the LAGR project) will be discussed in Section 5.3.

5.2 Variational Approach

In this section, we utilize the calculus of variations to derive the optimality conditions for control parameters α_i^j that shape the membership functions μ_i for behavior κ_{nj}, for $j = 1, \ldots, M$ and $i = 1, \ldots, N$, and the optimal switching times $(\tau_1, \ldots, \tau_{M-1})$ with respect to the performance criterion (107) assuming an approximation trajectory with M modes. The approximation trajectory in this case is given by (105). As discussed earlier, the central theme in utilizing variational arguments is to adjoin the cost J with the constraint via a co-state (or lagrange multiplier) $\lambda(t)$. The main idea is to perturb the control parameters and compute the Gateaux (or directional) derivative of performance index in the direction of the perturbation to gain access to the optimality conditions. Since we will be differentiating the performance index, we must make some mild assumptions about differentiability. Namely, assume that L and ψ are twice differentiable in their second argument. Instead of deriving the optimality conditions explicitly, we can use the solution derived in Section 4.1, as both problem formulations are very similar. The results will be summarized in a theorem below, where the proof follows from the derivation in Section 4.1.

Theorem 5.2.1 *Given a function $y(t) \in Y$ and a set of twice differentiable functions $\kappa_i : Y \to U$ and $\mu_i : Y \times \mathbb{R}^k \to \mathbb{R}$ for $i = 1, 2, \ldots, N$, with $\tilde{x}(t) \in X$ and $\tilde{y}(t) \in Y$ given by (105) and (106), an extremum to the performance index*

$$J = \int_0^T L(y(t), \tilde{y}(t))dt + \psi(y(T), \tilde{y}(T))$$

is attained when the control parameters α_i^j and θ_i are chosen as follows:

Euler-Lagrange Equations:

$$\dot{\lambda}(t) = -\frac{\partial L}{\partial \tilde{y}}\frac{\partial h}{\partial \tilde{x}} \; - \; \lambda(t)\Big[\frac{\partial f}{\partial \tilde{x}} + \frac{\partial f}{\partial \tilde{u}}\frac{\partial \kappa_{ni}}{\partial \tilde{y}}\frac{\partial h}{\partial \tilde{x}}\Big] \text{ when } t \in (\tau_{i-1}, \tau_i),$$

Boundary Conditions:

$$\lambda(\tau_M) = \frac{\partial \psi}{\partial \tilde{y}}\frac{\partial h}{\partial \tilde{x}}(\tau_M),$$

$$\lambda(\tau_i-) = \lambda(\tau_i+) \text{ for } i = 1, \ldots, M-1,$$

Optimality Conditions:

$$\frac{\partial J}{\partial \tau_i} = \lambda(\tau_i-)\big[\tilde{f}_i(\tau_i-) - \tilde{f}_{i+1}(\tau_{i+1}+)\big] \equiv 0 \text{ for } i = 1, \ldots, M-1, \text{ and}$$

$$\frac{\partial J}{\partial \alpha_{l_r}^i} = \int_{\tau_{i-1}}^{\tau_i} \lambda \frac{\partial \tilde{f}}{\partial u}\frac{\partial \mu_l}{\partial \alpha_{l_r}^i}\kappa_l \; dt \equiv 0 \text{ for } i = 1, \ldots, M, \; l = 1, \ldots, N, \text{ and } r = 1, \ldots, k.$$

5.2.1 Numerical Algorithms

In the previous section, we derived the optimality conditions for minimizing (107) given a fixed number of modes. In this section, we first present a numerical algorithm that utilizes these optimality conditions to converge to a stationary solution for the optimal switching times and shaping parameters. We call this the *inner algorithm*, which is complemented by the *outer algorithm* that increments the number of modes and weighs the benefit of adding additional modes. The idea here is to start with an approximation trajectory using a single mode and add modes to the approximation trajectory as long as it is beneficial to do so.

Table 4. The inner algorithm for Learning from Example.

- Initialize with a guess of the control variables $\vec{\tau}^{(0)}$ and $\vec{\alpha}_i^{(0)}$ for $i = 1, \ldots, M$, and let $p = 0$.
- `while` $p < 1$ `or` $|J^{(p)} - J^{(p-1)}| < \epsilon$
  - Compute the approximation function $\tilde{x}(t)$, observation $\tilde{y}(t)$, and cost $J^{(p)}$ forward in time from 0 to T using (105), (106), and (107).
  - Compute the co-state $\lambda(t)$ backward in time from T to 0.
  - Compute the gradients $\nabla J(\vec{\tau}^{(p)})$, and $\nabla J(\vec{\alpha}_i^{(p)})$ for $i = 1, \ldots, M$.
  - Update the control variables as follows:

$$\vec{\tau}^{(p+1)} = \vec{\alpha}^{(p)} - \gamma^{(p)} \nabla J(\vec{\tau}^{(p)}),$$
$$\vec{\alpha}_i^{(p+1)} = \vec{\alpha}_i^{(p)} - \gamma^{(p)} \nabla J(\vec{\alpha}_i^{(p)}),$$

    for $i = 1, \ldots, M$.
  - $p = p + 1$
- `end while`

Table 5. The outer algorithm for Learning from Example.

- Initialize with $k = 1$
- `while` $k < 2$ `or` $|J^*_{(k)} - J^*_{(k-1)}| > \rho$
    - Obtain $J^*_{(k)}$ using the inner algorithm with number of modes $M = k$.
    - $k = k + 1$
- `end while`

The inner algorithm, which employs a gradient descent method, is shown in Table 4. Note that the choice of the step-size $\gamma^{(p)}$ can be critical for the method to converge. An efficient method among others is the use of Armijo's algorithm presented in [62]. Because of the non-convex nature of the cost function J, this gradient descent algorithm will only converge to a local minimum. Hence the attainment of a "good" local minimum can be quite dependent on the choice of a "good" initial guess for the control variables.

The inner algorithm, outlined above, provides the optimal control variables $\vec{\alpha}_i = [\alpha^i_{1_1}, \alpha^i_{1_2}, \ldots, \alpha^i_{N_k}]^T$ (for $i = 1, \ldots, M$) and optimal switching times vector $\vec{\tau} = [\tau_1, \ldots, \tau_{M-1}]^T$ given a fixed number of modes. However, recall that we do not know the number of modes *a priori*. Thus, we propose an *outer algorithm* to figure out the number of modes necessary to approximate the observed trajectory. The main idea here is to start by assuming that the observed trajectory can be approximated with a single mode, then continue to increment the number modes as long as there is a "sufficient" reduction in the performance criterion. The outer algorithm is presented in Table 5.

The parameter ρ is the thresh hold that weighs the benefit of the reduction in cost J^* versus the increase in complexity introduced by the adding an additional mode in the approximation trajectory. The choice of an appropriate ρ may be critical to

convergence of the algorithm, as choosing a small value for ρ may cause the number of modes to increase indefinitely.

5.2.2 Example

In this section, we introduce a simple example to demonstrate the viability of the proposed method. Consider the task of navigating a unicycle from a known initial configuration to a specified goal location. Recall, the unicycle dynamics are given as

$$\begin{aligned} \dot{x}_1 &= v\cos(x_3), \\ \dot{x}_2 &= v\sin(x_3), \\ \dot{x}_3 &= \omega. \end{aligned} \tag{108}$$

In the system above, (x_1, x_2) is the Cartesian coordinates of the center of the unicycle and x_3 is its orientation with respect to the x_1-axis. The initial configuration of the robot is given as x_0, while the Cartesian coordinate x_g are given as (x_{g_1}, x_{g_2}). The observed trajectory from a training run, along with the initial configuration and the desired goal, are shown in Figure 18. We assume that the linear velocity v is fixed, thus the control consists of the angular velocity control term. Also, we assume that we have a state observer, i.e., $y(t) = x(t)$. Now, we wish to learn the number of modes needed to approximate this training trajectory as well as a description of the individual modes.

As mentioned earlier, we will start of with previously established behaviors. For this example, we start with two known behaviors, namely, "go-to-goal" and "avoid-obstacles." Recall, the feedback laws corresponding to each of these behaviors are given as

$$\begin{aligned} \kappa_g(x) &= \omega_g = C_g(\phi_g - x_3), \\ \kappa_o(x) &= \omega_o = C_o(\pi + \phi_o - x_3). \end{aligned}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle, respectively. Both of these angles are

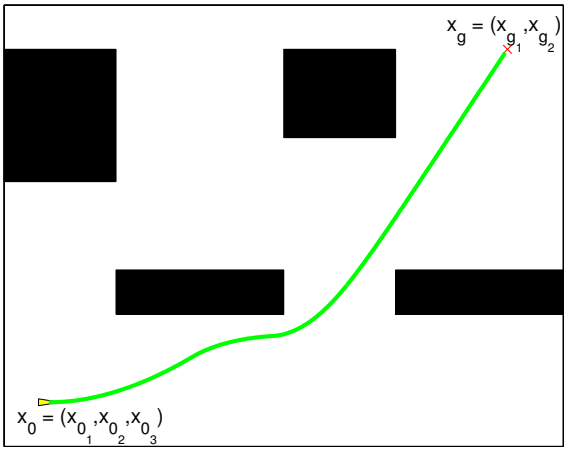


Figure 18. The observed trajectory from a training run of a unicycle navigating from x_0 to x_g.

measured with respect to the x_1-axis and can be expressed as

$$\phi_g = \arctan(\frac{x_{g_2} - x_2}{x_{g_1} - x_1}) \text{ and } \phi_o = \arctan(\frac{x_{ob_2} - x_2}{x_{ob_1} - x_1}), \tag{109}$$

where (x_{g_1}, x_{g_2}) and (x_{ob_1}, x_{ob_2}) are the Cartesian coordinates of the goal and the nearest obstacle, respectively. Moreover, the new behaviors will be given by the linear combination of these known behaviors:

$$\kappa_n(x) = \alpha_g \kappa_g(x) + \alpha_o \kappa_o(x). \tag{110}$$

As outlined earlier, we start by attempting to approximate the trajectory with one mode and then increment the number of modes as necessary. For our simulations, the cost is given by

$$L(y(t), \tilde{y}(t)) = 0.05||\tilde{y}(t) - y(t)||^2, \text{ and} \tag{111}$$

$$\psi(y(\tau_M), \tilde{y}(\tau_M)) = 10||\tilde{y}(\tau_M) - y(\tau_M)||^2. \tag{112}$$

Also, the linear velocity $v = 1$ m/s, the gains $C_g = C_o = 1$, and the thresh hold in the outer algorithm is set to $\rho = 0.1$. The step-size in the inner algorithm can be chosen using the Armijo algorithm. The optimum cost J^* as a function of the number of modes is shown in Figure 32.

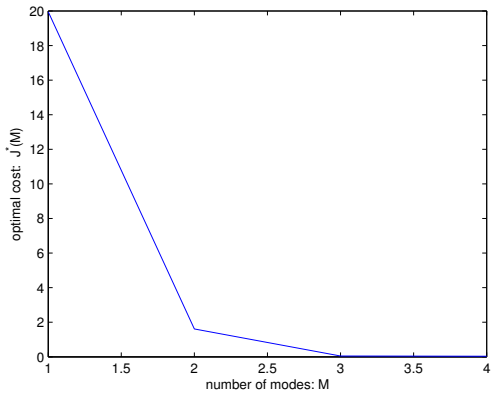


Figure 19. The optimal cost as a function of the number of modes given by the outer algorithm.

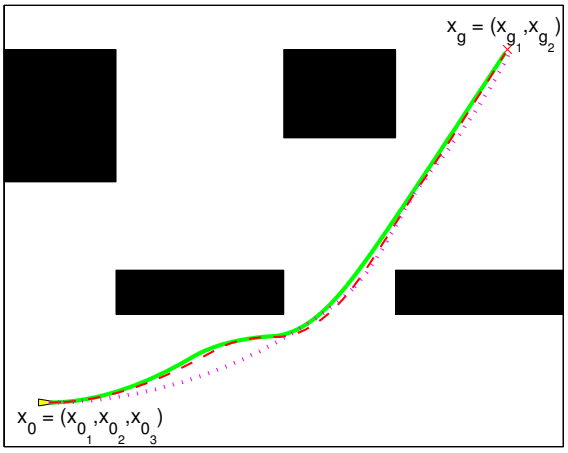
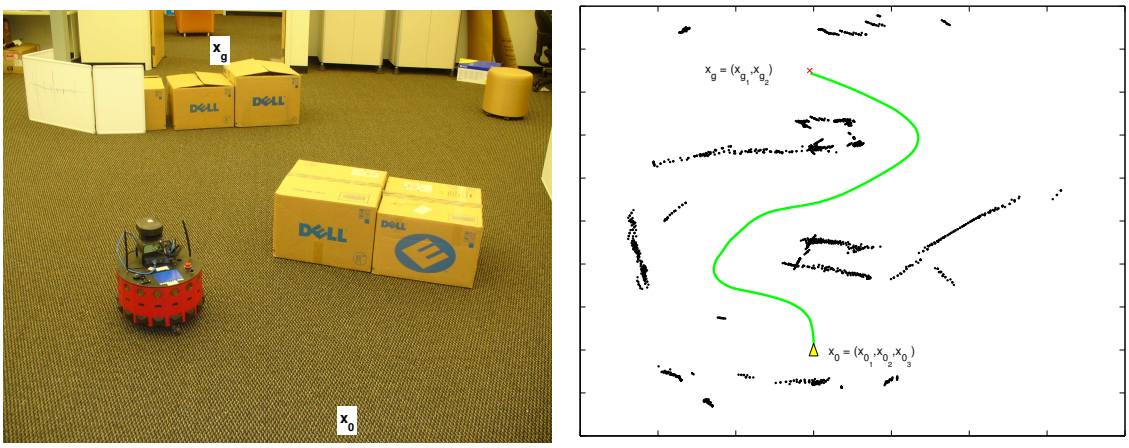


Figure 20. Depicted is the approximation trajectory ($\tilde{x}$) obtained by using three modes (dashed) and two modes (dotted) along with the original observed trajectory.



