Construction and Automated Deployment of Local Potential Functions for Global Robot Control and Navigation

David C. Conner, Alfred A. Rizzi, and Howie Choset

CMU-RI-TR-03-22

November 2003

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This report presents a method for addressing integrated navigation and control tasks for constrained dynamical systems. Specifically, we focus on solving this problem for an idealized point robot subject to bounds on velocity and acceleration. The basic approach divides the overall task into discrete sub-tasks, and achieves those sub-tasks using feedback control policies. The method is based on sequential composition of safe, reliable, and robust feedback control policies.

This report presents an extension to the method of sequential composition that allows a new class of feedback control policies with goal sets, not just goal points. Where previous work only allowed for goal sets that were completely contained in the domain of another control policy, this extension allows the deployment of policies whose goal intersects the domains of multiple policies. This extension allows for a larger class of policies to be deployed, making it easier to build near globally convergent overall control policies.

This report also details the new local control policies, defined over cells in the configuration space. The policies cause a large subset of initial states to exit the cell in a specified manner. The resulting composition of local control policies induces a global control strategy that brings any initial condition contained in the union of the domains of the control policies the goal, provided that there is a single connected component of free space containing both the start and goal configurations. The underlying control policies are designed to respect environmental constraints such as obstacles, velocity bounds, and acceleration bounds. Control policies for fully actuated kinematic and dynamical systems are developed.

Contents

List of Figures

List of Symbols

Generic Terms

x	Vector quantity
$X(q)$	Vector field
A	Matrix
x	Scalar quantity
x_i	i^{th} component of a vector
$\mathcal{V}$	Set or collection
$\|\|x\|\|$	Euclidean norm (2-norm)
$\|\|\|A\|\|\|$	Spectral norm of matrix A

Vector Space Notation

$\mathcal{W}$	Work space of system
$\mathcal{Q}$	Configuration space of dynamical system
$\mathcal{X}$	State space of dynamical system
$\mathcal{O}_i$	i^{th} obstacle in the work space
$\mathcal{QO}_i$	i^{th} configuration space obstacle
$\mathcal{XO}_i$	i^{th} state space obstacle
$\mathcal{FS}$	Free space
$\mathcal{FS}_\mathcal{W}$	Free work space
$\mathcal{FS}_\mathcal{C}$	Free configuration space
$\mathcal{FS}_\mathcal{X}$	Free state space
q	point in configuration space $\mathcal{Q}$
q_b (q_d)	point in unit ball (disk)
$\dot{q}$	configuration velocity
$\ddot{q}$	configuration acceleration
$\mathcal{K}$	Cellular decomposition - collection of cells
$\mathcal{P}$	Cell (or polytope) in decomposition $\mathcal{K}$
$\mathcal{G}$, $\mathcal{G}(V,E)$	Adjacency (or connectivity) graph with vertex set V and edge set E
$\mathcal{P}_g$	Cell containing the goal point $q_g \in \mathcal{P}^g \subset \mathcal{FS}$, the *goal cell*
$\mathcal{B}$	Unit ball
$\mathcal{B}(q,\rho)$	Ball of radius ρ centered at q
$\mathcal{D}$	Unit disk (ball in plane)
$\mathbf{S}^n$	n-Dimensional sphere
n	Unit normal vector defining hyper plane $n \cdot x = d$
p	Point on the hyper plane specifying boundary $n \cdot x < n \cdot p$ or, center of polytope face

Control Policy Notation

$\succeq$	$\Phi_2 \succeq \Phi_1$ Policy 2 prepares policy 1
$\Phi_\mathcal{P}$	Component control policy defined for cell $\mathcal{P}$
$\Gamma'_\mathcal{U}$	Directed acyclic graph defining partial order
$\bigvee$	Composition Operator

Φ_S	Save control policy
Φ_A	Align control policy
Φ_T	Track control policy
Φ_F	Flow control policy $\Phi_F = \bigvee \{\Phi_T, \Phi_A\}$
Φ_G	Goal control policy $\Phi_F = \bigvee \{\Phi_T, \Phi_A\}$ with different vector field
$\mathcal{V}$	Set of permissible velocities $\|\|\dot{q}\|\| \leq \mathrm{V}_{\max}$
$\mathcal{A}$	Set of permissible accelerations $\|\|\ddot{q}\|\| \leq \mathrm{A}_{\max}$
$\mathcal{Z}$	Set of velocities aligned with vector field $\dot{q}^T X > 0$
$\mathcal{S}$	Savable set $\mathcal{S} \subset \mathcal{FS}_{\mathcal{X}}$
ζ_c	Collision ratio $\zeta_c = \frac{d_b}{d_c}$
d_b	Required braking distance given collision speed
d_c	Projected collision distance
μ	User defined collision avoidance ratio $\mu \in (0, 1)$
υ	Collision avoidance transition variable
λ	Vector field scaling constant
α	Vector field scaling constant for convergence
e	Velocity error vector $e = X(q) - \dot{q}$

1 Introduction

The goal of this research is to develop feedback control strategies that allow for automatic deployment of control policies in constrained environments, such that the resulting control system instantiates a natural, provably correct behavior for systems operating with their full dynamic capabilities. The classic navigation problem in robotics is to find a trajectory that takes the system from a start configuration to a specified goal configuration, while avoiding obstacles in the environment. By system we mean a mechanical system, capable of motion in its surrounding environment, with inputs used to specify the motion. The work in this report focuses on fully actuated, or *holonomic*, systems in $\mathbb{R}^n$. The inputs can take the form of velocities, in which case we refer to the system as *kinematic*, or accelerations (forces), in which case we refer to the system as *dynamical*.

Conventional robot architectures have separated the planning and control problem to a degree that provably correct planning algorithms offer no guarantees of dynamical performance. The problem of defining a trajectory that solves the navigation problem is often decomposed into a path planning problem, followed by a controls problem. However, a complete path may be impossible to follow for a dynamical system. Another typical approach is to operate far below the capabilities of the system in order to approximate kinematic behavior. We present a new approach to decomposing this problem so that the planning and controls problem is solved simultaneously by inducing a global control policy that brings any initial configuration to the goal provided that there is a single connected component of free-space containing both the start and goal configurations. Our approach is to define a palette of control policies, and a switching strategy among the individual control policies, such that the resulting composition simultaneously solves both the navigation and control problems. Our approach allows us to leverage the robustness afforded by feedback controls, while generating a globally convergent control policy.

Our task is to design a *control policy*, a set of rules and governing equations, that determines the control inputs to the system such that motion from the start to the goal is guaranteed to be safe, robust, and reliable with respect to system perturbations. The control policies considered in this research take the form of memoryless state feedback policies. The systems of interest are subject to dynamic constraints in the form of allowable motions and bounded control inputs. Costs such as elapsed time and energy consumption are often considered in *optimal control* theory; the goal is to design a control policy that minimizes the value of some cost function – often a function of elapsed time, or energy, or a combination of both. In our work we do not consider optimal control policies, but seek to design control policies that respect the fundamental dynamic constraints of allowable motions and control inputs, while at the same time *satisficing* from a time and energy standpoint [64]. By incorporating the constraints into the low-level control policies, and planning in the space of available control policies, we offer a methodology that is provably safe and correct in both kinematic and dynamical environments.

After introducing an overview of related work, we discuss the decomposition and planning problem inherent in our work. This serves to motivate the type and scope of our low-level control policies, while offering proof of the correctness of the control policy composition. We develop extensions to the method of sequential composition that allow feedback control policies with goal sets, not just goal points. Previous work allowed only for goal sets that were completely contained in the domain of another control policy, this report presents an extension that allows the deployment of policies whose goal set intersects the domains of multiple policies. This extension allows for a large class of policies to be deployed, making it easier to build near globally convergent overall control policies.

Next we develop the general form of the low-level control policies, and provide proofs of the applicability to a simple dynamical system. We then outline the development of hybrid switching control policies required to overcome dynamical constraints, such as acceleration limits, and increase the domain of attraction for the low-level control policies. We present specific constructions for policies defined over convex polytopes

in configuration space. We conclude by outlining the next steps in our research plan, which is to develop automated systems of control policy deployment for real mobile robots subject to non-holonomic constraints.

2 Related Work

As introduced earlier, the navigation problem in robotics has a long history, and a vast literature. Numerous techniques have been developed over the years in an attempt to solve this problem [42]. Researchers have typically broken the problem into different parts, solving one part and leaving the rest to others. In spite of the plethora of available techniques, no single technique has been universally accepted. Some techniques work only in ideal conditions, others solve local problems but not global problems. Techniques that deal with obstacles do not address non-holonomic constraints, and vice versa. Many of the techniques capable of generating a valid path or trajectory through a cluttered environment do not address convergence issues if the system starts off the desired path. The research presented in this document addresses these shortcomings in a bottom up manner for a limited class of holonomic systems by designing control policies that respect dynamic constrains on the system.

This section begins with a review of some of the relevant literature related to the problem of moving a robot or other dynamical system from one point to another in a cluttered environment. An overview of techniques used in the navigation and control of mobile robots is provided. The section concludes with a discussion of hybrid control theory and an overview of the method of sequential composition, which forms the foundation for this research.

2.1 Navigation and Control

The navigation problem can be divided into three classes:

i) path planning followed by trajectory generation or path following,

ii) local obstacle avoidance, or

iii) integrated trajectory planning.

This section considers each class in turn.

2.1.1 Path Planning

Path planning attempts to determine a valid path in the free space. Examples of path planning include navigation from start to goal, mapping, and coverage tasks. The focus of this review is on path planning for navigation tasks. The desired path is represented by a continuous function,

$$\tilde{q}(s) : [0,1] \rightarrow \mathcal{FS}_{\mathcal{Q}} \,,$$

that maps an interval to the free configuration space such that $q(0) = q_0$, the start configuration, and $q(1) = q_g$, the goal configuration [42]. A valid path, also called an *admissible* path, is a path in the free space that satisfies given constraints, such as curvature bounds. In general, path planning requires full knowledge of the environment [42]. The basic navigation problem ignores differential constraints, and assumes the system can move in any direction. Latombe [42] terms this type of system a *free flying robot*, while others use the term *holonomic* robot [27]. The lack of motion constraints makes the planning problem easier.