(a) (b)

Figure 21. (a) The experimental setup, (b) the observed trajectory from a training run from x_0 to x_g.

Observe that the outer algorithm quickly terminates. In fact, $|J^*(4) - J*(3)| = |0.0496 - 0.0382| = 0.0114 < \rho = 0.1$. Hence, we deduce that the observed trajectory can be approximated using three modes. The resulting trajectory from using three modes (dashed) and two modes (dotted) along with the original observed trajectory is depicted in Figure 20.

5.2.3 Navigation Using the Magellan Pro

The simulated navigation example, presented above, illustrated the operation of the proposed method for the Learning From Example problem. In this section, we will take the promising simulation results to obtain effective navigation strategies for the Magellan Pro robot from training runs. The training data was obtained from a joystick operated run guided by a human operator. The training data consisted of the state of the robot (i.e., $[x_1, x_2, x_3]$) and the range sensor readings from the entire run. The experimental setup is shown in Figure 21 (a), while the observed training trajectory, along with relative obstacles (gathered from the sensor readings), is shown in Figure 21 (b).

Recall, the dynamics of the Magellan Pro can be effectively captured by a unicycle

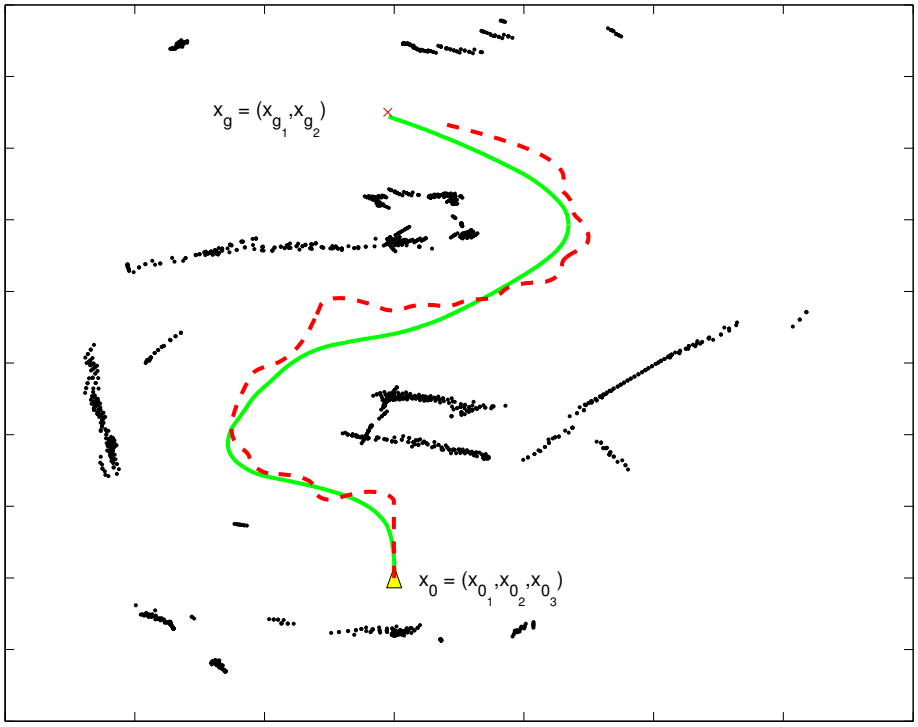


Figure 22. Depicted is the approximation trajectory ($\tilde{x}$) obtained by using the learned behavior (dashed) along with the original observed trajectory.

model (108). As usual, we start out by assuming two known behaviors, namely, "go-to-goal" and "avoid-obstacles." However, for this experiment, we let the new behaviors take on the more general form:

$$\kappa_n(x) = \mu_g(x)\kappa_g(x) + \mu_o(x)\kappa_o(x), \tag{113}$$

where the membership functions are defined as

$$\mu_g(x, \alpha_g) = 1 - e^{-\alpha_g \|x - x_o\|^2} \text{ and } \mu_o(x, \alpha_o) = e^{-\alpha_o \|x - x_o\|^2}, \tag{114}$$

as done in Section 4.3.2.

As outlined earlier, we start by attempting to approximate the trajectory with one mode and then increment the number of modes as necessary. For this experiment, the cost will be given by Equations (111)-(112), as done in the previous section. It turns out that the training trajectory can be effectively approximated using one new mode (i.e., additional modes did not significantly reduce the cost). In particular,

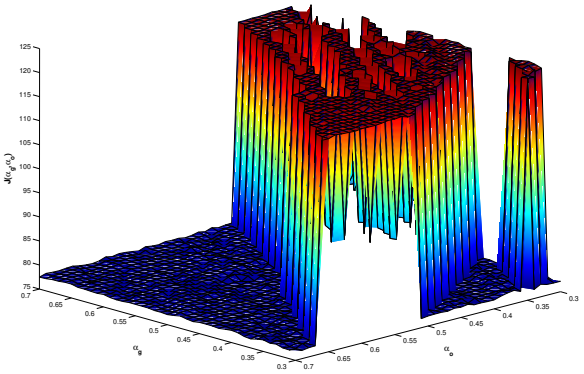


Figure 23. Depicted is the cost J as a function of the control parameters α_g and α_o. The cost surface is color scaled from low cost (blue) to high cost (red).

the optimal shaping parameters were found to be $\alpha_g^* = 0.67$ and $\alpha_o^* = 0.32$, and the corresponding cost $J^* = 76.57$. The approximation trajectory, obtained using the learned behavior, along with the training trajectory are shown in Figure 22.

For this experiment, we are only optimizing over the shaping parameters for membership functions corresponding to the "go-to-goal" behavior (α_g) and the "avoid-obstacles" behaviors (α_o). Since this optimization only involves two parameters, we can easily compute the cost $J(\alpha_g, \alpha_o)$ over a discrete set of these parameters. Figure 23 depicts the approximate cost function parameterized by α_g and α_o. Note that the cost function is highly discontinuous, as we may have expected. Moreover, there are many apparent local extremum to the performance index J. Even so, it is obvious that there are many choices for α_g and α_o that result in significant improvement over other choices (e.g., $J(\alpha_g = 0.67, \alpha_o = 0.32) = 76.57$, while $J(\alpha_g = 0.5, \alpha_o = 0.5) = 119.59$). As mentioned earlier, the performance of the gradient descent algorithm depends on the initial guess of the control vector and the step-size $\gamma^{(p)}$. It is highly recommended that the Armijo step-size be used for the descent algorithm, as this guarantees the algorithm will converge. The choice of these parameters may be especially critical for extreme cases such as the one shown in Figure 23.

Figure 24. The LAGR Robot.

5.3 Learning Applied to Ground Robots (LAGR) Project

The Learning From Example problem addressed in this chapter is one of the explicit objectives of the DARPA sponsored LAGR (Learning Applied to Ground Robots) project. The project involves taking a government-provided robot (shown in Figure 24) and only modifying the software to improve navigation performance. The robot's task is to reach a known goal location (provided via GPS coordinates) as fast as possible, over the course of three separate runs, in an initially unknown environment. Each run starts from the same start location and with the same orientation, and the success of each test is evaluated in terms of the time required to complete (if the task was completed) all three runs.

The primary objective of the project is to employ learning techniques to improve performance during each run as well as between the different runs, thus resulting in better overall performance for each test. Additionally, Learning From Example is also an explicit goal of the program. The Learning From Example component is tested separately several times throughout the duration of the project (Phase I of the project spanned 18 months). For the Learning From Example tests, we are provided training data (log-files containing all observations and control actions) that

is to be processed autonomously before each test. The training data provides cues for successfully navigating a particular course (which may be intuitive to a human operator), and the learning process is suppose to autonomously extract these cues, while the controller is to use the learned cues to improve performance. Again, the evaluation of performance is done over three runs starting from the same location.

The LAGR robot is equipped with four cameras, a Garmin GPS receiver, a front bumper switch, and an inertial navigation measuring unit (IMU). The cameras are paired together so that each pair can provide stereo depth maps with a range of approximately 6 meters. Since the observations are mainly vision driven, the perception system (used to extract useful features (such as distance, traversability, etc.) from the different observations) constitutes a major challenge, but this is not addressed here. From the control perspective, the robot uses a hybrid control architecture that combines a high-level planner (deliberative layer) with several low-level controllers (reactive layer). The different behaviors are combined through a voting scheme that is based on the Distributed Architecture for Mobile Navigation (DAMN) [85]. The actual control architecture will be detailed shortly. It is imperative to continue using this established architecture for the Learning From Example tests, as developing a different architecture specifically for this test would be unfeasible. In other words, the learning must be seamlessly integrated into the existing control architecture. As we will detail in the next section, we cannot easily apply the solution to the Learning From Example problem presented previously because of the structure of the DAMN architecture. Thus, in this section, we present an alternative solution to the optimization problem by treating the problem as a combinatorial optimization problem. Next, we briefly describe the control architecture. For a detailed explanation of the control architecture, as well as the perception and mapping modules, refer to [72].

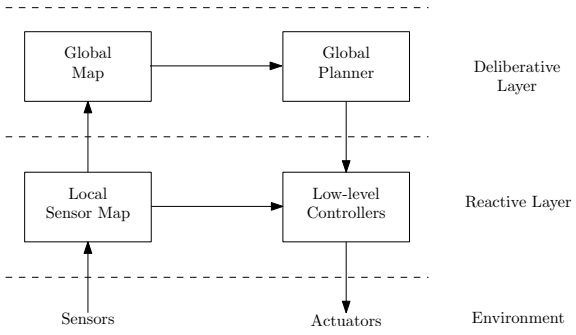


Figure 25. Standard Hybrid Control System Block Diagram.

5.3.1 LAGR Control Architecture

As mentioned earlier, the control architecture on the LAGR robot corresponds to a hybrid control strategy that combines the deliberative and reactive layers to make control decisions to drive the robot. In this architecture, behaviors are combined with the planned path (which can be considered a high-level behavior; though, typically behaviors are thought of as reactions to immediate sensory information) through a voting scheme, and this architecture is illustrated in Figure 25.

This control architecture is implemented in a manner heavily influenced by the DAMN architecture. In this implementation, individual behaviors (each designed for specific interests related to the robot's overall objective) are each given an allotment of "votes." The behaviors may cast these votes either *for* or *against* potential actions in a manner that works to achieve their particular goal. An arbitrator tallies the votes, choosing the action with the most support. This implementation utilizes straight-line paths at a resolution of 5 degrees around the robot as the control set. Similar implementations have been successfully deployed in several robotic navigation tasks [79, 86].

As is true with many other behavior-based implementations, a potential danger in this architecture is the misallocation of each behavior's gain (or in this case its allotment of votes). If the behavior's voting weights are not properly balanced, one behavior's input may dominate the tally, either preventing the robot from achieving

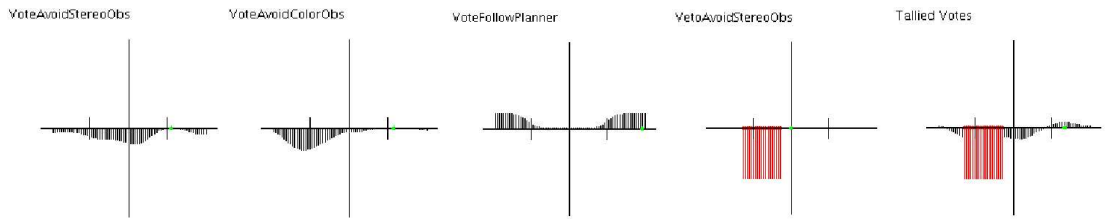


Figure 26. A graphical representation of the voting scheme employed to navigate the robot. The x-axis of each plot represents an ego-centric angular distribution of possible paths around the robot from $-\pi$ to $+\pi$, with 0 being in front of the robot. The y-axis represents the relative preference of each path, according to the respective controller. Vetoes are drawn as large negative values. The last plot represents the sum of the votes provided by all the controllers. The largest non-vetoed value is chosen for action by the robot.

higher-level goals or allowing the robot to enter an undesirable state. Noting that this weighting is typically an empirical process and dependent on both the implementation and the robot's environment, an additional layer of robustness has been added by supplementing the voting scheme with "vetoes" [74]. Each behavior, in addition to getting an allotment of votes to apply to the control set, is given the opportunity to veto each action in the set. The arbitrator respects the vetoes by disregarding any action that has been vetoed by any behavior, no matter how many votes it has garnered.

Strategically, vetoes are only used in cases when the robot faces "imminent danger." Of course, what qualifies as imminent danger is specific to each behavior. However the strength of this strategy lies in the fact that because a behavior needs only to consider imminent danger, complex calculations over the robot's configuration space that would be impossible in a full planner can be carried out. This allows the behaviors potentially to consider the full dynamics of the robot, including collision checking of rotations and the feasibility of maneuvers given the slope of the terrain. When all of these building-blocks are combined, the result is a hybrid architecture with respect to the control actions through a combination of deliberative planning and reactive behaviors.

5.3.2 Learning Behaviors From Example

As mentioned earlier, one of the goals of this project is to solve the Learning From Example problem by integrating the learning process within the existing architecture. We start by assuming some relevant features are known or specified and that a collection of behaviors exploiting these features have already been designed (e.g., "follow-path," "follow-path-to-goal," "turn-left-at-orange-fence"). These behaviors complement the initial set of behaviors which may include "go-to-goal," "avoid-obstacles," etc. Keeping with our constructivist view of learning, we wish to learn new behaviors (cued by the training examples) as a combination of these previously established behaviors. In Section 5.2, we let the new behaviors be constructed through a linear combination of the existing behaviors. Note, however, that other combinations are also allowed. Since the robot already has a control architecture that combines different behaviors through an arbitration mechanism, it would be beneficial to use this combination mechanism for learning new behaviors that would reproduce trajectories from the training examples.

It should already be apparent that the variational approach used in Section 5.2 is not easily applicable for this construction of the new mode. This is due to the fact that variational methods typically make assumptions about differentiability that are not valid for constructing new behaviors through an arbitration (using votes and vetoes) of existing behaviors that choose actions from discrete control set. Note that there are variational approaches that may be applicable with non-differentiable functions using non-differentiable calculus, but this is not pursued in this development. Instead of focusing on variational methods, we can view this problem as a combinatorial optimization problem and use one of the many algorithms known for solving such problems.

Using the LAGR control architecture, the constructivist approach to the Learn From Example problem involves finding the weights (or allotment of votes) associated

with each behavior in the behavioral combination. Clearly, the overall behavior will vary greatly based on the individual behavioral weights, i.e., the importance designated to each behavior. As mentioned earlier, it is desirable for these weights to be evenly distributed so that one behavior does not dominate arbitration. The overall objective is to select these behavioral weights such that the overall robot behavior closely resembles the behavior observed from the training data.

In the previous formulation, we determined the number of new behaviors necessary by using the outer algorithm (see Table 5) and the optimal instants to switch between these combinations as well as the weights for each combination using the inner algorithm (see Table 4). In this development, however, we will only seek one behavioral combination to best approximate the entire training trajectory (i.e., fix $M = 1$ and develop an algorithm similar to the inner algorithm). The main reason for this being to reduce the complexity of the combinatorial optimization problem, as will be seen shortly.

Combinatorial optimization problems typically involve searching over a large solution space to optimize some performance criterion. In our case, this involves searching over all possible allotment of votes for each behavior in the behavioral combination. Clearly, this space is very large (since each behavior can have many possible number of votes), and this space grows as the number of behaviors grows. Combinatorial optimization algorithms solves this hard problem by reducing the effective size of the solution space, and by exploring the space efficiently. The domain of combinatorial optimization is optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution. By assuming that $M = 1$, we do not have to solve for the switching instants and the set of feasible solutions (i.e., the allotments of votes) can be easily restricted to assume only integer values. Note that including the switching instants to the solution set would significantly increase the complexity of the the combinatorial optimization

problem.

A number of heuristic algorithms for solving combinatorial optimization problem exist in literature. It is important to note that solutions to combinatorial optimization problems is generally not unique (i.e., there can be many solutions). As is common in most optimization problems, most algorithms cannot differentiate between local optimal solutions and a rigorous optimal solution. Thus, local optimal solutions are considered solutions to the optimization problem. We will use a local search method, which is similar in spirit to the gradient descent algorithm presented earlier, called *hill climbing*. In the gradient descent algorithm, the local minimum is found by iteratively taking a step proportional to the negative of the gradient of the performance index(found using variational techniques). A local search algorithm starts from a candidate solution and then iteratively moves to a neighbor solution. In hill climbing, the next candidate is selected from a set of neighboring solutions locally minimizing the performance criterion (i.e., there is no calculation of the gradient, we just need to evaluate the set of neighboring solutions).

Recall, each behavior is a mapping from the set of observations to the set of control values, i.e., $\kappa : Y \rightarrow U$. Given a collection of such behaviors $K = \{\kappa_1, \kappa_2, \ldots, \kappa_N\}$, the new (or learned) robot behavior is given as a combination of these behaviors through the weights (i.e., allotment of votes) associated with each behavior. Thus, the output of the new behavior (e.g., desired angular velocity) is given by $u = \kappa_n(y) = \zeta(\kappa_1, \ldots, \kappa_N, \alpha_1, \ldots, \alpha_N)$, where $\alpha_i \in \mathbb{R}$ is the weight associated with behavior κ_i. The Learning From Example problem, again, involves finding $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ such that the performance criterion

$$J = \int_{\tau_0}^{\tau_M} L(y(t), \tilde{y}(t))dt + \psi(y(\tau_M), \tilde{y}(\tau_M)) \tag{115}$$

is minimized, where L again is the instantaneous cost and ψ is the terminal cost.

The algorithm for solving this problem is given in Table 6. We start with an initial guess of the control vector (referred to as the candidate), which gives each

Table 6. A hill climbing algorithm.

- Initialize with $\vec{\alpha}^{(0)} = [1, 1, \ldots, 1]^T$, and $k = -1$
- `do`
  - k = k+1
  - Generate a set of neighboring solutions $\mathbb{S}_k$ around $\vec{\alpha}^{(k)}$
  - Evaluate each solution with respect to the performance criterion
  - Let
$$\vec{\alpha}^{(k+1)} = \arg\min_{\vec{\alpha} \in \mathbb{S}_k} J(\vec{\alpha_k})$$
- `while` $(\vec{\alpha}^{(k+1)} == \vec{\alpha}^{(k)}$ `or` $||J(\vec{\alpha}^{(k+1)}) - J(\vec{\alpha}^{(k)})|| < \varepsilon)$

behavior an equal number of votes. Next, we generate a set of neighboring solutions around the candidate solution. For example, with $N = 2$, if $\vec{\alpha}^{(k)} = [1, 1]$ at time k, then $\mathbb{S}_k = \{[1-\delta, 1-\delta], [1-\delta, 1], [1-\delta, 1+\delta], [1, 1-\delta], [1, 1], [1, 1+\delta], [1+\delta, 1-\delta], [1+\delta, 1], [1+\delta, 1+\delta]\}$. Here δ encodes how far from the candidate solutions, the new neighboring solutions are allowed to be. Note that this is just one way to generate the set of neighboring solutions, there are many other acceptable approaches. Next, we evaluate each of the neighboring solutions with respect to the performance criterion given by (115). Then, we update the candidate solution by selecting the best neighboring solution (i.e., the control vector in $\mathbb{S}$ locally minimizing the performance criterion), and repeat the process by generating samples around the new candidate solution. The search terminates when either the new candidate is the same as the old candidate (i.e., $\vec{\alpha}^{(k+1)} = \vec{\alpha}^{(k)}$) or the cost difference between the new candidate and the old candidate is less than ϵ (i.e., $||J(\vec{\alpha}^{(k+1)}) - J(\vec{\alpha}^{(k)})|| < \epsilon$). Note that the selection of δ may be critical in the convergence of this algorithm. If δ is too big, the solution may not be close to the optimal solution. On the other hand, the algorithm may take a long time to converge if δ is too small. The selection of an appropriate

δ will vary based on the application, and for our application $\delta = 1$ is a reasonable choice. Also, the choice of an appropriate ϵ may be critical to the convergence of the algorithm, as choosing a small value for ϵ may cause the algorithm to converge slowly or not converge.

5.3.3 Experimental Results

Learning behaviors from training examples was tested at a vacant lot in Mableton, GA. The lot is primarily flat, but strewn with piles of landscaping waste (e.g., dead trees, bushes, and brush) providing many challenging obstacles. This example corresponds to one of the LAGR tests, where the objective was for the robot to drive autonomously from the start position to a specified goal location while emphasizing the behavior learned from the provided positive training examples. The course was designed such that appropriate learning from training examples significantly simplified the navigation task (i.e., leading to a path through much easier terrain with fewer obstacles). A relevant feature (namely, a white path laid out using lime) was clearly identified. In light of this feature, we added "follow-path," "follow-path-to-goal," "follow-path-from-goal" and "avoid-path" behaviors to complement the existing "go-to-goal," "follow-smooth-gradient," "follow-free-space," "avoid-obstacles" and "veto-obstacles" behaviors. Then we used our algorithm to learn the optimal weights for each of these behaviors using the given training examples. We started by letting each behavior have equal weights and let $\delta = 1$.

Moreover, we let the terminal cost $\psi = 0$, and the instantaneous cost be $L = ||Q(u_t) - Q(u)||^2$, where $Q : U \rightarrow U_q$ is a finite precision quantization operator. Here, u_t represents the angular velocity from the training data and u is the output of our learned behavior given the same observation. Note that we choose to quantized the angular velocity ($Q(u)$) before computing the error in order to avoid fitting noise, which may be the case if we attempt to minimize the true error between angular velocities. This problem occurs because angular velocity is not always a relevant

(a) (b) (c)

Figure 27. Sample images from the test run navigating using the learned behaviors.

measure of the actual desired behavior. For example, suppose we have two drivers manually driving a robot through the same course. One driver may be more aggressive and make sharper turns (higher angular velocities), while the less aggressive driver may make slower turns. However, both drivers successfully maneuver through the course and it is not clear that one approach is better than the other. In order to avoid this pitfall, we introduce a coarse quantization (e.g., straight, soft-left, hard-left, soft-right, hard-right) which helps us converge to a meaningful solution for $\vec{\alpha}$. Note that although angular velocities may not perfectly represent the human operator's decisions, it is all we have to go on. The algorithm converged with positive weights ($\alpha_i = 1$) for "follow-path," "follow-path-to-goal," and "follow-smooth-gradient" behaviors and zero weight for the other behaviors. Figure 27 shows sample shots from the test of this learned behavior in a course we constructed. The angular velocity was quantized to 5-levels (straight: $u_q = 0$ if $-\frac{\pi}{8} \leq u \leq \frac{\pi}{8}$, soft-left: $u_q = -1$ if $-\frac{\pi}{2} \leq u < -\frac{\pi}{8}$, hard-left: $u_q = -2$ if $u < -\frac{\pi}{2}$, soft-right: $u_q = 1$ if $\frac{\pi}{8} < u \leq \frac{\pi}{2}$, hard-right: $u_q = 2$ if $u > \frac{\pi}{2}$).

5.4 Conclusions

In this chapter, we introduced a constructivist framework for the Learning From Example problem. This framework fits within the control-centric approaches as described in the introduction, but with the fundamental difference that we assume some *a priori* knowledge of possibly relevant behaviors. Assuming we have a collection of

such behaviors, a sequence of new behaviors is learned, where each learned behavior is a combination of the existing behaviors, to approximate the example training trajectory. This constructivist view is inspired by the adaptive multi-modal control framework presented in Chapter 4. The Learning From Example problem was first approached using variational methods, and then an alternative solution, motivated by the LAGR project, using local search, combinatorial optimization methods was presented. A small-scale navigation example was presented to highlight the operation of the proposed approach, and the viability of the approach was further verified by successful application to the LAGR project.

CHAPTER 6

MULTI-MODAL, MULTI-DIMENSIONAL SYSTEMS

Up until this point, we have focused on several control aspects of hybrid systems (or more specifically, multi-modal systems). In this chapter, we focus our attention on constrained (physical) systems (see for example [87]). In our previous development, we assumed that a number of modes had been designed and concentrated on sequencing this modes to optimize some performance criterion. Next, we introduced a framework for adaptively enhancing the mode set to further improve the performance of the system. Much importance was placed on the tradeoff between expressiveness of the system and complexity of the control programs.

In this chapter, we shift our focus from such aspects of multi-modal control to address constrained multi-modal (or more generally constrained hybrid) systems. Also, we do not attempt to solve the general optimal control problem for constrained hybrid systems, but rather focus our attention on systems for which the mode sequence is fixed and given (as is commonly done for optimal control of hybrid systems [31, 36, 34]). In this case, the control parameters become the control signals within each individual mode and parameterized characterizations of the switching conditions and transition relations. We, moreover, make the model more interesting by considering constrained systems, in which different modes experience different state constraints. Rather than viewing these constraints as constraints, we, however, choose to let them induce a change in the dimension of the state space. The benefit from this is that the infinite dimensional state constraints, that typically incur significant computational overhead, are replaced by a highly non-standard model in which the dimensions changes. We will refer to such models as Multi-Mode, Multi-Dimension (or M^3D) models, following the work in [88] where such models were first introduced.

Although this work can be viewed as an extension of the [88], what is new here is

threefold. The most important novelty lies in the fact that we focus on algorithmic aspects of optimal control of M^3D systems. In other words, based on variational arguments, we will derive computational algorithms for such systems. Secondly, the class of systems under consideration here is significantly richer than what was considered in [88], with the main additional complication being that the control space also changes dimensions between different modes. Moreover, we also include an additional control parameter to control the state transitions between different modes in order to characterize a richer class of systems. Finally, we append the performance criterion to account for these transitions.

Thirdly, we will develop and study a fairly elaborate model of an ice-skater for illustrating both the main modelling ingredients as well as highlight the algorithmic aspects of the proposed optimal control methodology. This model will operate in four different modes as the skater moves forward. A mode characterizes the particular motion of each skate. As the skating motion changes, the corresponding mode transition is triggered.