The vastness of the path planning literature renders it impossible to give a full accounting in this section. Therefore, this section focuses on the why and results of path planning. For the "how," and further references, the reader is referred to the accepted reference for motion planning, Latombe's *Robot Motion Planning* [42].

Loosely, we will divide the path planning methods into two camps – those that provide a continuous path and those that provide a discrete set of way points along the desired path.

A continuous path from initial configuration to goal can be found using roadmap methods, such as Canny's silhouette method, visibility graphs, Voronoi graphs, and freeway methods [42, 23]. Roadmaps are one dimensional subsets of the free configuration space that have the properties of accessibility, connectivity, and departability. The robot can access the roadmap from any initial configuration (accessibility), navigate along the roadmap to a point near the goal (connectivity), and depart the roadmap near the goal where a path to the goal can be found (departability).

Potential fields are also used to find a continuous path by performing gradient descent along the potential function beginning at the initial configuration [31, 42]. A potential field can be constructed by summation of an attractive potential, which increases with distance from the goal, and a repulsive potential, which increases as obstacles are approached. Unfortunately, for general potential fields, the path may only lead to a local minima that does not correspond to the overall goal [31, 40, 36]. Rimon and Koditschek [57, 55, 58, 56] developed an analytic method of finding a *navigation function*, which is a potential field with a single minimum at the goal. Their approach mapped a class of obstacles to a model space, and defined the navigation function on the model space. The induced potential on the configuration space was guaranteed to be a navigation function as well. Connolly *et al.* [17, 16, 18] developed a potential field with a unique minimum based on a numeric solution to Laplace's heat equation, which yields a harmonic potential function.

The second class of methods used to generate a path is to approximate the continuous path with a set of discrete way points. Approximate cellular decompositions are often used to define the region corresponding to the discrete points [42]. In $\mathbb{R}^2$, a grid representation is common, both because of the ease of representation and its utility as a method for merging sensor data [49]. Each grid point corresponds to a region of free space or obstacle. The value associated with each grid point may represent the cost to traverse the cell, or represent the probability that the region is occupied by an obstacle. The path is found by searching for adjacent grid points with the minimum associated cost to traverse the cell. The grid discretization extends to hyper-cubes in higher dimensions, although other region shapes corresponding to each grid point are possible. Approximate cell decompositions are only *resolution complete*, and may require refinement of the resolution [42]. For moderately complex environments, or configuration space dimensions greater than two or three, the number of cells to search may become intractable, which has given rise to probabilistic methods.

Probabilistic RoadMaps (PRM) sample the free configuration space at some specified number of locations [29, 46]. The roadmap is built as a graph, where nodes in the graph corresponding to the sample points are connected to their neighbors if an admissible path in the free space exists between the two corresponding sample points. A fast, strictly local planner is used to query sample points to see if an admissible path exists. If admissible paths between the start and goal points can be found to points in the graph, the complete path is found by searching the graph corresponding to the roadmap. While none of the PRM techniques are complete, they are *probabilistically complete* in that as the number of random sample points and execution time goes to infinity a solution, if it exists, will be found with probability one.

While solving the path planning problem generates an admissible path, it does not actually cause the system to move toward the goal. For this, a control policy must be used to map the current state and desired position on the path to the system inputs. Given the path, the development of such a control policy is known as the path following problem.

2.1.2 Path and Trajectory Following

Given an admissible path, possibly constructed by one of the methods given above, the task becomes to design the control inputs to move the system along the path from start to goal. For an idealized system, without differential constraints or input bounds, this is relatively straightforward. If the system is kinematic,

we have

$$\begin{aligned} \dot{q} &= f(u) \\ &= \tilde{q}'(s)\dot{s}, \end{aligned} \tag{1}$$

where $\dot{s}$ is the time derivative of the time scaling $s(t) : [0,T] \rightarrow [0,1]$. In the simpliest case of a constant speed trajectory, $\dot{s} = \frac{1}{T}$, where $T = \frac{L}{\mathrm{V_{max}}}$ with L the total path length and $\mathrm{V_{max}}$ the desired speed.

If the system is dynamical, $\ddot{q} = f(u)$, the path following problem can become more complicated. Acceleration bounds may prevent the system from exactly following the path for arbitrary velocities. If the path is continuous ($\tilde{q}(s) \in \mathrm{C}^0$), but has a discontinuous first derivative ($\tilde{q}(s) \notin \mathrm{C}^1$), then there is no bounded control input capable of following the path. As such, the admissiblity of path must consider the dynamics of the system that will attempt to follow the path. For paths with continuous second derivatives, $\tilde{q}(s) \in \mathrm{C}^1$, there exists an algorithm to determine the time optimal time-scaling for which the path can be followed [4, 65].

In spite of these methods of control, systems rarely follow paths exactly, whether due to differential constraints, limited actuation, or the general imprecision of the motors and control policies [43]. Therefore, paths that are perfectly valid from a planning perspective, may lead to collisions when applied to a physical robot.

One approach to avoiding the collision problem is to plan in the space of admissible trajectories (*trajectory planning*), as opposed to decoupled path planning and motion control [45]. However, before moving on to integrated trajectory planning, we discuss a more basic method of obstacle avoidance, which can be coupled with path following or used as a navigation method by itself.

2.1.3 Obstacle Avoidance

The most basic form of navigation is moving toward the goal while avoiding local obstacles. Although all of the local obstacle avoidance methods described in this section suffer from the problem of "local minima" by virtue of their being based on local information, the methods are able to operate in real time, while moving in cluttered and changing environments. Often these methods are layered on one of the global path planning algorithms described earlier, where points along the desired path serve as intermediate goals to guide the local algorithm to the global goal [41, 70]. This form of local obstacle avoidance helps to overcome some of the problems associated with path following error. The coupling of local obstacle avoidance with global path planning allows planning system to operate on a coarser scale, which speeds the planning stage making this approach feasible for real-time operation [41].

The potential field method can serve as both a local obstacle avoidance method and a global planning method, as discussed earlier; here, the focus in on local obstacle avoidance [31, 40]. The basic approach calculates an attractive potential with respect to the distance to the (intermediate) goal, and repulsive potentials with respect to obstacles, with control effort applied along the gradient vector [31]. The input magnitudes used for control are sometimes based on the magnitude of the potential gradient, and sometimes on some specified magnitude in the direction of the gradient.

Several variations of the basic potential field method of obstacle avoidance have been developed. Krogh [40] defines a *generalized potential field* that includes velocity information, along with acceleration bounds, in the potential calculation. The acceleration direction field is a scaled linear combination of gradient vectors of the generalized potential field. Kim and Khosla [32] defined potential functions using harmonic functions based on the solution to Laplace's equation. Their method used source and sink functions, along with a constant flow condition, to solve the obstacle avoidance problem. The method required tuning of parameters within certain bounds to prevent failure.

All of these potential methods can result in a stable (zero gradient) points at local minima other than at the global goal, and thus require higher level information [42, 31, 40]. Additional way points, random motion,

and equipotential "wall" following are techniques used to escape local minima. In spite of the problem of local minima, potential field techniques are still in widespread use [66].

Borenstein and Koren [6] attempted to overcome local minima problems found in their implementation of a *Virtual Force Field* potential method with a method called the *Vector Field Histogram* (VFH). The VFH method created a polar histogram of the *polar obstacle density* function, and sought to steer through the open corridor closest to the goal heading. The VFH method was extended to account for vehicle width and trajectory with the VFH+ method [69]. The next extension, termed VFH*, allowed for semi-global planning by virtue of simulation of a finite number of steps of the VFH+ algorithm and an A* search of the resulting tree [70]. This last extension was shown to overcome several situations that are problematic for potential fields, and the older VFH and VFH+ algorithms.

Another approach to obstacle avoidance is based on searching through a set of admissible trajectories to find one that avoids the obstacle while generally moving toward the goal. Two methods, the *Curvature Velocity Method* (CVM) [63] and the *Dynamic Window Approach* (DWA) [25], take this approach. In both, the feasible trajectories are encoded as circular arcs. The search space was limited by constraints, and focused on short term collision avoidance. Short comings in both methods, such as missing narrow corridors, led to further extensions [34, 67].

All of these local obstacle avoidance methods trade reactive speed and simplicity for global performance guarantees. As local methods, these techniques are most effective when incorporated with the global path planning methods. While global paths make it more likely that a complete trajectory can be found, the local methods may invalidate feasible paths, such as those through a narrow doorway in the case of potential fields.

2.1.4 Trajectory Planning

The decoupling of the planning stage with the motion control leads to several problems, as discussed in Section 2.1.2. This has led to the integration of this coupled problem into a single step, which is termed trajectory planning [45]. Trajectory planning can be loosely grouped into potential field methods and kinodynamic planning methods.

In addition to being used for path planning and obstacle avoidance, potential field methods are used to trajectory control. For kinematic systems, setting the velocity equal to the negative gradient will move the system to a local minimum. For dynamical systems, the addition of a dissipative term in the control law will result in convergence to a local minimum for any system whose total energy is less than or equal to the potential on the boundaries of the free configuration space [37, 38].

To avoid the problems associated with local minima, Koditschek [37] defines a *navigation function* to be a twice differentiable potential function that is Morse and *admissible*. In this context, admissible is defined as taking a uniform non-zero value on the boundaries, and a unique minimum on the free configuration space. Morse functions are those where all critical points are non-degenerate, that is the matrix of second derivatives (Hessian) is non-singular. Rimon and Koditschek [57] present a constructive method for creating a navigation function for any space with obstacles that can be represented as overlapping unions of star shaped obstacles. The method maps the environment to a sphere world with a series of diffeomorphic transformations, which yield disjoint spheres as obstacles. A navigation function for the sphere world is given. Their work proves that a mapping via diffeomorphism preserves the properties of the navigation function [57, 35].

Connolly *et al.* [17, 16, 18] generated a navigation-like function, which was admissible but not necessarily Morse. The function, which they termed a *weak navigation function* was found through numerical evaluation of Laplace's equation, which yields a *harmonic* potential function. Although not Morse due to the possibility of isolated degenerate saddle points, the function is uniformly maximal on the boundary, and has a unique minimum [17]. Since the saddle point is detectable, any perturbation away from the saddle will lead away from the saddle with probability one. Connolly *et al.* [19] used Hamiltonian dynamics to derive an energy reference control policy based on harmonic potential functions. The control policy can be used to drive the system to the goal ($H = 0$) or to drive the system along some equipotential manifold ($H = H_{ref}$)

contained in the state space. The induced manifold is under-constrained, and allows for the specification of additional constraints.

Most potential methods used in control do not account for input bounds, and typically have unbounded potential at the obstacle boundary [57]. Although Rimon and Koditschek's method has bounded potential, it still does not account for arbitrary bounds on the inputs [57]. Potential methods that do not account for the total energy may allow collisions for certain initial conditions [37]. For example, a system near a boundary moving towards the boundary may not stop before collision with the boundary under gradient control if the total energy is not respected.

Another method of trajectory planning, termed *kinodynamic* planning, specifically considers the navigation problem in light of *dynamic constraints*, such as acceleration and velocity bounds, and *kinematic constraints* such as obstacles [12, 13, 21]. The method also allows for *strictly kinodynamic constraints* that do not fall into either category, such as speed dependent obstacle avoidance margin. A solution to the kinodynamic planning problem is a mapping from time to generalized forces or accelerations. The original formulation dealt with holonomic systems, and did not not address non-holonomic constraints. The approach given in [21] transformed the continuous state space navigation problem into a graph search problem by generating a set of regularly spaced nodes corresponding to states reachable from a given state with the application of a τ-*bang* of the maximum acceleration applied for time step τ in each basis directions. This gives an safe approximation to the time optimal path, where the closeness of the approximation is governed by $0 < \epsilon < 1$. If a given node lies near an obstacle, as defined by a safety parameter δ, then the node is pruned from the graph. The algorithm was refined and extended to provide an exact solution of a point mass moving in a plane with polygonal obstacles [13].

Dynamic programming, which is another method of solving kinodynamic planning problems [45], leverages Bellman's principle of optimality, which states

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [3].

Numerical dynamic programming techniques are used to solve the kinodynamic planning problem, often based on a cost-to-go iteration scheme [45]. Atkeson *et al.* [1] apply variations of the dynamic programming technique used in reinforcement learning to kinodynamic problems. The solution to a dynamic programming problem results in a global control policy that specifies the optimal control action for a given state. Although these techniques are extremely powerful, they suffer from the well known *curse of dimensionality*, and are limited to low dimensional state spaces or coarse approximations. Roy and Thrun [61] use dynamic programming to plan a path in a low dimensional configuration space, then use gradient ascent to locally optimize the trajectory in the full dimensional continuous space.

Because of the relatively high dimension of the state space, twice the dimension of the configuration space, randomization techniques are attractive for use in kinodynamic planning problems [43, 45, 45, 53, 33, 44]. Mazer *et al.* [53, 52] present the *Ariadne's Clew Algorithm* [1]. The technique uses two algorithms – SEARCH and EXPLORE. SEARCH optimizes a parameterization of a trajectory through the free space from some initial point, minimizing the final distance to the goal. If the final distance is zero, the algorithm terminates and returns the path. If the distance is non-zero, the EXPLORE algorithm adds a landmark from the initial point, maximizing the distance to other landmarks. The design of the algorithm is well suited to parallel execution, and has been implemented on parallel computer architectures. The path optimization was performed using genetic algorithms, which are also well suited to parallel execution.

Lavalle and Kuffner [43, 44] introduce the Rapidly-exploring Random Tree (RRT) as a probabilistically complete method of solving high dimensional kinodynamic planning problems. The basic RRT builds a tree

[1] Ariadne's Clew is named for the ball of thread given to Thesus by Ariadne in Greek mythology; Thesus unwound the string in the Labyrinth as a navigation aid.

by randomly choosing a point in state space, finding the node in the tree closest to the chosen point, and applying a control for a unit of time to move the system from the node towards the chosen state. The new point generated by applying the control for a unit of time is then added to the tree. This biases the formation of new nodes toward unexplored space, while preserving randomness. To facilitate convergence, two RRTs are constructed – one originating at the start state and one at the goal state. If a node from the first tree can reach a node from the second tree, the path is found by graph search over the trees. Unlike other planning techniques, the RRT readily extends to systems with non-holonomic constraints. New points are only generated by admissible motions from a node to the new point. As with PRM, the resulting paths are feasible; however, in general the paths are not optimal [45]. Kindel *et al.* [33] use similar techniques in the state-time space to solve kinodynamic planning problems with moving obstacles, where the obstacle trajectories are known a priori. The method forms a directed tree, oriented along the time direction. Relatively fast execution time enables replanning on the fly, which effectively removes the assumption that obstacles have known trajectories.

Although, the techniques using dynamic programming can generate the optimal policy for a discrete approximation over the entire free space, this is only tractable for low dimensional states spaces and relatively coarse approximations. Kinodynamic planning on the other hand generates the trajectory, but is not amenable to feedback policies. For any given trajectory, modeling errors and control imprecision will cause the actual trajectory to deviate from the desired trajectory. This will require replanning, and invalidates the guarantee of collision avoidance.

2.2 Switched Control Systems

Regardless of how the desired trajectory is arrived at, the implementation falls to the control system. The control system is responsible for determining the active control policy, and specifying the inputs to the controller hardware as dictated by the active control policy. This section provides an overview of so called *hybrid control systems* – systems that combine discrete and continuous dynamics. Finally, this section presents an overview of a specific technique for organizing hybrid control systems called *sequential composition*. Sequential composition forms the foundation for the techniques developed in this report.

2.2.1 Hybrid Control

The use of discrete behavior modules, and the switching between them, results in what is known as a hybrid control system [8, 26, 10]. The hybrid control system is defined as a system containing both discrete and continuous elements. Hybrid systems theory provides a conceptual framework for analyzing the performance of a robotic system under the influence of a given control architecture. There are two main formalisms for describing hybrid systems: hybrid automata [26] and the general hybrid dynamical systems (GHDS) framework [8]. The hybrid automata and GHDS formalisms represent a convergence of paradigms arising from computer science and control theory respectively. For our purposes, the formalisms are interchangeable. A special sub-class of hybrid systems, called switched systems, has also been defined [47].

The hybrid automata model of [26] emerged from a computer science background, and has the framework of a discrete automata. Associated with each node in the automata is a n-dimensional set of real variables that represent the continuous component of the model. The edges connecting each node are called control switches, and are associated with events triggered by jump conditions or guards dependent on the discrete node and continuous variables. In the simplest case, a node could correspond to a discrete behavior, where the a control policy induces the desired behavior. The control policy associated with a given behavior induces a specific dynamic response modeled by the continuous state space associated with each node of of the hybrid automata. Figure 1 shows a basic example.

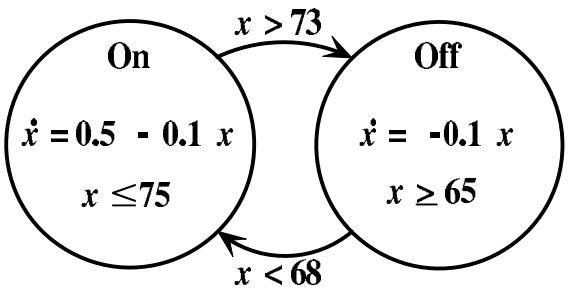


Figure 1: A simple hybrid automata model of a thermostat. The real variable x denotes the temperature of the room, with temperature dynamics associated with each node. From the On mode, the jump condition $x > 73$ states that the system *may* switch to Off mode in response to an external control signal. The guard condition $x \leq 75$ forces a switch if the temperature goes higher than 75.

Another formalism for describing hybrid systems is the *Controlled General Hybrid Dynamical System (CGHDS)*. As defined by Branicky [8, 10], a controlled general hybrid dynamical system H_c is a system

$$H_c = [\mathbf{Q}, \mathbf{\Sigma}, \mathbf{A}, \mathbf{G}, \mathbf{V}, \mathbf{C}, \mathbf{F}] \ ,$$

where

- $\mathbf{Q}$ is a set of index states,
- $\mathbf{\Sigma} = \{\Sigma_q\}_{q \in \mathbf{Q}}$ is a collection of controlled dynamical systems.
- $\mathbf{A} = \{A_q\}_{q \in \mathbf{Q}}, A_q \subset \mathcal{X}_q$ for each $q \in Q$ is the collection of *autonomous jump sets*.
- $\mathbf{G} = \{G_q\}_{q \in \mathbf{Q}}$ where $G_q : A_q \times V_q \to \mathcal{S}$ is the *autonomous jump transition map*, parameterized by the transition control set V_q, a subset of the collection $\mathbf{V} = \{V_q\}_{q \in \mathbf{Q}}$; they are said to represent the discrete dynamics and controls.
- $\mathbf{C} = \{C_q\}_{q \in \mathbf{Q}}, C_q \subset \mathcal{X}_q$ for each $q \in Q$ is the collection of *controlled jump sets*.
- $\mathbf{F} = \{F_q\}_{q \in \mathbf{Q}}$ where $F_q : C_q \to 2^{\mathcal{S}}$ is the collection of *controlled jump destination maps*.