The outline of this chapter is as follows: In Section 6.1, the ice-skater model will be introduced as a vehicle for illustrating the various modelling issues. Following this, in Section 6.2, M^3D systems will be formally introduced and optimality conditions will be derived using variational arguments. This section, moreover, contains a description of the development of a computational algorithm, which is then applied to the ice-skater model, in Section 6.3. The conclusions are given in Section 6.4.

6.1 A Motivating Example

In this section, we introduce a M^3D model for an ice-skater. Figure 28 shows the trajectories of both the left and right skate (dotted lines) with respect to the forward motion (from left to right). The human body is modelled by three masses: m for each leg and M for the torso and head. The skating motion is modelled as a M^3D system

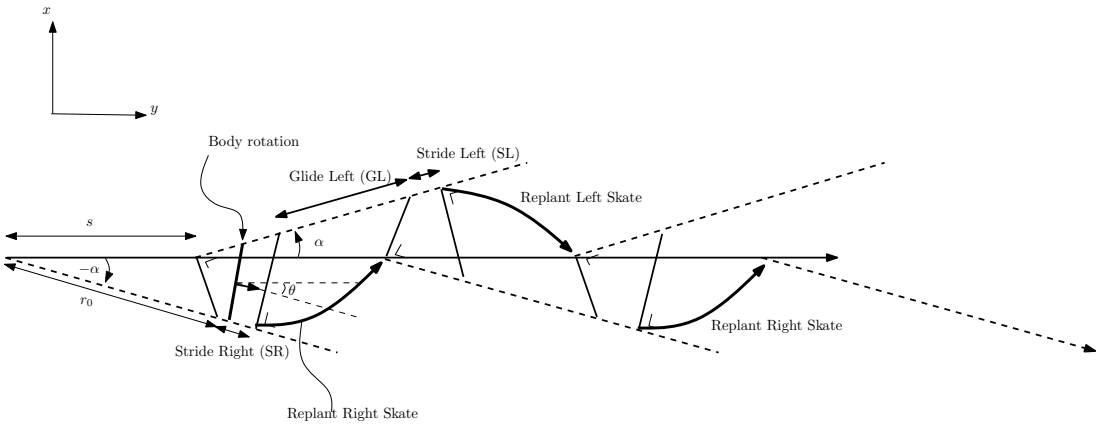


Figure 28. Skating trajectories using the proposed M^3D model.

having four modes. These modes are the 'Stride-Right' (SR) mode, the 'Glide-Left' (GL) mode, the 'Stride-Left' (SL) and the 'Glide-Right' (GR) mode. The detailed dynamics of each mode are presented next.

- *SL mode*:

  Throughout the skating motion, the angles of the left and the right skate with respect to the x-axis are denoted by α_l and α_r, respectively. During this mode, the skater applies a force u on the right skate along the line of the body as shown in Figure 29. The mass of the torso (M) and of the right leg (m) are assumed to be resting on the left skate during this acceleration. Therefore, the total mass going along the left skate is $m + M$. Accordingly, the mass on the right skate is m. As the right skate pushes outward, the same force is being applied to the right and left skate by the ice, but in the opposite direction. The components perpendicular to each skate edge are cancelled by the forces normal in the plane. The remaining components along the skate edge are responsible for the forward motion. The friction between the ice and both skates is proportional to the normal force. The proportionality constant, in turn, is a function of the velocity [89]. This friction, however, is significantly smaller than air friction that accounts for 75% of the resistance [89]. The air friction force satisfies $\mu_k v^2$, where μ_k is a constant depending on the drag coefficient, frontal area,

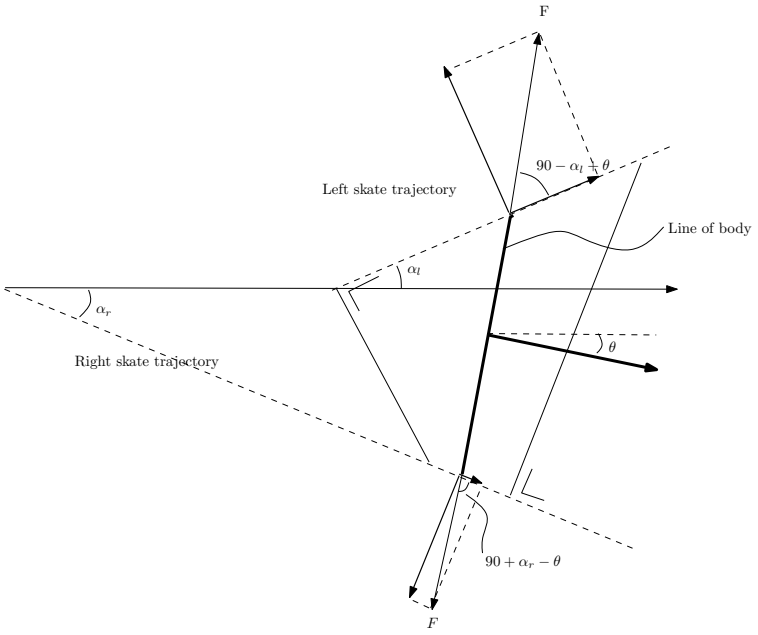


Figure 29. Depicted is the force applied during the SL mode

and the posture of the skater [90]. A physical constraint is the distance R, $R = \sqrt{(x_l - x_r)^2 + (y_l - y_r)^2}$ between the two skates. Furthermore, the heading angle θ is constrained to be $\alpha_l \leq \theta \leq \alpha_r$. Using Newton's second law, the state equations are readily obtained:

$$\begin{aligned}
\dot{x}_l &= v_l \cos(\alpha_l), \\
\dot{y}_l &= v_l \sin(\alpha_l), \\
\dot{v}_l &= \frac{u}{m+M} \sin(\alpha_l - \theta) - \frac{\mu_k}{m+M} v_c^2, \\
\dot{x}_r &= v_r \cos(\alpha_r), \\
\dot{y}_r &= v_r \sin(\alpha_r), \\
\dot{v}_r &= \frac{u}{m} \sin(\alpha_r - \theta) - \frac{\mu_k}{m} v_c^2,
\end{aligned}$$

where μ_k is the air friction coefficient, and $v_c = \frac{(m+M)v_l + m v_r}{M+2m}$ is the velocity of the center mass. Also, the heading angle $\theta = \tan^{-1}(\frac{x_l - x_r}{y_l - y_r})$.

- *GL mode*:

  This mode is the continuation of the previous mode, where the skater rests on his left skate while the right skate is lifted in the air for repositioning. The state equations, obtained by setting the applied forces to zero in the previous mode, are

$$\begin{aligned} \dot{x}_l &= v_l \cos(\alpha_l), \\ \dot{y}_l &= v_l \sin(\alpha_l), \\ \dot{v}_l &= -\frac{\mu_k}{M+2m} v_l^2. \end{aligned}$$

- *SR mode*:

  After the right skate has been replanted, the right skate begins its striding phase, while the left skate applies the force. This is similar to the SL mode with the role reversal between the left and right skates. The state equations are

$$\begin{aligned} \dot{x}_l &= v_l \cos(\alpha_l), \\ \dot{y}_l &= v_l \sin(\alpha_l), \\ \dot{v}_l &= \frac{u}{m} \sin(\theta - \alpha_l) - \frac{\mu_k}{m} v_c^2, \\ \dot{x}_r &= v_r \cos(\alpha_r), \\ \dot{y}_r &= v_r \sin(\alpha_r), \\ \dot{v}_r &= \frac{u}{m+M} \sin(\theta - \alpha_r) - \frac{\mu_k}{m+M} v_c^2, \end{aligned}$$

  where $v_c = \frac{mv_l + (m+M)v_r}{M+2m}$.

- *GR mode*:

  The end of the previous mode leads to the Glide-Right mode, where the skater glides on his right skate. This is the right skate analogue of the 'GL mode,' and

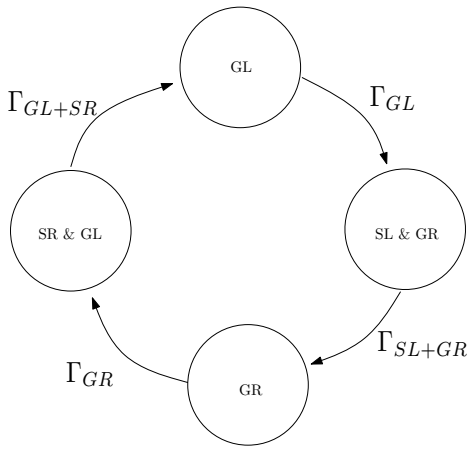


Figure 30. State Transition

the corresponding state equations are

$$\begin{aligned} \dot{x}_r &= v_r \cos(\alpha_r), \\ \dot{y}_r &= v_r \sin(\alpha_r), \\ \dot{v}_r &= -\frac{\mu_k}{M+2m} v_r^2. \end{aligned}$$

The boundary conditions at mode switching instants can be determined by physical arguments. Assuming the conservation of momentum, the velocity of the left skate at the onset of the GL from SL mode is $v_l^+ = \frac{mv_r^- + (M+m)v_l^-}{M+2m}$. Since the position of left skate is determined from the end of SL, the position of the left skate at the onset of GL satisfies $x_l^+ = x_l^-$ and $y_l^+ = y_l^-$. We further denote this set of conditions F_{SL}. During the GL mode, the right skate is being repositioned a distance of r_x units forward to prepare for the SR mode. Therefore, at the onset of the SR mode, $x_r^+ = x_l^- + r_x$ and $y_r^+ = y_l^- - r_y$. Furthermore, the position and velocity of the left skate continues from the end of GL. Hence, $x_l^+ = x_l^-$, $y_l^+ = y_l^-$, $v_l^+ = v_l^-$, and $v_r^+ = v_l^-$. We denote this transition map as F_{GL}. By similar arguments, the transition map F_{SR} from the SR mode to the GR mode is $x_r^+ = x_r^-$, $y_r^+ = y_r^-$, $v_r^+ = \frac{(m+M)v_r + mv_l}{M+2m}$, and the transition map F_{GR} from GR to SL is $x_l^+ = x_r^- + r_x$, $y_l^+ = y_r^- + r_y$, $v_l^+ = v_r^-$,

$x_r^+ = x_r^-$, $y_r^+ = y_r^-$, and $v_r^+ = v_r^-$.

6.2 Optimal Control Framework

Having motivated the utility of optimal control of multi-dimensional hybrid systems in the previously, in this section we began by formalizing the optimal control problem. Then, we use a variational arguments to derive the necessary conditions for optimality. Once these conditions are obtained, we will present a numerical algorithm that utilizes these optimality conditions to converge to a stationary solution for the optimal control parameters. This algorithm is particularly interesting since we cannot use the standard gradient descent algorithm here because of the change in dimension of the control space.

6.2.1 Problem Formulation

The dynamical system discussed in this chapter corresponds to a specific class of hybrid systems, where the dimension of the state and control space changes between different modes of operation. We assume that switches between the different dynamics is time-driven, where the switching-time vector $\vec{\tau} = [\tau_1, \ldots, \tau_{N-1}]^T$ is also a control parameter. Moreover, the ordering of the modes is assumed known and fixed. Also, the initial time $\tau_0 = 0$ and final time $\tau_N = T$ will be assumed fixed. It will be beneficial to introduce an identifier $p(i)$, taking values in a finite set, denoting the mode of operation during the time interval $[\tau_{i-1}, \tau_i)$. As mentioned earlier, the dimensions of the state and control spaces vary from mode to mode. Hence, we let $x^{p(i)} \in \mathbb{R}^{n^{p(i)}}$, while $u^{p(i)} \in \mathbb{R}^{m^{p(i)}}$. Now, the state evolution during time interval $[\tau_{i-1}, \tau_i)$ is given

$$\dot{x}^{p(i)} = f^{p(i)}(x^{p(i)}(t), u^{p(i)}(t)), \tag{116}$$

where $f^{p(i)} : \mathbb{R}^{n^{p(i)}} \times \mathbb{R}^{m^{p(i)}} \to \mathbb{R}^{n^{p(i)}}$ is a twice differentiable continuous-state transition function in mode $p(i)$. Thus the control, thus far, consists of a continuous time input $u^{p(i)}(\cdot)$ for each mode $p(1), \ldots, p(N)$ and the switching time vector $\vec{\tau}$.

Note that since the state trajectory switches between different dimensions, the state trajectories are discontinuous at the switching instants. The transition functions at the switching time instants are given as follows:

$$x^{p(i+1)}(\tau_i+) = F^{p(i)}(x^{p(i)}(\tau_i-), w^{p(i)}), \tag{117}$$

for $i = 1, \ldots, N$. Here, $F^{p(i)} : \mathbb{R}^{n^{p(i)}} \times \mathbb{R}^{k^{p(i)}} \rightarrow \mathbb{R}^{n^{p(i+1)}}$ is a twice differentiable discrete-state transition function, and $w^{p(i)} \in \mathbb{R}^{k^{p(i)}}$ is a control parameter. For ease of notation, let's parameterize the state and control vectors by their sequential index rather than the identifier $p(i)$. Thus if we start with the initial state $x_1(0)$, the state trajectory will be given as follows:

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t)), \text{ when } t \in [\tau_{i-1}, \tau_i) \tag{118}$$

$$x_{i+1}(\tau_i+) = F_i(x_i(\tau_i), w_i), \tag{119}$$

for $i = 1, \ldots, N$. Note here once again, that $x_i \in \mathbb{R}^{n_i}$, $u_i \in \mathbb{R}^{m_i}$ when $t \in [\tau_{i-1}, \tau_i)$, and $w_i = \mathbb{R}^{k_i}$.

Now that we have a characterization of the state trajectory, we can formulate an optimal control problem. More specifically, the problem is to determine the optimal continuous control signals $u_i(t)$ for $i = 1, \ldots, N$, discrete control signals w_i for $i = 1, \ldots, N-1$, and the switching time vector $\vec{\tau} = [\tau_1, \ldots, \tau_{N-1}]^T$ in order to minimize a performance index

$$J = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_N} L_i(x_i, u_i)dt + \sum_{i=1}^{N-1} \phi_i(x_i(\tau_i-), w_i) + \Phi(x_N(\tau_N)). \tag{120}$$

Here $L_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}$ is the instantaneous cost in mode i, while $\phi_i : \mathbb{R}^{n_i} \times \mathbb{R}^{k_i}$ is a state transition cost between modes and $\Phi : \mathbb{R}^{n_N} \rightarrow \mathbb{R}$ is the terminal cost. In the next subsection, we will derive the optimal control via calculus of variations. For this reason, we assume that L_i (for $i = 1, \ldots, N$), ϕ_i (for $i = 1, \ldots, N-1$), and Φ are twice differentiable.

6.2.2 Optimality Conditions

In this section, we derive the optimality conditions for the problem defined above using a variational approach. This approach avoids the explicit computation of the perturbations with a clever choice of the Lagrange multipliers. Adjoining the dynamical constraints (118) to the cost (120) via different Lagrange multipliers (or co-states), $\lambda_i(t) \in \mathbb{R}^{1\times n_i}$, defined over time interval (τ_{i-1}, τ_i), will not alter the value of J. Moreover, by adjoining the state transition constraints at the switching times (119) via Lagrange multipliers $\mu_i \in \mathbb{R}^{1\times n_{i+1}}$, and assuming that the optimal control variables are chosen, we obtain the optimal cost $\bar{J}_0$.

Defining the Hamiltonians,

$$H_i(x_i, \lambda_i, u_i) = L_i(x_i, u_i) + \lambda_i f_i(x_i, u_i), \tag{121}$$

the augmented (but unaltered from an evaluation point of view) cost is given by

$$\bar{J}_0 = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \Big[H_i(x_i, \lambda_i, u_i) - \lambda_i \dot{x}_i\Big] dt + \sum_{i=1}^{N-1} \mu_i \Big[F_i(x_i(\tau_i-), w_i) - x_{i+1}(\tau_i+)\Big] + \\ + \sum_{i=1}^{N} \phi_i(x_i(\tau_i-), w_i). \tag{122}$$

In the equation above, we let $\phi_N(x_N(\tau_N-), w_N) = \Phi_N(x_n(\tau_N))$.

Now, we perturb (122) in such a way that $u_i \to u_i + \epsilon\nu_i$ for $i = 1, \ldots, N$, $\tau_i \to \tau_i + \epsilon\theta_i$, and $w_i \to w_i + \epsilon\omega_i$ for $i = 1, \ldots, N-1$. With $\epsilon << 1$, this perturbation induces a sequence of perturbations $\{\eta_i\}$ in the state trajectories x_i, i.e., $x_i \to x_i + \epsilon\eta_i$. Thus, the perturbed cost, denoted by $\bar{J}_\epsilon$, is given by

$$\bar{J}_\epsilon = \sum_{i=1}^{N} \int_{\tau_{i-1}+\epsilon\theta_{i-1}}^{\tau_i+\epsilon\theta_i} \Big[H_i(x_i + \epsilon\eta_i, \lambda_i, u_i + \epsilon\nu_i) - \lambda_i(\dot{x}_i + \epsilon\dot{\eta}_i)\Big] dt + \\ + \sum_{i=1}^{N-1} \mu_i \Big[F_i\big((x_i + \epsilon\eta_i)|_{(\tau_i+\epsilon\theta_i)-}, w_i + \epsilon\omega_i\big) - (x_{i+1} + \epsilon\eta_{i+1})|_{(\tau_i+\epsilon\theta_i)+}\Big] + \\ + \sum_{i=1}^{N} \phi_i((x_i + \epsilon\eta_i)|_{(\tau_i+\epsilon\theta_i)-}, w_i + \epsilon\omega_i). \tag{123}$$

Note that $\theta_0 = \theta_N = 0$ since the initial and final times are assumed fixed. The first first order approximation of (123) yields

$$\begin{aligned}\bar{J}_\epsilon = &\sum_{i=1}^{M}\int_{\tau_{i-1}}^{\tau_i}\Big[H_i(x_i,\lambda_i,u_i)-\lambda_i\dot{x}\Big]dt+\sum_{i=1}^{N}\int_{\tau_i}^{\tau_i+\epsilon\theta_i}L_i(x_i,u_i)dt-\\
&-\sum_{i=1}^{N}\int_{\tau_{i-1}}^{\tau_{i-1}+\epsilon\theta_{i-1}}L_i(x_i,u_i)dt+\epsilon\sum_{i=1}^{N}\int_{\tau_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial x_i}\eta_i+\frac{\partial H_i}{\partial u_i}\nu_i-\lambda_i\dot{\eta}_i\Big]dt+\\
&+\sum_{i=1}^{N-1}\mu_i\Big[F_i\big((x_i(\tau_i+\epsilon\theta_i)-),w_i+\epsilon\omega_i\big)-x_{i+1}(\tau_i+\epsilon\theta_i)+)\Big]+\\
&+\epsilon\sum_{i=1}^{N-1}\mu_i\Big[\frac{\partial F_i}{\partial x_i}\eta_i(\tau_i-)-\eta_{i+1}(\tau_i+)\Big]+\\
&+\sum_{i=1}^{N}\phi_i\big((x_i((\tau_i+\epsilon\theta_i)-),w_i+\epsilon\omega_i\big)+\epsilon\sum_{i=1}^{N}\frac{\partial\phi_i}{\partial x_i}\eta(\tau_i-).\end{aligned}\tag{124}$$

Note that we explicitly used the fact that $f_i(x_i(t),u_i(t))-\dot{x}_i(t)$ is zero in the open intervals $(\tau_{i-1},\tau_{i-1}+\epsilon\theta_{i-1})$ and $(\tau_i,\tau_i+\epsilon\theta_i)$.

Now the first variation in the performance index (120) can be expressed as the limit for $\epsilon\to 0$ of

$$\delta J=\lim_{\epsilon\longrightarrow 0}\frac{\bar{J}_\epsilon-\bar{J}_0}{\epsilon}.\tag{125}$$

Thus using (122) and (124), it follows that

$$\begin{aligned}\delta J=&\sum_{i=1}^{N}\int_{\tau_{i-1}}^{\tau_i}\Big[\frac{\partial H_i}{\partial x_i}\eta_i+\frac{\partial H_i}{\partial u_i}\nu_i-\lambda_i\dot{\eta}_i\Big]dt+\\
&+\sum_{i=1}^{N}L_i(x_i,u_i)|_{\tau_i}\theta_i-L_i(x_i,u_i)|_{\tau_{i-1}}\theta_{i-1}+\\
&+\sum_{i=1}^{N-1}\mu_i\Big[\frac{\partial F}{\partial x_i}\dot{x}_i(\tau_i-)\theta_i+\frac{\partial F}{\partial w_i}\omega_i-\dot{x}_{i+1}(\tau_i+)\theta_i\Big]+\\
&+\sum_{i=1}^{N-1}\mu_i\Big[\frac{\partial F_i}{\partial x_i}\eta_i(\tau_i-)-\eta_{i+1}(\tau_i+)\Big]+\\
&+\sum_{i=1}^{N}\Big[\frac{\partial\phi_i}{\partial x_i}\dot{x}_i(\tau_i-)\theta_i+\frac{\partial\phi_i}{\partial w_i}\omega_i+\frac{\partial\phi_i}{\partial x_i}\eta(\tau_i-)\Big].\end{aligned}\tag{126}$$

Reordering the sum and reorganizing terms, remembering that $\theta_0 = \theta_N = 0$, we get

$$\begin{aligned}\delta J = &\sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \Big[\frac{\partial H_i}{\partial x_i}\eta_i + \frac{\partial H_i}{\partial u_i}\nu_i - \lambda_i \dot{\eta}_i\Big] dt + \\ &+ \sum_{i=1}^{N-1} \Big[L_i(x_i, u_i) - L_{i+1}(x_{i+1}, u_{i+1})\Big]_{\tau_i} \theta_i + \\ &+ \sum_{i=1}^{N-1} \Big[\mu_i \frac{\partial F}{\partial x_i} f_i(\tau_i -) - \mu_i f_{i+1}(\tau_i +) + \frac{\partial \phi_i}{\partial x_i} f_i(\tau_i -)\Big]\theta_i + \\ &+ \sum_{i=1}^{N-1} \mu_i \Big[\frac{\partial F_i}{\partial x_i}\eta_i(\tau_i -) - \eta_{i+1}(\tau_i +)\Big] + \\ &+ \sum_{i=1}^{N-1} \frac{\partial F}{\partial w_i}\omega_i + \sum_{i=1}^{N} \Big[\frac{\partial \phi_i}{\partial w_i}\omega_i + \frac{\partial \phi_i}{\partial x_i}\eta(\tau_i -)\Big]. \end{aligned} \tag{127}$$

Using integration by parts, the integral terms in (127) further reduces to

$$\begin{aligned}\delta \mathcal{K} = &\sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \Big[\frac{\partial H_i}{\partial x_i}\eta_i + \frac{\partial H_i}{\partial u_i}\nu_i - \dot{\lambda}_i \eta_i\Big] dt - \\ &- \sum_{i=1}^{N} \Big[\lambda_i(\tau_i -)\eta_i(\tau_i -) - \lambda_i(\tau_{i-1} +)\eta_i(\tau_{i-1} +)\Big]. \end{aligned} \tag{128}$$

Substituting $\delta \mathcal{K}$ into δJ, and choosing λ_i in the intervals (τ_{i-1}, τ_i) to solve

$$\dot{\lambda}_i = -\frac{\partial H_i}{\partial x_i}(x_i, \lambda_i, u_i), \tag{129}$$

yields

$$\begin{aligned}\delta J = &\sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} A_i \nu_i dt + \sum_{i=1}^{N-1} B_i \omega_i + \sum_{i=1}^{N-1} C_i \theta_i + \\ &+ \sum_{i=1}^{N-1} \Big[\lambda_{i+1}(\tau_i +) - \mu_i\Big]\eta_{i+1}(\tau_{i+1} +) + \\ &+ \sum_{i=1}^{N-1} \Big[\mu_i \frac{\partial F_i}{\partial x_i} + \frac{\partial \phi_i}{\partial x_i} - \lambda_i(\tau_i -)\Big]\eta_i(\tau_i -) + \\ &+ \Big[\frac{\partial \Phi}{\partial x_N} - \lambda_N(\tau_N -)\Big]\eta_N(\tau_N -), \end{aligned} \tag{130}$$

where

$$A_i = \frac{\partial H_i}{\partial u_i}, \tag{131}$$

$$B_i = \frac{\partial \phi_i}{\partial w_i} + \mu_i \frac{\partial F}{\partial w_i}, \text{ and} \tag{132}$$

$$C_i = \Big[L_i(x_i, u_i) - L_{i+1}(x_{i+1}, u_{i+1})\Big]_{\tau_i} + \\ + \Big[\mu_i \frac{\partial F}{\partial x_i} f_i(\tau_i -) - \mu_i f_{i+1}(\tau_i +) + \frac{\partial \phi_i}{\partial x_i} f_i(\tau_i -)\Big]. \tag{133}$$

Here, we used the fact that $\phi_N(x_N(\tau_N -), w_N) = \Phi_N(x_n(\tau_N))$ and $\eta_1(0+) = 0$. The computation of the perturbations $\{\eta_i\}$ is avoided by choosing

$$\mu_i = \lambda_{i+1}(\tau_i +), \tag{134}$$

$$\lambda_i(\tau_i -) = \mu_i \frac{\partial F_i}{\partial x_i} + \frac{\partial \phi_i}{\partial x_i}, \text{ and} \tag{135}$$

$$\lambda_N(\tau_N -) = \frac{\partial \Phi}{\partial x_N}. \tag{136}$$

These conditions specify the boundary conditions of the co-state defined by (129).