The hybrid state space of H_c is defined as $\mathcal{S} = \bigcup_{q \in \mathbf{Q}} \mathcal{X}_q \times \{q\}$. Figure 2 shows a graphical representation. States that enter A_q jump to another hybrid state according to G_q and V_q; states that enter C_q may jump if commanded. The General Hybrid Dynamical System is, as its name implies, quite general. If $|\mathbf{Q}|$ is finite, each $\mathcal{X}_q$ is a subset of $\mathbb{R}^n$, and each $\Gamma_q = \mathbb{R}$ (or$\mathbb{R}^n$), then the CGHDS is the usual *hybrid system*, which corresponds directly to the hybrid automata of [26].

A hybrid system is referred to as a *switched system* if the following conditions are met:

- the dynamics induced by $f_q : \mathcal{X}_q \times \mathcal{U}_q \to \mathcal{X}_q$ are Lipschitz continuous in $\mathcal{X}_q$,
- only a finite number of switches occur within a finite time period [8].

For a real system, infinite switching could only result from a deadlocked control system.

Tools have been developed to assist in the analysis of hybrid systems [14, 26, 48]. From a computer science perspective, the important issues are reachability, decidability, and liveness. Given a hybrid system,

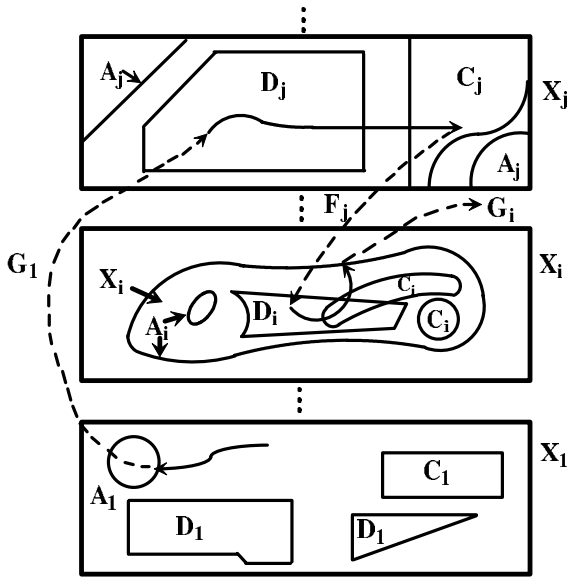


Figure 2: Representation of a controlled general hybrid dynamical system. Based on [8].

with controls and jump maps, is a given state reachable (reachability) or is the set up nodes that satisfy a given condition empty (decidability). For specified jump transition maps (switching policy), liveness means that the system is free of deadlock. Most often, the tools and techniques are based on finite partitions of the state space that yield conservative approximations of the solution.

From a controls perspective, stability is a major concern. In general, the stability analysis of hybrid or switched systems can be difficult [7, 9, 20, 47]. The very act of switching between two dynamical systems can make stable systems unstable, or unstable systems stable. Conceptually, the simplest criteria for stability of a hybrid system is the existence of a common Lyapunov function for all of the switched systems [47]. Unfortunately, the construction of a common Lyapunov function is difficult for general systems. For systems with multiple Lyapunov functions, as long as the value of the Lyapunov function associated with a given control policy decreases relative to the last time the policy was active, the switching is stable [9, 20]. One way to guarantee stability is to carefully construct the control policies and switching strategy. This is the approach taken in this report.

2.2.2 Sequential Composition

Our approach is based on the technique of sequential composition [11]. This technique enables the construction of switched control policies that have guaranteed behavior, and provable convergence properties. In its original form, sequential composition used state regulating feedback control policies whose domains of attraction partitioned the state space into cells. For a given control policy, Φ, the *safe domain of attraction* is defined be the largest positive invariant set that does not intersect any obstacles.Henceforth, we will use the term *domain*, denoted $\mathscr{D}(\Phi)$, to denote the safe domain of attraction for a given policy. The control policy can be thought of as a funnel, as shown in Figure 3, where the vertical represents the value of a Lyapunov function, and the domain is the "shadow" cast by the funnel on the state space below [11].

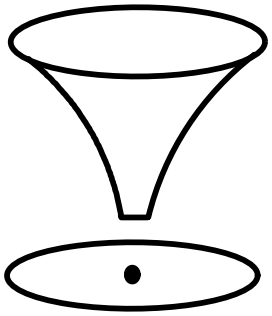


Figure 3: An idealized Lyapunov function and its associated domain of attraction represented by its shadow on the state space. The goal point is shown as a large dot within the domain.

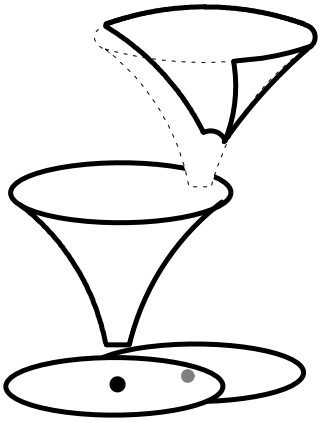


Figure 4: Composition of control policies leads to an enlarged domain of attraction. Note that the active policy switches to the highest priority policy as soon as the state enters its domain.

The basic idea behind sequential composition is to compose multiple control policies in a way that enlarges the domain of attraction, while preserving the underlying convergence guarantees. The goal point of one policy is designed to lie within the domain of another policy. By properly prioritizing the policies, and switching to a higher priority policy once the state enters the domain of the higher priority policy, Burridge *et al.* [11] were able to construct a switching control policy with a domain of attraction equal to the union of the domains of attraction of the component policies (Figure 4). By composing functions with limited domains, constraints in the state space can be respecting while building an approximate globally convergent control policy. In this way, the free state space may be covered by incrementally covering an additional region of the free state space. Figure 5 shows a conceptual example of this process.

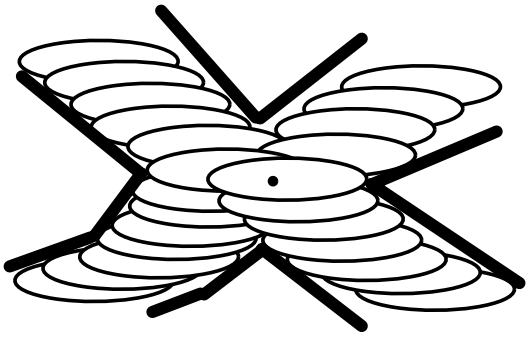


Figure 5: Multiple control policies with overlapping invariant safe domains can be used to build an approximate globally convergent safe control policy that respects constraints in the environment.

The composition of control policies is based on a formal notion of *prepares* [11]. For a given control policy, Φ_1, the domain of the policy is denoted $\mathscr{D}(\Phi_1)$ and the goal is denoted $\mathscr{G}(\Phi_1)$. Given two control policies, Φ_2 is said to *prepare* Φ_1, denoted $\Phi_2 \succeq \Phi_1$, if $\mathscr{G}(\Phi_2) \subset \mathscr{D}(\Phi_1)$. For these two policies, we assume that control switches from Φ_2 to Φ_1 as soon as the state enters $\mathscr{D}(\Phi_1)$. Let $\mathcal{U} = \{\Phi_1, \ldots, \Phi_M\}$ denote a finite collection of parameterized control policies defined over the free state space of a given system. The prepares relationship induces a *digraph* (directed graph), $\Gamma_{\mathcal{U}}$, over the collection of parameterized control policies. In general the graph is cyclic; however, an acyclic graph, $\Gamma'_{\mathcal{U}}$, may be generated over the collection of policies by searching $\Gamma_{\mathcal{U}}$ breadth first, beginning at the node corresponding to the policy that stabilizes the overall goal, and adding only those links and nodes that connect back to previously visited nodes. Thus $\Gamma'_{\mathcal{U}} \subset \Gamma_{\mathcal{U}}$; $\Gamma'_{\mathcal{U}}$ is a partial order over the collection of control policies. By construction, $\Gamma_{\mathcal{U}}$ is a connected graph containing a node corresponding to the policy that stabilizes the goal. Given the collection of policies $\mathcal{U}$and $\Gamma'_{\mathcal{U}}$, the associated partial order, an overall control policy $\Phi = \bigvee \mathcal{U}$ is the induced switching policy defined by the partial order, where $\bigvee$ denotes the operation of control policy composition defined by the prepares relationship. The collection $\mathcal{U}$ of parameterized control policies is known as the *palette* of available control policies. The act of selecting and ordering the policies from the palette is known as *deploying* the control policy. The collection of parameterized control policies and associated partial order is known as a *control policy deployment*; we refer to the composition of the component control policies as the *overall control policy*. For the given deployment, which brings some limited domain to the goal,

$$\mathscr{D}(\mathcal{U}) = \bigcup_{\Phi_i \in \Gamma'_{\mathcal{U}}} \mathscr{D}(\Phi_i) \ .$$

The edge connecting two nodes in $\Gamma'_{\mathcal{U}}$ corresponds to a transition that occurs when the state of the system enters the domain of the higher priority control policy. Under this transition map, $\Gamma'_{\mathcal{U}}$ is a finite state automata. Thus, sequential composition gives a constructive method for building a finite state machine (finite state automata), with predictable transitions guaranteed by the feedback control policies. Reachability and decidability can be determined by the discrete transitions on $\Gamma'_{\mathcal{U}}$, without recourse to analysis of the underlying continuous system.

The overall control policy induced by sequential composition is fundamentally a hybrid control policy. However, the method of composition based on the prepares relationship obviates the need for complex stability analysis of the form given in [7, 9, 20, 47]. The stability of the underlying control policies guarantees the stability of the overall policy because the partial order results in monotonic switching [11]. Disturbances are robustly handled provided their magnitude and rate of occurrence is on a scale small compared to the convergence of the individual policies.

All that is required to determine the switching policy is a partial order over the collection of policies. The resulting switched policy covers the maximum region of the free state space, while guaranteeing that the state is brought to the goal, for a given collection of control policies. Let the cell within the state space where Φ_2 is active be denoted $\mathscr{C}(\Phi_2)$; in this example, $\mathscr{C}(\Phi_2) = \mathscr{D}(\Phi_2) \setminus \mathscr{D}(\Phi_1)$. For an arbitrary collection of policies, the following algorithm recursively builds the required partial order [11]. In fact, the algorithm builds a total order.

Algorithm 1: B-R-K Sequential Composition
Input: Finite collection of parameterized control policies $\mathcal{U} = \{\Phi_1, \ldots, \Phi_M\}$
Output: Ordered collection of parameterized control policies $\mathcal{U}' = \{\Phi_1, \ldots, \Phi_{O(M)}\}$, where $O(M)$ is an index set.

(1) Assume $\mathcal{G}(\Phi_1)$ is the overall goal, and let $\mathcal{V} = \{\Phi_1\}, \mathcal{U}' = \{\}$
(2) **while** $|\mathcal{V}| > 0$
Remove the first element,Φ_j, from $\mathcal{V}$.
(4) Let $\mathcal{C}(\Phi_j) = \mathcal{D}(\Phi_j) \setminus \mathcal{D}(\mathcal{U}')$
(5) **if** $\mathcal{C}(\Phi_j) \neq \emptyset$
(6) Add Φ_j to the end of $\mathcal{U}'$
(7) Append the list of all control policies from $\mathcal{U}$ that prepare Φ_j to the back of $\mathcal{V}$.
(8) Remove the list of policies that prepare Φ_j from $\mathcal{U}$.
(9) **endif**
(10) **endwhile**

The algorithm induces a total order over the collection of available control policies, which determines the policy switching. The system determines the highest priority policy containing the current state, and implements that control policy. Given the collection of parameterized policies, this ordering specifies the control policy deployment. Generating the parameterized control policies such that they fill the free state space is a control policy design problem.

In the work of Burridge *et al.* [11], the task was to control a robotic system that juggled a ping bong ball by repeatedly batting the ball with a paddle. A generic control policy was used to control the ball's setpoint, the horizontal position (x, y) and apex height above the horizontal. The domain of the control policy was limited, and depended on the setpoint and the control policy gains. After developing a generic policy that reliably brought the ball to a stable, periodic trajectory, with a stationary apex, the task then became to move the ball, by juggling, through its domain, while avoiding obstacles. In this case, the obstacles were sensor limits (camera field of view) and a physical obstacle placed in the workspace. The domains of attraction were not available in closed form, so conservative approximations obtained through experiment and simulation were used. The parameterization and deployment of the policies was specified by hand.

Rizzi [59] used sequential composition to simplify programming of robot motions. Rizzi specified overlapping convex polytopes in the configuration space for the case of an idealized dynamical robot, $\ddot{x} = u$ with $x, u \in \mathbb{R}^n$, with both velocity and acceleration constraints in the form of Euclidean norm bounds. Over each polytope, a Save control policy, Φ_S, was designed to bring the maximum subset of the tangent bundle over the polytope to rest within the polytope without violating the polytope boundaries. A velocity regulation policy, Φ_V, used a position dependent velocity reference to bring the system to rest at a specified goal point within the polytope without violating the polytope boundaries. A third policy, called Join and denoted Φ_J, was used to transition between the Save and Velocity Regulator policies. The domains were such that $\mathcal{D}(\Phi_V) \subset \mathcal{D}(\Phi_J) \subset \mathcal{D}(\Phi_S)$. By specifying a goal point within the boundary of an overlapping polytope, the control policies could be composed to move the idealized system through space. The method was extended by Quaid and Rizzi [54] to more complicated bounds on acceleration and velocity. In this latter work, the polytopes represented safe *reservation* regions that the system could travel. Only one agent in this multi-agent framework was allowed access to the reservation area at a time. Parameterizing the policies required specifying a goal point and the polytope vertices, along with a few additional parameters.

3 Overview: Decomposition, Planning, and Control

Our approach to addressing the coupled navigation and controls problem is to define a palette of feedback control policies, and a switching strategy among the individual control policies, such that the resulting composition simultaneously solves both the navigation and control problems. Our approach allows us to leverage the robustness afforded by feedback controls, while generating an approximately globally convergent control policy. We begin by discussing ways to decompose the global task into subtasks that can be controlled by individual control policies. The control policies considered in this research take the form of memoryless state feedback policies. We choose to parameterize our control polices based on local information obtained by decompositions of the free configuration or workspace. Therefore, we begin with a discussion of the relationship between decompositions in work, configuration, and state spaces. We then discuss the deployment problem, and present an extension to sequential composition that allows a larger class of policies to be deployed. The design of the new control policies is deferred until the next section.

3.1 Decomposition of Tasks

The fundamental idea behind the approach outlined in this report is that of *task* decomposition. Global navigation and control tasks are decomposed into a series of sub-tasks, where the solution of each sub-task brings us closer to the completion of the global task. We contrast this with the idea of *problem* decomposition, where we decompose the global problem into a series of problem steps, such as *plan path* and *follow path*. Solving the path planning problem does not cause the system to move closer to the goal.

One approach to automatically generating task decompositions, is to base the tasks on regions of the free space. This is a natural and intuitive decomposition that humans perform every day. For example, to get to class, one must leave their office, go down the hall, leave the building, cross the road, enter another building, go down another hall, and finally enter the class room. By decomposing the tasks into regions of free space (office, hallway), the global navigation and control problem is reduced to a more local problem (go around the desk to get to the door), which is (hopefully) more tractable than the global problem.

3.2 Decomposition of Free Space

The robot to be controlled lives simultaneously in at least three abstract spaces: workspace, configuration space, and state space. The *workspace*, $\mathcal{W}$, of the robot is its surrounding environment; the world that is seen. Generally, either the plane, modeled as $\mathbb{R}^2$, or space, modeled as $\mathbb{R}^3$, is considered as the workspace. A *configuration* of the robot is the set of variables required to specify the location of each point of the robot in its workspace relative to some fixed frame of reference [42, 50]. The *configuration space*, $\mathcal{Q}$, is the set of all configurations of the robot. The *state* of the system is set of variables required to uniquely specify the system, such that given a known initial state and the system inputs over time, it is possible to solve for the state of the system at all future times [2]. For dynamical systems, the state is both configuration and velocity. The *state space*, $\mathcal{X}$, of a dynamical system is the set of all possible values of the state

$$\mathcal{X} = \mathcal{TQ} = \{(q, \dot{q}) \mid q \in \mathcal{Q}, \dot{q} \in \mathcal{T}_q\mathcal{Q}\} ,$$

where $\mathcal{TQ}$ is the tangent bundle of the configuration space, and $\mathcal{T}_q\mathcal{Q}$ is the tangent space at configuration q. For kinematic systems, the state space is just the configuration space, that is $\mathcal{X} = \mathcal{Q}$. This report assumes that each of these three spaces ($\mathcal{W}$,$\mathcal{Q}$,$\mathcal{X}$) is bounded.

The research presented in this report considers the problem of navigation in complex or cluttered spaces, where the workspace is populated with a finite number of obstacles, $\mathcal{O}_i$, that occupy certain points in the workspace. The boundary of the space is considered to be obstacle zero, that is $\mathcal{O}_0$. These obstacles, $\mathcal{O}_i$, can

be mapped directly to the configuration space ($\mathcal{C}-$space) by

$$\mathcal{QO}_i = \{q \in \mathcal{Q} \mid \mathcal{R}(q) \cap \mathcal{O}_i \neq \emptyset\} ,$$

where $\mathcal{R}(q)$ is the set of points in the workspace occupied by the robot at configuration q [42, 23]. Analogous to the $\mathcal{C}-$space obstacles, we could define a state space obstacle

$$\{(q, \dot{q}) \in \mathcal{X} \mid \mathcal{R}(q) \cap \mathcal{O}_i \neq \emptyset\} .$$

However, this does not capture the full complexity of state space. There are regions of state space where the corresponding configuration is not contained in an obstacle, but whose velocity makes collision with an obstacle unavoidable for systems with bounded acceleration. Therefore, we define a state space obstacle as

$$\mathcal{XO}_i = \{(q, \dot{q}) \in \mathcal{X} \mid \mathcal{R}(q) \cap \mathcal{O}_i \neq \emptyset \text{ or } (\nexists \Phi(q, \dot{q}) \text{ s.t. } \forall t\, \mathcal{R}(q(t)) \cap \mathcal{O}_i = \emptyset)\} ,$$

where Φ is some arbitrary control policy that satisfies the given constraints on the system.

The *free space* is the set of points within a given space that is not occupied by obstacles. More formally, for k obstacles,

$$\begin{aligned} \mathcal{FS}_{\mathcal{W}} &= \mathcal{W} \setminus \bigcup_{i=0}^{k} \mathcal{O}_i , \\ \mathcal{FS}_{\mathcal{Q}} &= \mathcal{Q} \setminus \bigcup_{i=0}^{k} \mathcal{QO}_i , \\ \mathcal{FS}_{\mathcal{X}} &= \mathcal{X} \setminus \bigcup_{i=0}^{k} \mathcal{XO}_i , \end{aligned}$$

This report will use the term *free space*, denoted $\mathcal{FS}$, to apply generically to the free work, free configuration, or free state spaces. Where the distinction is important, we will use the subscripts as above.

Consider a decomposition of the free work or configuration space into cells, with the collection of cells known as a *cellular decomposition* [42]. The standard definition of a cellular decomposition only requires that the union of the cells cover the free space. In this work, we present a more restrictive definition:

Definition: A *disjoint cellular decomposition*, $\mathcal{K}$, is a finite collection of cells in the free (work or configuration) space,

$$\mathcal{K} = \{\mathcal{P}_i \subset \mathcal{FS} \mid i = 1 \ldots n, \mathcal{P}_i \cap \mathcal{P}_j = \emptyset \; \forall i \neq j\} ,$$

where each cell $\mathcal{P}_i$ is defined as a simply connected compact open set. The union of the closures of the cells, $\bar{\mathcal{P}}_i$, in the collection covers the free space, either exactly ($\mathcal{FS} = \bigcup_i \bar{\mathcal{P}}_i$) or to an approximation at some resolution($\mathcal{FS} \approx \bigcup_i \bar{\mathcal{P}}_i$).