With this choice of the co-state, the first order variation of J reduces to

$$\delta J = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} A_i \nu_i dt + \sum_{i=1}^{N-1} B_i \omega_i + \sum_{i=1}^{N-1} C_i \theta_i. \tag{137}$$

Since the control parameters are independent, the necessary conditions for optimality are the vanishing of A_is, B_is, and C_is in (137). These results are summarized in a theorem below:

Theorem *Given a multi-dimensional, multi-modal system of the form (118) and (119), an extremum to the performance index J in (120) is attained when the control variables u_i (for $i = 1, \ldots, N$), τ_i and w_i (for $i = 1, \ldots, N-1$) are chosen as follows: Euler-Lagrange Equations:*

$$\dot{\lambda}_i = -\frac{\partial H_i}{\partial x_i}(x_i, \lambda_i, u_i) \text{ with } t \in (\tau_{i-1}, \tau_i), \text{ for } i = 1, \ldots, N,$$

Boundary Conditions:

$$\lambda_N(\tau_N-) = \frac{\partial \Phi}{\partial x_N}, \text{ and}$$
$$\lambda_i(\tau_i-) = \lambda_{i+1}(\tau_i+)\frac{\partial F_i}{\partial x_i} + \frac{\partial \phi_i}{\partial x_i} \text{ for } i = 1, \ldots, N-1,$$

Optimality Conditions:

$$\frac{\partial H_i}{\partial u_i} = 0,$$
$$\frac{\partial \phi_i}{\partial w_i} + \lambda_{i+1}(\tau_i+)\frac{\partial F}{\partial w_i} = 0, \text{ and}$$
$$H_i(\tau_i-) - H_{i+1}(\tau_i+) = 0,$$

where H_i is the Hamiltonian

$$H_i(x_i, \lambda_i, u_i) = L_i(x_i, u_i) + \lambda_i f_i(x_i, u_i).$$

6.2.3 Numerical Algorithms

Now that we have the necessary conditions for optimality, we introduce a numerical algorithm that utilizes these conditions to attain optimal control values, shown in Table 7.

This algorithm is similar to a gradient descent algorithm, however there is one big distinction. The switching times τ_i and discrete control w_i can be readily updated in the negative gradient direction as usual. However, the continuous control u_i cannot be updated using the standard approach because of the change in dimensions between modes. To see why this happens, consider the situation depicted in Figure 31. Here, if we update the control u_i using the usual update method, the $u_i^{(p+1)}(t) \in \mathbb{R}^{m_i}$ when $t \in [\tau_{i-1}^{(p)}, \tau_i^{(p)})$. However, upon updating the switching times, there will be two regions of conflict assuming the switching times change.

There are four distinct cases of conflict that can occur for each control u_i. To address the update issue and the regions of conflict, we propose the a sub-function for updating the continuous control u_i called `update-u` (shown in Table 8).

Table 7. A descent algorithm for M^3D systems.

- Initialize with a guess of the control variables $\tau_i^{(0)}$, $w_i^{(0)}$, for $i = 1, \ldots, N-1$, and $u_i^{(0)}(t)$ with $t \in [\tau_{i-1}^{(0)}, \tau_i^{(0)})$ for $i = 1, \ldots, N$, and let $p = 0$.

- `while` $p < 1$ `or` $|J^{(p)} - J^{(p-1)}| < \epsilon$

  1. Compute the state trajectories $x_i(t)$, for $i = 1, \ldots, N$, and cost $J^{(p)}$ forward in time from 0 to T using (118), (119), and (120).
  2. Compute the co-states $\lambda_i(t)$, for $i = 1, \ldots, N$, backward in time from T to 0 using (129), and (134) - (136).
  3. Compute A_i, B_i, C_i for $i = 1, \ldots, N$ using (131)-(133).
  4. Update the control variables τ_i and w_i as follow :

$$\tau_i^{(p+1)} = \tau_i^{(p)} - \gamma_\tau^{(p)} C_i,$$
$$w_i^{(p+1)} = w_i^{(p)} - \gamma_w^{(p)} B_i,$$

     for $i = 1, \ldots, N-1$.

  5. Update the control u_i using the `update-u` sub-function (defined below):

$$u_i^{(p+1)} = \texttt{update-u}(u_i^{(p)}).$$

  6. $p = p + 1$

- `end while`

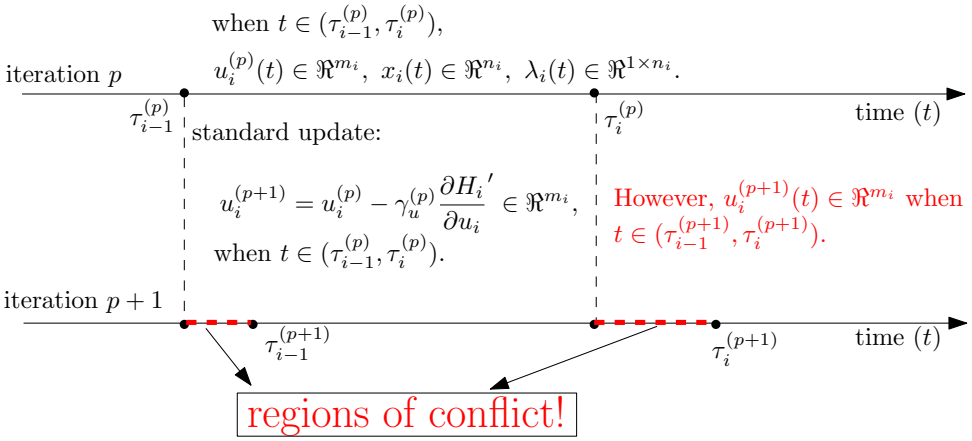


Figure 31. Depicted here is a situation where the standard update method leads to a conflict in dimensions of the control u_i.

Table 8. An algorithm to update u during each iteration.

$u_i^{(p+1)} =$ `update-u`$(u_i^{(p)})$

- $u_{temp}(t) = u_i^{(p)} - \gamma_u^{(p)} \frac{\partial H_i}{\partial u_i^{(p)}}'$
- `if` $\left(\tau_{i-1}^{(p+1)} \geq \tau_{i-1}^{(p)} \ \& \ \tau_i^{(p+1)} \geq \tau_i^{(p)}\right)$
  - $u_i^{(p+1)}(t) = u_{temp}(t);\ t \in [\tau_{i-1}^{(p+1)}, \tau_i^{(p)})$,
  - $u_i^{(p+1)}(t) = u_{temp}(\tau_i^{(p)}) + (t - \tau_i^{(p)})\dot{u}_{temp}(\tau_i^{(p)});\ t \in [\tau_i^{(p)}, \tau_i^{(p+1)})$.
- `elseif` $\left(\tau_{i-1}^{(p+1)} \geq \tau_{i-1}^{(p)} \ \& \ \tau_i^{(p+1)} \leq \tau_i^{(p)}\right)$
  - $u_i^{(p+1)}(t) = u_{temp}(t);\ t \in [\tau_{i-1}^{(p+1)}, \tau_i^{(p+1)})$.
- `elseif` $\left(\tau_{i-1}^{(p+1)} \leq \tau_{i-1}^{(p)} \ \& \ \tau_i^{(p+1)} \geq \tau_i^{(p)}\right)$
  - $u_i^{(p+1)}(t) = u_{temp}(t);\ t \in [\tau_{i-1}^{(p)}, \tau_i^{(p)})$,
  - $u_i^{(p+1)}(t) = u_{temp}(\tau_i^{(p)}) + (t - \tau_i^{(p)})\dot{u}_{temp}(\tau_i^{(p)});\ t \in [\tau_i^{(p)}, \tau_i^{(p+1)})$,
  - $u_i^{(p+1)}(t) = u_{temp}(\tau_{i-1}^{(p)}) + (\tau_{i-1}^{(p)} - t)\dot{u}_{temp}(\tau_{i-1}^{(p)});\ t \in [\tau_{i-1}^{(p+1)}, \tau_{i-1}^{(p)})$.
- `elseif` $\left(\tau_{i-1}^{(p+1)} \leq \tau_{i-1}^{(p)} \ \& \ \tau_i^{(p+1)} \leq \tau_i^{(p)}\right)$
  - $u_i^{(p+1)}(t) = u_{temp}(t);\ t \in [\tau_{i-1}^{(p)}, \tau_i^{(p+1)})$,
  - $u_i^{(p+1)}(t) = u_{temp}(\tau_{i-1}^{(p)}) + (\tau_{i-1}^{(p)} - t)\dot{u}_{temp}(\tau_{i-1}^{(p)});\ t \in [\tau_{i-1}^{(p+1)}, \tau_{i-1}^{(p)})$.
- `end if`

The idea here is to *trim* and *extend* the control u_i as necessitated by the change in the switching times. The extension is done by using a first order Taylor approximation. The instance shown in Figure 31 corresponds to the when $\tau_{i-1}^{(p+1)} > \tau_{i-1}^{(p)}$ and $\tau_i^{(p+1)} > \tau_i^{(p)}$. In this case, since τ_{i-1} increased, the beginning of u_i (i.e., $u_i(t)$ when $t \in [\tau_{i-1}^{(p-1)}, \tau_{i-1}^{(p)})$) is trimmed. Also since τ_i increased, the end of u_i (i.e., $u_i(t)$ when $t \in [\tau_i^{(p-1)}, \tau_i^{(p)})$) must be extended. Trimming u_i is simple, as this involves ignoring $u_i(t)$ for the conflicting time period (i.e., when $t \in [\tau_{i-1}^{(p-1)}, \tau_{i-1}^{(p)})$). Extending u_i is little more involved, in this case we try to approximate what u_i should be in the conflicting time period by using Taylor expansion, as shown in Table 8. Thus, we let

$$u_i^{(p+1)}(t) = u_{temp}(\tau_{i-1}^{(p)}) + (\tau_{i-1}^{(p)} - t)\dot{u}_{temp}(\tau_{i-1}^{(p)}), \text{ when } t \in [\tau_{i-1}^{(p+1)}, \tau_{i-1}^{(p)}).$$

The other cases are similar.

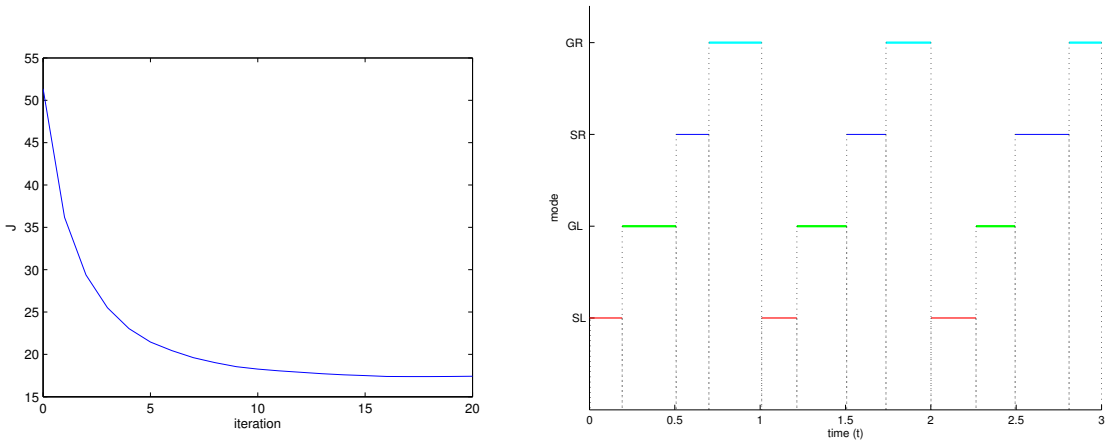
6.3 Optimal Control of an Ice Skater

We will now use these algorithms to derive the optimal control of the ice skater using the model presented in Section 6.1. In particular, we will address the problem of starting from rest (i.e., $v_c(0) = 0$) and achieving a desired velocity v_d in final time T, while minimizing the control energy (or work done). With this goal in mind, the following performance criterion is proposed:

$$J = \int_0^T C_1 u(t) d(t) dt + C_2 (v_c(T) - v_d)^2, \tag{138}$$

where C_1 and C_2 are scalar gains, and $d(t)$ represents the distance travelled. In order to fit this performance index into the general framework presented in Section 6.2, we will have to mildly modify the state equations derived in Section 6.1. First, note that $u(t) = 0$ in the GL and GR modes, hence $L_i = 0$ in GL and GR modes. In the SR & GL and GR & SL modes, we introduce a new state $d(t)$ to keep track of the distance travelled, which evolves as

$$\dot{d}(t) = v_c(t), \tag{139}$$



(a) (b)

Figure 32. (a) The evolution of the cost as a function of the iteration, (b) the active mode as a function of time for the optimal switching times.

where $d(t)$ is initialized to be 0 at the beginning of the SR & GL and GR & SL modes. With this augmented state $x_i = [x_l, y_l, v_l, x_r, y_r, v_r, d]^T$, $L_i(x_i, u_i) = C_1 u_i(t) d(t)$ in the SR & GL and GR & SL modes. Moreover, we note that $\phi_i(x_i(\tau_i-), w_i) = 0$ and $\Phi(x_N(\tau_N)) = C_2(v_c(\tau_N) - v_d)^2$.

For the purpose of the simulation, we will let the state transitions (F_i) be autonomous (i.e., no discrete control w_i), and fix $\alpha_l = \frac{\pi}{6}$ and $\alpha_r = -\frac{\pi}{6}$. In this case the control consists of the switching times τ_i and the continuous control $u_i(t)$. We will start in the SR & GL mode and transition between different modes as specified in Figure 29. In the simulation, $x_0 = [0, 0, 0, 0.25, -0.25, 0]$, $M = 40\ kg$, $m = 20\ kg$, $\mu_k = 0.157\ \frac{kg}{m}$ ([90]), $v_d = 2\ \frac{m}{s}$, $T = 3\ s$, $C_1 = 0.01$, and $C_2 = 50$. The evolution of the performance index as a function of the iteration is shown in Figure 32 (a), while Figure 32 (b) shows the optimal switching times by displaying the active mode as a function of time. Finally, the skating trajectory using the optimal control u_i and optimal switching times τ_i is shown in Figure 33.

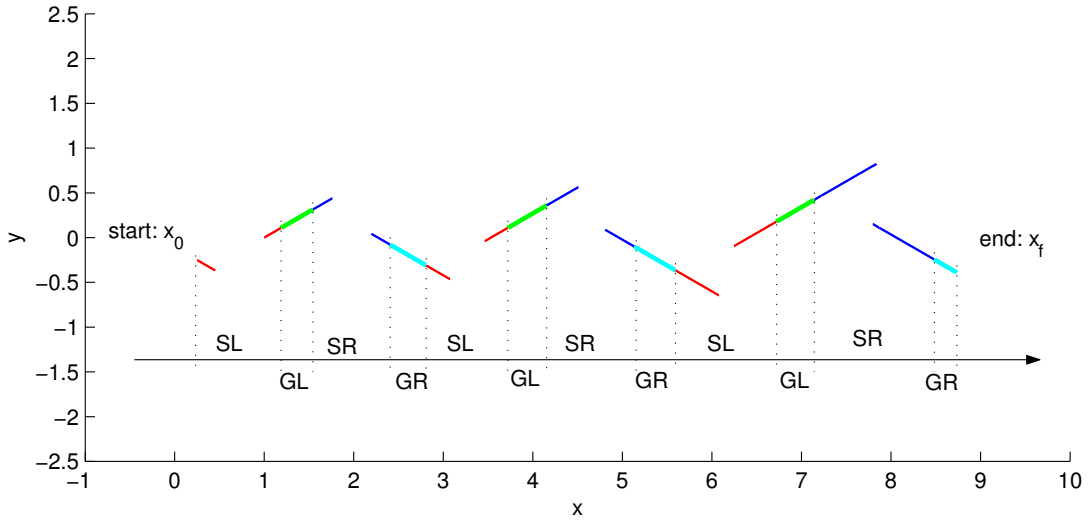


Figure 33. Depicted is the optimal trajectory staring in SR & GL mode and switching between the GL, SL & GR, GR modes.

6.4 Conclusions

In this chapter, we introduced an algorithmic framework for the optimal control of systems that experience different constraints during different modes of operation. These constraints can be handled using traditional methods (e.g., using Lagrange multipliers), but this typically adds significant computational overhead. Instead, we introduced a non-standard Multi-Mode, Multi-Dimension (M^3D) model to capture these infinite-dimensional state constraints and derived optimality conditions for such systems using variational arguments. We, moreover, derived a detailed M^3D model for an ice-skater, and demonstrated the viability of the presented methods through an optimal control example of the ice-skater.

CHAPTER 7

CONCLUSIONS AND EXTENSIONS

7.1 Conclusions

The general contribution of this thesis is captured by the title of the thesis, namely to incorporate the concept of optimality to multi-modal control and apply the theoretical results to robotics applications for developing successful navigation strategies for autonomous mobile robot. To this end, the main contributions of this thesis are

- *From local rules to global behaviors:* Given a collection of modes, we presented an algorithm that utilizes rapidly-exploring random trees for reachability analysis to characterize the expressiveness of the multi-modal system. Moreover, the algorithm uses reinforcement learning at the modal level to learn control programs (i.e., mode strings) that complete a desired task while minimizing a prescribed performance criterion.

- *Adaptive multi-modal control:* We developed a variational framework for adaptive multi-modal control, where a given collection of modes is adapted by adding new modes to the set instead of changing the existing modes. We showed how designing new modes to replace recurring mode string fragments can increase the expressiveness of the system (thus, possibly improving performance) while decreasing the specification complexity of the control programs. We presented a gradient descent algorithm to construct such replacement modes, which utilizes optimality conditions obtained using the calculus of variations.

- *Learning from example:* We presented a constructivist approach to the Learning From Example problem, which is inspired by adaptive multi-modal control. First, we used the variational framework to learn new modes as needed to approximate a given training trajectories. Next, the constructivist framework for

learning from example was applied to the DARPA sponsored LAGR project. The LAGR project motivated a need for new solutions not relying on differentiability assumptions (as the variational approach does), which was addressed by posing the learning problem as an combinatorial optimization problem, and we presented an algorithm for solving this problem using a hill climbing method.

- *Multi-modal, multi-dimensional (M^3D) systems:* We addressed the optimal control of multi-modal systems with infinite dimensional constraints. We formulated the constraints as M^3D systems, where the dimensions of the state and control spaces change between dimensions to account for the constraints, to ease the computational burdens associated with traditional methods. We derived the optimality conditions for this formulation and presented an algorithmic framework for the optimal control of M^3D systems.

- *Robotics applications:* We used multi-modal control strategies to develop effective navigation strategies for autonomous mobile robots. Verified the theoretical results by conducting simulated experiments using Matlab and actual experiments in a lab setting using the Magellan Pro mobile robot platform. Moreover, we used human operated training runs to develop effective navigation strategies following the constructivist framework for learning from example. We successfully used these results on the LAGR robot to learn effective strategies for the LAGR competition.

The publications associated with these contributions are [72, 91, 92, 93, 94, 95, 96, 97]. In closing, the main strength of multi-modal control lies in breaking up complex control task into simpler tasks. This idea of designing individual modes with respect to particular control tasks and then sequencing these modes to achieve the overall desired behavior helps modularize the control system. This has the same effect on complex control systems that object-oriented programming has for large-scale

computer programs, namely it allows greater simplicity, flexibility, and adaptability.

7.2 Extensions

The main focus of this thesis has been to develop theoretical aspects of optimality in multi-modal control. In particular, we developed theoretical algorithms in a general setting to address some key research issues arising in optimal, multi-modal control, and used these theoretical results to obtain successful navigation strategies for autonomous mobile robots. However, there are still a number of research issues that remain unexplored and could be addressed further. In this section, we discuss some of these open issues with particular focus on the application to mobile robot navigation.

The algorithms in this thesis were developed under the assumption that sufficient knowledge of system dynamics, initial conditions, and the environment was available. Using this knowledge, optimal control programs are learned in Chapter 2 to complete desired tasks. The computation of the control programs, in this case, takes place off-line, while the system executes the programs on-line. In such a setting, the control system may be susceptible to disturbances or variations to the assumptions about the dynamics, initial conditions, and the environment (which is inevitably encountered when mobile robots operate autonomously). The optimal control program should be interpreted as a high-level plan over a set of available modes. In this case, the disturbances in the dynamics can be handled by a low-level controller. At the end of Chapter 2, we conducted a robustness analysis to show that the learning algorithm can account for small errors in the estimate of the initial condition; however, the sensitivity of the algorithm with respect to errors in the environment is still unexplored. It is our belief that since our algorithm plans over a set of feedback controllers, it should be more robust to unexpected obstacles than traditional path planners. In fact, robustness to unexpected obstacles is one of the strengths of the reactive approach as opposed to purely deliberative approach (see [14] for a more thorough discussion on

these different approaches). However, more research must be done to substantiate this belief. Moreover, taking cues from deliberative approaches such as the D^* path planning algorithm (see [51]), that updates the optimal path dynamically whenever unknown obstacles are encountered, it would be possible to extend our learning algorithm to update the control programs on-line whenever unmodelled obstacles are detected. The abundance of literature on reinforcement learning in unknown environments (see [52, 53, 55, 58]) can facilitate this extension.

So far, we have outlined some extensions to the learning algorithm, which focused on making the algorithm more useful in dynamic environments that typically arise in mobile robot navigation. We can make similar changes to the adaptive multi-modal control framework. Recall that adaptive multi-modal control was introduced as way of using experience from previous tasks to make the multi-modal control system more expressive with the promise of improving the overall performance of the system. The idea here was to improve the mode set off-line between different tasks. A natural extension to this would be to make this adjustment on-line. In particular, if we encounter recurring mode fragments during a particular run, we can try to find replacement modes, using the techniques outlined in Chapter 4, to improve the performance of the system on-line. Once such replacement modes are found, then the optimal policy can be dynamically updated using the enhanced mode set.

Finally, we will conclude with a few comments concerning the implementation of multi-modal control on mobile robot platforms. As mentioned in the introduction, there are many proposed frameworks for modelling and simulation of multi-modal systems. In this thesis, we stayed within the MDL framework since this framework is well suited for analysis and development of multi-modal control systems. However, this architecture may not be well-suited for implementation on robotic platforms. On Hybrid automata may, on the other hand, be better suited for implementation since it provides a truly reactive control architecture without the necessity of computing

control programs. In this case, the interrupts would automatically determine the next mode of operation for the system, thus eliminating the need for specification of control programs. As such, it would beneficial to use the MDL framework for the design and simulation of the multi-modal system. Once the performance of the system has been verified in simulations, the multi-modal system can be implemented on the robotic platform using a hybrid automaton. We addressed this problem of translating the multi-modal system from a MDL framework to a hybrid automata architecture in [96], where hybrid automata are generated from MDL mode strings. To this end, more research on implementation and on-line computations must be conducted to fully utilize multi-modal control for mobile robot navigation.

REFERENCES

[1] M. Egerstedt. "On the Specification Complexity of Linguistic Control Procedures," *International Journal of Hybrid Systems*, Vol. 2, No. 1-2, pp. 129-140, March & June, 2002.

[2] M. Egerstedt and R.W. Brockett. "Feedback Can Reduce the Specification Complexity of Motor Programs," *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, pp. 213-223, Feb. 2003.

[3] R.W. Brockett. "On the Computer Control of Movement," *IEEE International Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.

[4] R.W. Brockett. "Hybrid Models for Motion Control Systems," *Perspectives in Control*, H. Trentelman and J. C. Willems, Eds, Birkh, Boston, pp. 29-54, 1993.

[5] E. Frazzoli. "Explicit Solutions for Optimal Maneuver-Based Motion Planning," *IEEE Conference on Decision and Control*, 2003.

[6] E. Frazzoli, M. A. Dahleh and E. Feron. "Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries," *IEEE Transaction on Robotics and Automation*, Vol. 21(6), pp. 1077-1091, 2005.

[7] A. Bicchi, A. Marigo and B. Piccoli. "On the Reachability of Quantized Control Systems," *IEEE Transactions on Automatic Control*, Vol. 4(47), pp. 546-563, April 2002.

[8] A. Bicchi, A. Marigo and B. Piccoli. "Encoding Steering Control with Symbols," *IEEE International Conference on Decision and Control*, pp. 3343-3348, 2003.

[9] D. Hristu-Varsakelis and R. Brockett. "Experimenting with Hybrid Control," *Control Systems Magazine*, Vol. 22, No. 1, pp. 82-95, Feb. 2002.

[10] D. Hristu-Varsakelis, M. Egerstedt and P.S. Krishnaprasad. "On The Structural Complexity of the Motion Description Language MDLe," *IEEE Conference on Decision and Control*, Dec. 2003.

[11] L. Crawford and S.S. Sastry. "Learning Controllers for Complex Behavioral Systems," *Neural Information Processing Systems Tenth Annual Conference*, 1996.

[12] V. Manikonda, P.S. Krishnaprasad and J. Handler. "Languages, Behaviors, Hybrid Architectures and Motion Control," *Mathematical Control Theory*, pp. 199-226, 1998.

[13] C. J. Tomlin, G. J. Papas, J. Kotsecka, J. Lygeros and S. S. Sastry. "Advanced Air Traffic Automation: A Case Study in Distributed Decentralized Control," *Control Problems in Robotics, Lecture Notes in Control and Information Sciences* 230, Springer-Verlag, London, 1998.

[14] R. Arkin. *Behavior-Based Robotics*, MIT Press, Cambridge, Massachusetts, 1998.

[15] R. Brooks. "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. 2(1), pp. 14-23, 1986.

[16] T. A. Henzinger. "The Theory of Hybrid Automata," *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pp. 278-292, 1996.

[17] J. Lygeros, C. Tomlin and S. Sastry. "Controllers for Reachability Specications for Hybrid Systems," *Automatica*, Vol. 35(3), March 1999.

[18] V. Manikonda, P.S. Krishnaprasad and J. Handler. "A Motion Description Language and a Hybrid Architecture for Motion Planning with NonHolomic Robots," *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 2021-2028, May 1995.

[19] H.S. Witsenhausen. "A Class of Hybrid-State Continuous-Time Dynamic Systems," *IEEE Transactions on Automatic Control*, Vol. AC-11, No. 2, pp. 161-167, April 1966.

[20] M.S. Branicky, V.S. Borkar and S.K. Mitter. "A Unified Framework for Hybrid Control: Model and Optimal Control Theory," *IEEE Transactions on Automatic Control*, Vol. 43(1), pp. 31-45, Jan. 1998.

[21] A. Nerode and W. Kohn. "Models for Hybrid Systems: Automata, Topologies, Stability," *Hybrid Systems: Lecture Notes in Computer Science*, Springer-Verlag, Vol. 736, pp. 317-356, 1993.

[22] R. Alur, C. Courcoubetis, T.A. Henzinger and P-H Ho. "Hybrid automata: An algortihmic approach to the specification and verification of hybrid systems," *Hybrid Systems: Lecture Notes in Computer Science*, Springer-Verlag, Vol. 736, 1993.

[23] R. Alur, T.A. Henzinger and E.D. Sontag. "Hybrid Systems III: Verification and Control," *Lecture Notes in Computer Science*, Springer-Verlag 1996.

[24] M.S. Branicky. "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems," *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 1998.

[25] T. A. Henzinger, P. W. Kopke, A. Puri and P. Varaiya. "What's Decidable About Hybrid Automata?," in *Journal of Computer and System Sciences*, Vol. 57, pp. 94-124, 1998.