For general environments and cells, the problem of constructing any cellular decomposition is known to be NP-hard; for now, we will assume that a disjoint cellular decomposition exists, and is given to us [42]. Figure 6 shows a simple disjoint cellular decomposition of a non-simply connected region in $\mathbb{R}^2$.

The requirement that the union of the closures covers the free space implies that the closures are connected to neighboring cells by a common boundary region. We restrict the decomposition such that the common boundary region is a simply connected set of co-dimension one, and give the following definition:

Definition: Two cells $\mathcal{P}_i$ and $\mathcal{P}_j$ in $\mathcal{K}$ are *adjacent* if and only if they share a common simply connected boundary set of co-dimension one, *i.e.* $\dim\left(\bar{\mathcal{P}}_i \cap \bar{\mathcal{P}}_j\right) = n - 1$, where n is the dimension of the free space.

If the common boundary formed by the intersection of the closures is a disconnected set, then the cells should be split into multiple cells.

The connectivity of the cells is easily encoded into an *adjacency graph*, also called a *connectivity graph* [42].

Definition: The *adjacency graph* $\mathcal{G}(V, E)$ associated with the cellular decomposition $\mathcal{K}$ is a simple graph with vertices V and edges E such that

- The set of vertices, V, has a one-to-one correspondence to cells in $\mathcal{K}$.
- Two vertices $i, j \in V$ are connected by edge, $e \in E$, if and only if the corresponding cells $\mathcal{P}_i, \mathcal{P}_j \in \mathcal{K}$ are *adjacent*.

Figure 7 shows the adjacency graph associated with the decomposition of Figure 6.

3.3 Control Policy Design

The feedback control policies considered in this report are designed to have a limited domain with respect to the free state space. The policies are parameterized based on closed cells within the free configuration space, denoted $\bar{\mathcal{P}}$. With each configuration cell, $\mathcal{P}$, we associate one or more an individual control policies, which we term *component control policies* for $\mathcal{P}$, denoted $\Phi_{\mathcal{P}}$.

The extension of the configuration cell into the state space creates a generalized cylinder, as shown in Figure 8. The generalized cylinder is capped by the velocity constraints of the system. This generalized cylinder is the *tangent bundle* over the configuration cell, and is denoted $\mathcal{TP}$, where $\mathcal{TP} \subset \mathcal{X}$. For each component control policy, $\Phi_{\mathcal{P}}$, defined over the cell $\mathcal{P}$, there is an associated goal set, $\mathcal{G}(\Phi_{\mathcal{P}}) \subset \mathcal{TP}$. The goal set may be an attractive set, as in [59], or it may define a subset of the boundary of the generalized cylinder, as do the control policies developed in this report.

For dynamical systems with acceleration bounds, there are states in the generalized state space cylinder for which there does not exist a control policy that will drive the state to the goal set without leaving the free state space over the cell. Consider an initial condition adjacent to a cell boundary, with initial velocity oriented toward the boundary. For a system with bounded acceleration, collision with the boundary cannot be prevented. For this reason, the generalized cylinder is further cut by a surface defining the velocity constraints as a function of configuration and available control policies, as shown in Figure 9.

Rizzi [59] developed a switched control policy, $\Phi_{\mathcal{P}}$ defined over a given cell $\mathcal{P}$, such that the domain of the switched policies was

$$\mathcal{D}(\Phi_{\mathcal{P}}) = \{x = (q, \dot{q}) \in \mathcal{TP} \mid \forall t, x(t) \in \mathcal{TP}, \text{ and } \nexists t \text{ s.t. } (\|\ddot{q}(t)\| > \mathrm{A_{max}} \text{ or } \|\dot{q}(t)\| > \mathrm{V_{max}})\} .$$

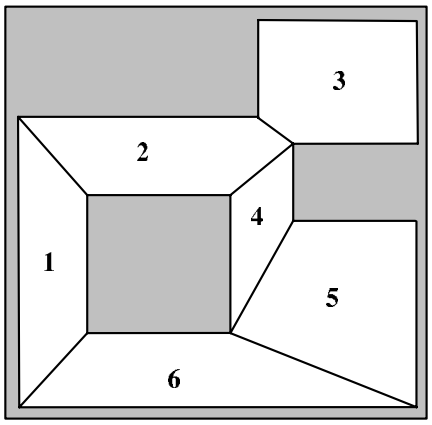


Figure 6: In this example, the free space (white) is decomposed into a set of labeled polygonal regions .

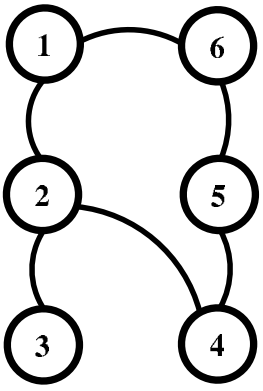


Figure 7: The adjacency graph corresponding to the cellular decomposition shown in Figure 6.

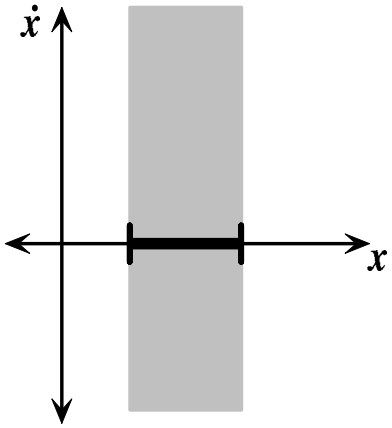


Figure 8: A simple 1D cell in configuration, and its extension as a generalized cylinder into the dynamical system's state space.

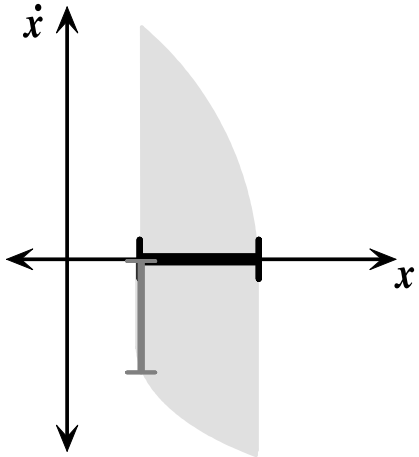


Figure 9: A simple 1D cell in configuration space, and its limited domain in state space given bounded accelerations. A goal set on the boundary of the tangent bundle is shown which corresponds to exitting the cell on the left boundary.

The policies were designed to be attractive to a given point $x_{\mathcal{P}} \in \mathcal{P}$, so that $\mathcal{G}(\Phi_{\mathcal{P}}) = \{x_{\mathcal{P}}, \mathbf{0}\} \in \mathcal{TP}$. The switched control policy Φ was designed to be positive invariant over the given configuration cell $\mathcal{P}$ for some subset of $\mathcal{TP}$. This subset was termed the *savable set*.

The newest control policies developed in this report define a goal set on the boundary of $\mathcal{TP}$. The policies are designed cause the system configuration to exit the configuration cell within a specified region of the cell boundary, which is termed the *outlet zone*. The boundary of a cell, excluding the outlet zone, is termed the *inlet zone*. It is assumed that the active control policy changes as the configuration crosses the outlet zone of a given cell. If the goal configuration is contained in a configuration cell, then the outlet zone is the empty set.

The domain of these control policies, $\mathcal{D}(\Phi_{\mathcal{P}}) \subset \mathcal{TP}$, is defined as the set of states in the generalized cylinder such that $\Phi_{\mathcal{P}}$ drives the state into the goal set, $\mathcal{G}(\Phi_{\mathcal{P}}) \subset \mathcal{D}(\Phi_{\mathcal{P}})$ without leaving the free state space contained in the generalized cylinder, or violating the dynamic constraints of the system. Note, that this definition excludes states that may not violate the constraints immediately, but unavoidably lead to a violation during the evolution under the influence of a given control policy. That is

$$\begin{aligned} \mathcal{D}(\Phi_{\mathcal{P}}) \quad = \{ x = (q, \dot{q}) \in \mathcal{TP} \mid \quad & \exists t^* \text{ s.t. } x(t^*) \in \mathcal{G}(\Phi_{\bar{\mathcal{P}}}) \subset \mathcal{TP} \text{ and} \\ & \forall t < t^*, x(t) \in \mathcal{TP} \text{ and} \\ & \nexists t < t^* \text{ s.t. } (\|\ddot{q}(t)\| > \mathrm{A}_{\max} \text{ or } \|\dot{q}(t)\| > \mathrm{V}_{\max}) \} \ . \end{aligned} \tag{2}$$

(Note, for kinematic systems, the definition drops the acceleration constraint.) The domain $\mathcal{D}(\Phi_{\mathcal{P}})$ is designed to be *conditionally positive invariant*, meaning that the trajectory under the influence of the control policy exits the domain only via the designated goal set $\mathcal{G}(\Phi_{\mathcal{P}})$ [28].

Returning to the task decomposition example, consider the example of navigating out of a given room. The cell in workspace is the room, and the doorway is the specified outlet zone. The control policy is designed to cause the maximal set of initial states contained in the generalized state space cylinder to exit the room via a specified doorway. The goal set encompasses those velocities that ensure the system exits the doorway (*i.e.* , zero velocity at the exit is not in the goal set).

In Section 4, this new type of control policy is developed for a particular class of configuration cells, and is designed to give a closed form solution based on local information. The control policy, based on a solution to Laplace's steady state heat equation, is intended to be but a single example of a valid control policy. Additional policies meeting the criteria above could provide an alternative solution, which could result in different paths through a specified configuraton cell, and add to the flexibility of the deployment scheme presented next. Although not explored in this report, other methods including other potential field techniques or dynamic programming, could be used to design control policies of this type [1, 42].

3.4 Control Policy Deployment

This research uses the idea of sequential composition to generate a switching control policy whose domain of attraction is greater than any single control policy. The prior work using sequential composition [11, 54, 59], discussed in Section 2.2.2, was based on point attractors. In contrast, the policies developed by this research result in state that flows through a goal set, not state that is attracted to a goal point. We will refer to policies of this latter type as *set attractors*, realizing that "attraction" is only with respect to a given cell. The original deployment scheme developed in [11] requires that a single policy prepare another policy. This does not work for set attractors if the goal set of one policy intersects the domains of multiple policies, but is not contained within the domain of any single policy. In this section, we present an extension to the original deployment scheme to allow control policies with set attractors whose goal sets intersect the multiple policies, as well as point attractors.

The prepares test is now based on the union of higher priority control policies. If the goal set of a component control policy is covered by the union of the domains of higher priority control policies, the component control policy may be added to the deployment. This extended definition of prepares allows us to deploy policies whose domains encompass more of the free state space over the configuration cell. Sequential composition guarantees that the deployment is free of deadlock, as the system will always enter the domain of a higher priority control policy. Because of the definition of prepares depends on the order of the deployed policies, the digraph that specifies the prepares relationship must be constructed concurrent with the order. Beginning at the cell containing the goal configuration, $\mathcal{P}_g$, a convergent control policy Φ_g that drives the state to the goal is deployed and added to the root node of the digraph $\Gamma'_{\mathcal{U}}$. Any control policies that prepare the goal policy are added to the collection of deployed control policies and the prepares digraph. We now present two algorithms for specifying the deployment and prepares graph. The first, based on [11], is a greedy algorithm that deploys control policies as encountered.

Algorithm 2: Extended Sequential Composition
Input: Finite collection of parameterized control policies $\mathcal{U} = \{\Phi_1, \ldots, \Phi_M\}$
Output: Ordered collection of parameterized control policies $\mathcal{U}' = \{\Phi_1, \ldots, \Phi_{O(M)}\}$, where $O(M)$ is an index set.

(1) Remove Φ_1 from $\mathcal{U}$, and let $\mathcal{U}' = \{\Phi_1\}$
(2) Define $\Gamma'_{\mathcal{U}}$ as a digraph with one node corresponding to Φ_1
(3) $Flag = 1$
(4) **while** $|\mathcal{U}| > 0$ **and** $\mathcal{FS}_{\mathcal{X}} \backslash \mathcal{D}(\mathcal{U}') \neq \emptyset$ **and** $Flag = 1$
$Flag = 0$
(6) **foreach** $\Phi_i \in \mathcal{U}$
(7) **if** $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\mathcal{U}')$ [**and** $\mathcal{D}(\Phi_i) \backslash \mathcal{D}(\mathcal{U}') \neq \emptyset$]
(8) Remove Φ_i from $\mathcal{U}$.
(9) Add Φ_i to the end of $\mathcal{U}'$ ($\mathcal{U}' = \mathcal{U}' \bigcup \{\Phi_i\}$)
(10) Add a node to $\Gamma'_{\mathcal{U}}$ for Φ_i, and an edge from the Φ_i node to each node j in $\Gamma'_{\mathcal{U}}$ such that $\mathcal{G}(\Phi_i) \cap \mathcal{D}(\Phi_j) \neq \emptyset$
(11) $Flag = 1$
(12) **endif**
(13) **endfor**
(14) **endwhile**

The algorithm deploys the first available policy, continuing until either the free state space is filled or the remaining policies do not add to the overall domain. By adding the optional condition, $\mathcal{D}(\Phi_i) \backslash \mathcal{D}(\mathcal{U}') \neq \emptyset$, to line 7, the algorithm may avoid redundancy by only adding policies that increase the overall domain.

In Algorithm 2, as in the original algorithm in [11], the order of deployment is dependent on the ordering in the original collection $\mathcal{U}$. This dependency can be avoided with minor modifications to Algorithm 2, which are presented in Algorithm 3.

Algorithm 3: Extended Sequential Composition with Reordering.
Input: Finite collection of parameterized control policies $\mathcal{U} = \{\Phi_1, \ldots, \Phi_M\}$
Output: Ordered collection of parameterized control policies $\mathcal{U}' = \{\Phi_1, \ldots, \Phi_{O(M)}\}$

(1) Remove Φ_1 from $\mathcal{U}$, and let $\mathcal{U}' = \{\Phi_1\}$
(2) Define $\Gamma'_{\mathcal{U}}$ as a digraph with one node corresponding to Φ_1
(3) $Flag = 1$
(4) **while** $|\mathcal{U}| > 0$ **and** $\mathcal{FS}_{\mathcal{X}} \backslash \mathcal{D}(\mathcal{U}') \neq \emptyset$ **and** $Flag = 1$
$Flag = 0$
(6) $\mathcal{V} = \{\}$
(7) **foreach** $\Phi_i \in \mathcal{U}$
(8) **if** $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\mathcal{U}')$ [**and** $\mathcal{D}(\Phi_i) \backslash \mathcal{D}(\mathcal{U}') \neq \emptyset$]
(9) Remove Φ_i from $\mathcal{U}$.
(10) $\mathcal{V} = \mathcal{V} \bigcup \{\Phi_i\}$
(11) Add a node to $\Gamma'_{\mathcal{U}}$ for Φ_i, and an edge from the Φ_i node to each node j in $\Gamma'_{\mathcal{U}}$ such that $\mathcal{G}(\Phi_i) \cap \mathcal{D}(\Phi_j) \neq \emptyset$
(12) $Flag = 1$
(13) **endif**
(14) Order $\mathcal{V}$ based on some metric
(15) Add ordered $\mathcal{V}$ to the end of $\mathcal{U}'$
(16) **endfor**
(17) **endwhile**

The algorithm is constructed to allow for reordering the control policies at each priority layer. Policies that prepare previously deployed policies are placed in a temporary queue $\mathcal{V}$ and removed from $\mathcal{U}$. Once the remaining policies in $\mathcal{U}$ are processed, the policies in $\mathcal{V}$ are reordered, and then added to the deployed policies in $\mathcal{U}'$. Although not developed in this report, the metric is expected to use a cummulative average cost of traversing a given control policy domain, along with the cost of traversing each higher priority policy in the order. In this way, the final ordering is not dependent on the original ordering of $\mathcal{U}$.

Although the definition is the same, the notion of prepares is different for kinematic and dynamical systems. For kinematic systems, the test is based on set membership in the configuration space. For dynamical systems, the test is based on set membership in the full state space – both configuration and velocity. This generally results in more complicated tests for membership for dynamical systems. Because a given control policy results in a capped generalized cylinder in state space, it is anticipated that multiple policies over a given cell, and multiple overlapping cells, must be considered to completely cover the free state space for dynamical systems.

3.5 Hybrid Systems Analysis

The resulting composition of the component control policies forms a hybrid control policy [8, 10, 26]. The hybrid control policy, defined as a policy containing both discrete logic and continuous control laws, causes the dynamics of our closed loop system to become hybrid as well. Since it fits well with the notion of the digraph specifying the partial order, this report uses the description of the hybrid automata model of [26]. The nodes of the hybrid automata correspond to the nodes in the digraph $\Gamma'_{\mathcal{U}}$, with edges that correspond to the specified prepares relationship. The events are triggered once the continuous state enters the domain of the next highest priority control policy in the partial order. Because the domains of the component control policies are (conditionally) positive invariant, the state will not exit the domain of an active policy until the state is in the domain of a higher priority control policy.

The continuous dynamics of the system under the influence of the component feedback control policies are governed by

$$\dot{x}(t) = f_i(x(t)) ,$$

where $x \in \mathbb{R}^{\dim(\mathcal{X})}$ and f_i represents the dynamics under the influence of the i^{th} control policy. With the restriction that the function $f_i(x(t))$ is Lipschitz continuous, the dynamics of the system under the influence of the ordered control policies are those of a *switched system* [8]. Another requirement for a hybrid system to be classified as a switched system is that finite switches occur in finite time. This requirement is satisfied given a specification that the domain of each policy is a non-zero measure subset of the free state space. Since the domain of each policy is a full dimensional subset of the free state space, and the state space is compact, there are only a finite number of policies required to cover the free state space [39]. Because the control policies are deployed based on a partial order, the switching is monotonic, there can be at most a finite number of switches.

In general, the stability analysis of hybrid or switched systems can be difficult [7, 9, 20, 47]. Fortunately, for hybrid systems constructed using the sequential composition technique and the prepares relationship, the stability analysis is straightforward.

Lemma 3.1 *Given a collection of local asymptotically stable control policies $\mathcal{U}' = \{\Phi_i\}$ ordered as described in Section 3.4, the overall hybrid control policy $\Phi = \bigvee \mathcal{U}'$ is asymptotically stable provided that the switches between policies occur in finite time. Furthermore, the domain of the overall control policy, $\mathscr{D}(\Phi)$, is given by the union of the domains of the local control policies, that is*

$$\mathscr{D}(\Phi) = \bigcup_i \mathscr{D}(\Phi_i) .$$

Proof: By construction, each component control policy, Φ_i, is (conditionally) positive invariant over its domain $\mathscr{D}(\Phi_i)$, and brings any state in its domain to its goal set $\mathscr{G}(\Phi_i)$. This goal set, $\mathscr{G}(\Phi_i)$, is either the global goal, or the goal set lies within the domain of the higher priority control policies,

$$\mathscr{G}(\Phi_i) \subset \bigcup_{j=1\cdots i} \mathscr{D}(\Phi_j) \ .$$

We assume the existence of a properly constructed partial order, as described in Section 3.4. The partial order is represented as a acyclic digraph, denoted $\Gamma'_{\mathcal{U}}$. The root node Φ_1 of $\Gamma'_{\mathcal{U}}$ corresponds to a control policy, Φ_g, that asymptotically brings any state in $\mathscr{D}(\Phi_g)$ to the goal. Upstream nodes in the digraph correspond to component control policies, and are labeled Φ_i with $i \in 2, \ldots, N$, where $N = |\Gamma'_{\mathcal{U}}|$ is the number of nodes including the root.