[26] S. Hedlund and A. Rantzer. "Optimal Control of Hybrid Systems," *IEEE Conference on Decision and Control*, pp. 3972-3977, 1999.

[27] M.S. Shaikh and P.E. Caines. "On the Optimal Control of Hybrid Systems: Optimization of Trajectories, Switching Times and Location Schedules," *In Proc. of the 6th International Workshop on Hybrid Systems: Computation and Control*, Prague, The Czech Republic, 2003.

[28] P. Tabuada and G.J. Pappas. "From Discrete Specifications to Hybrid Control," *IEEE Conference on Decision and Control*, Dec. 2003.

[29] X. Xu and P. Antsaklis. "Optimal Control of Switched Systems Based on Parameterization of the Switching Instants," *IEEE Transactions On Automatic Control*, Vol. 49(1), Jan. 2004.

[30] D. Hristu-Varsakelis, P.S. Krishnaprasad, S. Andersson, F. Zhang, P. Sodre and L. D'Anna. "The MDLe Engine a Software Tool for Hybrid Motion Control," *ISR Technical Report*, 2000-54, 2000.

[31] M. Egerstedt, Y. Wardi and F. Delmotte. "Optimal Control of Switching Times in Switched Dynamical Systems," *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[32] A. Guia, C. Seatzu and C. V. Der Mee. "Optimal Control of switched autonomous linear systems," *IEEE Conference on Decision and Control*, 1999.

[33] X. Xu and P.J. Antsaklis. "An Approach for Solving General Switched Linear Quadratic Optimal Control Problems," *IEEE Conference on Decision and Control*, pp. 2478-2483, 2001.

[34] X. Xu and P.J. Antsaklis. "Optimal Control of Switched Systems via NonLinear Optimization Based on Direct Differentiations of Value Functions," *International Journal of Control*, 2002

[35] A. Bemporad and M. Morari. "Control of Systems Integrating Logic, Dynamics, and Constraints," *Automatica*, vol. 35, pp. 407-427, 1999.

[36] M.S. Shaikh and P. Caines. "On Trajectory Optimization for Hybrid Systems: Theory and Algorithms for Fixed Schedules," *IEEE Conference on Decision and Control*, Las Vegas, NV, Dec. 2002.

[37] H. Axelsson, Y. Wardi and M. Egerstedt. "Transition-Time Optimization for Switched Systems," *Proc. of IFAC World Congress*, Prague, The Czech Republic, July 2005.

[38] M. Egerstedt, Y. Wardi and H. Axelsson. "Transition-Time Optimization for Switched Systems," *IEEE Transactions on Automatic Control*, Vol. 51, No. 1, pp. 110- 115, Jan. 2006.

[39] M. Boccadoro, Y. Wardi, M. Egerstedt and E. Verriest. "Optimal Control of Switching Surfaces in Hybrid Dynamical Systems," *Journal of Discrete Event Dynamic Systems*, Vol. 15(4), pp. 433-448, 2005.

[40] S.J. Bradtke, B.E. Ydstie and A.G. Barto. "Adaptive Linear Quadratic Control Using Policy Iteration," *American Control Conference*, pp. 3475-3479, 1994.

[41] B. Lincoln and A. Rantzer. "Optimizing Linear Systems Switching," *IEEE Conference on Decision and Control*, 2001.

[42] K. Morgansen and R.W. Brockett. "Optimal Regulation and Reinforcement Learning for the Nonholonomic Integrator," *Proc. of the American Control Conference*, pp. 462-466, June 2000

[43] T. Balch and R.C Arkin. "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transaction on Robotics and Automation*, vol 14, pp 926-939, 1998.

[44] M. Egerstedt. "Behavior Based Robotics Using Hybrid Automata," *Lecture Notes in Computer Science: Hybrid Systems III: Computation and Control*, pp. 103-116, Pittsburgh, PA, Springer-Verlag, March 2000.

[45] J.E. Hopcroft and G. Wilfong. "Motion of Objects in Contact," *The International Journal of Robotics Research*, Vol. 4, pp. 32-45, 1986.

[46] P. Cheng, Z. Shen and S. M. LaValle. "Using Randomization to Find and Optimize Feasible Trajectories for Nonlinear Systems," *Proc. Annual Allerton Conference on Communications, Control, Computing*, pp. 926-935, 2000.

[47] J.C. Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[48] D. Wooden and M. Egerstedt. "Oriented Visibility Graphs: Low-Complexity Planning in Real-Time Environments," *IEEE Conference on Robotics and Automation*, Orlando, FL, May 2006.

[49] M. Zefran and V. Kumar. "Planning Smooth Motions on SE(3)," *IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April 1996.

[50] M. Zefran and V. Kumar. "A Variational Calculus Framework for Motion Planning", *International Conference on Advanced Robotics*, Monterey, CA, July 1997.

[51] A. Stentz. "Optimal and Efficient Path Planning for Partially-Known Environments," *IEEE International Conference on Robotics & Automation*, pp. 3310-3317, 1994.

[52] L.P. Kaebling, M.L. Littman and A.W. Moore. "Reinforcement Learning: A Survey," *Journal Of Artificial Intelligence Research*, Vol. 4, pp. 237-285, 1996.

[53] C.J.C.H. Watkins and P. Dayan. "Q-Learning," *Machine Learning*, Vol. 8(3/4), pp. 257-277, May 1992.

[54] T. Jaakkola, M.I. Jordan and S.P. Singh. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms", in *Neural Computation* Vol. 6(6), 1994.

[55] R.S. Sutton and A.G. Barto. *Reinforcement Learning, An Introduction,* MIT Press, Cambridge, MA, 1998.

[56] A. Bhatia and E. Frazzoli. "Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems." *Hybrid Systems: Computation and Control. Springer-Verlag*, 2004.

[57] M. Egerstedt. "Linguistic Control of Mobile Robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, Oct. 2001.

[58] R.S. Sutton. "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," *Neural Information Processing Systems*, 1996.

[59] R. C. Thompson. "Lecture 10 : Part I - Convergence domains of the Campbell Baker Hausdorff formula," in *John Hopkins Lecture Notes*, 1988.

[60] W. Rossmann. *Lie Groups: An Introduction Through Linear Groups*, Oxford University Press, Chapter 1.3, 2002.

[61] T. T. Georgiou. "Relative Entrophy and the Multivariable Multidimesional Moment Problem," *IEEE Transaction on Information Theory*, Vol. 52(3), pp. 1052-1066, March 2006.

[62] L. Armijo. "Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives," *Pacific Journal of Mathematics*, Vol. 16, pp. 1-3, 1966.

[63] M. Sugeno. "An Introductory Survey of Fuzzy Control," *Information Sciences*, Vol. 36, pp. 59-83, 1985.

[64] D. Driankov, H. Hellendoorn and M. Reinfrank. *An Introduction to Fuzzy Control*, Springer, Berlin, 1993.

[65] D. Park, A. Kandel and G. Langholz. "Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24(1), Jan. 1994.

[66] S.C. Lin and Y. Chen. "Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control," *IEEE Transactions on Evolutionary Computation*, Vol. 2, pp. 846-851, 1995.

[67] K. Tanaka, T. Hori and H.O. Wang. "New Robust and Optimal Designs for Takagi-Sugeno Fuzzy Control Systems," *IEEE International Conference on Control Applications*, Vol. 1, pp. 415-420, 1999.

[68] M. Egerstedt, T. Balch, F. Dellaert, F. Delmotte, and Z. Khan. "What are the Ants Doing? Vision-Based Tracking and Reconstruction of Control Programs," *IEEE International Conference on Robotics and Automation*, Barcelona, April 2005.

[69] A. Guillory, H. Nguyen, T. Balch and C. Isbell. "Learning Executable Agent Behaviors from Observation," *Proc. of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

[70] T. M. Jochem, D. A. Pomerleau and C. E. Thorpe. "Vision-Based Neural Network Road and Intersection Detection and Traversal," *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 344-349, 1995

[71] D. Kim, J. Sun, S. M. Oh, J. M. Rehg and A. Bobick. "Traversability Classification Using Unsupervised On-Line Visual Learning for Outdoor Robot Navigation," *IEEE Intl. Conference on Robotics and Automation*, Orlando, FL, May 2006.

[72] J. Sun, T. R. Mehta, D. Wooden, M. Powers, J. Regh, T. Balch and M. Egerstedt. Learning from Examples in Unstructured, Outdoor Environments, *Journal of Field Robotics*, Vol. 23. Issue 11-12 , pp. 1019-1036, Jan. 2007.

[73] D. Pomerleau. "ALVINN: An Autonomous Land Vehicle In an Neural Network," *Advances in Neural Information Processing Systems*, pp. 305-313, 1989.

[74] R. Sukthankar, D. Pomerleau and C. Thorpe. "A Distributed Tactical Reasoning Framework for Intelligent Vehicles," *Proc. of Intelligent Systems and Manufacturing*, 1997.

[75] B. Webb. "What Does Robotics Offer Animal Behaviour," *Animal Behaviour*, Vol. 60, pp. 545-558, 2000.

[76] I. Ulrich and I. Nourbakhsh. "Appearance-Based Obstacle Detection with Monocular Color Vision," *AAAI National Conference on Artificial Intelligence*, pp. 866-871, 2000.

[77] T. Balch, F. Dellaert, A. Feldman, A. Guillory, C. Isbell, Z. Khan, S. Pratt, A. Stein and H. Wilde. "How A.I. and Multi-Robot Systems Research Will Accelerate Our Understanding of Social Animal Behavior," *Proceedings of the IEEE*, Vol. 94, No. 7, pp. 1445-1463, July 2006.

[78] J. Michels, A. Saxena and A. Y. Ng. "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," *International Conference on Machine Learning*, 2005.

[79] J. Rosenblatt. "Maximizing Expected Utility for Optimal Action Selection under Uncertainty," *Autonomous Robots*, Vol. 9(1), pp. 17-25, 2000.

[80] F. Delmotte, M. Egerstedt and A. Austin. "Data-Driven Generation of Low-Complexity Control Programs," *International Journal of Hybrid Systems*, Vol. 4(1), pp. 53-72, 2004.

[81] G. Ferrari-Trecate, M. Muselli, D. Liberati and M. Morari. "A Clustering Technique for the Identification of Piecewise Affine and Hybrid Systems," *Automatica*, Vol. 39, pp. 205-217, 2003.

[82] R. Vidal, S. Soatto, Y. Ma and S. Sastry. "An Algebraic Geometric Approach to the Identification of a Class of Linear Hybrid Systems," *IEEE Conference on Decision and Control*, 2003.

[83] J. Bransford, A. L. Brown and R.R. Cocking. *How People Learn: Brain, Mind, Experience, and School*, Washington: National Academies Press, 2000.

[84] D. Wood. *How Children Think and Learn*, 2nd edition. Oxford: Blackwell Publishers Ltd, 1998.

[85] J. Rosenblatt. "DAMN: A Distributed Architecture for Mobile Navigation," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9(2), pp. 339-60, 1997.

[86] S. B. Williams, P. Newman, J. Rosenblatt, G. Dissanayake and H. Durrant-Whyte, "Autonomous Underwater Navigation and Control," *Robotica*, Vol. 19(5), pp. 481-496, 2001.

[87] T.D. Murphey and J.W. Burdick. "Feedback Control Methods for Distributed Manipulation Systems that involve Mechanical Contacts," *International Journal of Robotics Research*, Vol. 23, No. 7-8, pp. 763-781, 2004.

[88] E. I. Verriest. "Multi-Mode Multi-Dimensional Systems," *International Symposium on the Mathematical Theory of Networks and Systems*, Kyoto, Japan, July 2006.

[89] J.J de Koning, G de Groot and G. J van Ingen Schenau. "Ice Friction During Speed Skating," *Journal of Biomechanics*, Vol. 25(6), pp. 565-571, 1992.

[90] G. J van Ingen Schenau. "The Influence of Air Friction in Speed Skating," *Journal of Biomechanics*, Vol. 15(6), pp. 449-458, 1982.

[91] T. R. Mehta and M. Egerstedt. "Learning Multi-Modal Control Programs," *Hybrid Systems: Computation and Control*, Springer-Verlag, Vol. 3414, pp. 466-479, March 2005.

[92] T.R. Mehta, F. Delmotte and M. Egerstedt. 'Motion Alphabet Augmentation Based on Past Experience," *IEEE Conference on Decision and Control*, Dec. 2005.

[93] T.R. Mehta and M. Egerstedt. "An Optimal Control Approach to Mode Generation in Hybrid Systems," *Nonlinear Analysis: Theory, Methods and Applications*, Vol. 65(5), pp. 963-983, September 2006.

[94] T.R. Mehta and M. Egerstedt. "Optimal Membership Functions for Multi-Modal Control," *American Control Conference*, June 2006.

[95] T. R. Mehta, D. Young, E. Verriest and M. Egerstedt. "Optimal Control of Multi-Dimensional, Hybrid Ice-Skater Model," *American Control Conference*, July 2007.

[96] F. Delmotte, T.R. Mehta and M. Egerstedt. "MODEbox: A Software Tool for Obtaining Hybrid Control Strategies from Data," *IEEE Robotics and Automation Magazine*, To appear.

[97] T.R. Mehta and M. Egerstedt. "Multi-Modal Control Using Adaptive Motion Description Languages," *Automatica*, submitted Nov, 2006.