First, consider the case where the digraph has only one node, then $\mathscr{D}(\Phi) = \mathscr{D}(\Phi_g)$, and Φ_g is asymptotically stable by construction. Therefore, Lemma 3.1 is trivially true.

Assume the digraph has more than one node. Let Φ_i be any node with a single outlet edge, which connects to the root node Φ_1 in the digraph Γ. For any state in $\mathscr{D}(\Phi_i)$, the control policy Φ_i drives the state to $\mathscr{G}(\Phi_i)$. A switch from Φ_i to Φ_1 occurs when the state enters the domain of Φ_1 as the state approaches the $\mathscr{G}(\Phi_i)$, whereby the state begins to approach $\mathscr{G}(\Phi_1)$ under the influence of Φ_1. The switch is guaranteed to occur since $\mathscr{G}(\Phi_i)$ is contained in $\mathscr{D}(\Phi_1)$ and the state is attracted to the goal set $\mathscr{G}(\Phi_i)$ under the influence of Φ_i. Furthermore, we restrict the deployment such that this switch is guaranteed to occur in finite time. This switch corresponds to a transition from the upstream node Φ_i to the root node Φ_1 in the partial order. By construction, $\mathscr{D}(\Phi_i)$ is conditionally positive invariant under the influence of Φ_i, such that the state only leaves $\mathscr{D}(\Phi_i)$ when $x \in \mathscr{G}(\Phi_i) \subset \mathscr{D}(\Phi_1)$. Therefore, any state starting in $\mathscr{D}(\Phi_i) \bigcup \mathscr{D}(\Phi_1)$ is guaranteed to approach $\mathscr{G}(\Phi_1)$ under the assumption that $\Phi_i \succeq \Phi_1$. Thus $\mathscr{D}(\Phi_i) \bigcup \mathscr{D}(\Phi_1)$ is conditionally positive invariant under the influence of the switched control policy $\{\Phi_1, \Phi_i\}$, with goal set $\mathscr{G}(\Phi_1)$. Since the state enters $\mathscr{D}(\Phi_1)$ from $\mathscr{D}(\Phi_i)$ in finite time, the switched control policy is at worst asymptotic. Once the state enters $\mathscr{D}(\Phi_1)$, it asymptotically approaches $\mathscr{G}(\Phi_1)$. Given these facts, the proof of Lemma 3.1 is constructive:

Algorithm 4: Proof of Composition

Input: Acyclic digraph $\Gamma'_{\mathcal{U}}$ defining the partial order of control policies with associated domains and goal sets

Output: Single switched control policy Φ and associated domain and goal sets

(1) **while** $|\ \Gamma'_{\mathcal{U}}\ | > 1$

Select any node Φ_i in $\Gamma'_{\mathcal{U}}$ connected to node Φ_1, with the restriction that Φ_i has only one outlet edge

(3) Define $\Phi'_1 = \{\Phi_1, \Phi_i\}$, where Φ'_j is the switched control policy with domain $\mathscr{D}(\Phi'_j) = \mathscr{D}(\Phi_i) \bigcup \mathscr{D}(\Phi_1)$, and goal $\mathscr{G}(\Phi'_1) = \mathscr{G}(\Phi_1)$

(4) Replace Φ_1 with Φ'_1, and let $\mathscr{D}(\Phi_1) = \mathscr{D}(\Phi'_1)$

(5) Connect edges in $\Gamma'_{\mathcal{U}}$ entering node Φ_i to Φ_1

(6) Remove node Φ_i from $\Gamma'_{\mathcal{U}}$

(7) **endwhile**

After the final step, we have a single switched control policy $\Phi = \bigvee \mathcal{U}$, whose domain $\mathscr{D}(\Phi) = \bigcup_i \mathscr{D}(\Phi_i)$ and goal $\mathscr{G}(\Phi) = \mathscr{G}(\Phi_g)$. Thus, Lemma 3.1 is true by construction.

The restriction that Φ_i have only one departing edge before it is merged with Φ_1 ensures that there are no policies with an intermediate priority between Φ_i and Φ_1. Because Φ_1 is the highest priority, all policies of the next priority can only have one departing edge.

We conclude that the deployment of control policies based on the prepares relationship results in an asymptotically stable overall switched control policy. Both the stability and domain are inherited from the component control policies, obviating the need for detailed analysis of the stability of the hybrid switched system as in [9] or [47].

□

The restriction that switches from lower priority control policies occur in finite time is satisfied by the policy design methods given in this report. Likewise, the methods of [59] and [54] satisfy the finite switching time provided the intermediate goal points are placed in the interior of the overlapping cells.

4 Development of Component Control Policies

The control policy deployment outlined in Section 3.4 depends on developing asymptotically stable component control policies over (conditionally) positive invariant domains. In this section, we present a candidate component control policy for an arbitrary cell in a cellular decomposition based on a potential field free of local minima. Control Policies for both kinematic and dynamical systems are developed. The control policies are designed to bring a large portion of the free state space over the cell to a specified goal set on the boundary of the generalized cylinder.

Our approach to generating a potential field over an arbitrary cell is to map the cell to a unit n-ball centered at the origin,

$$\mathcal{B}(0,\rho) = \{x \in \mathbb{R}^n \mid ||x|| \leq \rho\} ,$$

which is bounded by the unit $(n-1)$-sphere. A potential function, $\gamma_b : \mathcal{B} \rightarrow \mathbb{R}$, is calculated, where γ_b is free of local minima. The potential function is designed to have its global minima along a region of the boundary, and its global maximum along the remaining boundary. The potential function in the arbitrary cell, $\mathcal{P}$, is given as the *pull back* of the potential in the n-ball, *i.e.*

$$\gamma = \gamma_b \circ \varphi ,$$

where $\varphi : \mathcal{P} \rightarrow \mathcal{B}$. Our control policies are designed to cause the system to flow along the integral curves of the negative gradient field. We begin with a description of our chosen model space, present our solution to the potential field, and then develop proofs of the validity of our pull back approach. We develop a general velocity reference control policy, and use it as the basis for a collection of hybrid control policies over each configuration cell that are safe, reliable, and robust.

4.1 The Unit Ball - $\mathcal{B}(0,1)$

By construction, each configuration cell in the cellular decomposition of the free space is a simply connected compact set. The boundary of such cells is likewise a simply connected compact $(n-1)$-surface, where n is the dimension of the free space. We assume the existence of a homeomorphism $\varphi : \mathcal{P} \rightarrow \mathcal{B}$ for each cell in the cellular decomposition. As we will describe later, we also require φ to be full rank, *i.e.* its Jacobian matrix is full rank [5]. Our control policy construction will require the derivatives of order less than or equal to some $k > 0$ to be continuous, therefore we require that φ be a C^k mapping from $\mathcal{P}$ to $\mathcal{B}$. We are only concerned with derivatives of the forward mapping, thus a C^k*-diffeomorphism* is not required..

Our work will focus on the use of convex sets to define the configuration cells in the decomposition, although the work readily extends to star shaped sets. The overall technique that we describe below is in fact quite general. From the generalizations to the Poincaré conjecture, and the (apparent) proof for 3-surfaces, we know simply connected compact surfaces are homeomorphic to the $(n-1)$-sphere, $\mathbf{S}^{n-1}$ [51]. The cell boundary surface is an orientable surface, which lends itself to a notion of inside and outside. Constructing a mapping for an arbitrary non-convex set is often difficult; but given such a mapping, the techniques described in this section readily extend to more complex set definitions.

Using this idea, we will define a *n-cell* to be a simply connected compact set in configuration space whose boundary,$\partial\mathcal{P}$, is homeomorphic to the $(n-1)$-sphere, and whose interior can be C^k continuously mapped to the open ball centered at the origin with unit radius, denoted $\mathcal{B}(0,1)$. In addition to the requirements given in Section 3.2, we require that each cell in our cellular decomposition be an n-cell.

For cells whose boundary is not a C^k continuous manifold, the required derivatives may not exist on the boundary of the cell. In this case, we approximate the cell by a region $\mathcal{P}^* \subset \mathcal{P}$ that differs from the cell only in regions of discontinuity and matches the boundary elsewhere, such that

$$\varphi \mid_{\mathcal{P}^*} \in C^k .$$

Our control policy will provide a continuous transition from the region $\mathcal{P} \backslash \mathcal{P}^*$.

Because our mapping is continuous and full rank, connected regions in the cell $\mathcal{P}$ remain connected in the ball $\mathcal{B}$. This causes the outlet zone of the cell to be mapped to a simply connected compact region of the $(n-1)$-sphere, which we also call the outlet zone and denote $\partial\mathcal{B}_{\text{out}}$. The remainder of the cell boundary maps continuously to the remainder of sphere, and we refer to this portion of the sphere as the inlet zone, denoted

$$\partial\mathcal{B}_{\text{in}} = \partial\mathcal{B} \backslash \partial\mathcal{B}_{\text{out}} .$$

The potential along the outlet zone is defined as

$$\gamma_b \left(\partial\mathcal{B}_{\text{out}}\right) = 0 ,$$

while the potential along the inlet zone is defined as

$$\gamma_b \left(\partial\mathcal{B}_{\text{in}}\right) = 1 .$$

The choice of one for the inlet boundary is arbitrary and could be scaled by any positive factor without changing the fundamental results. The choice of zero for the outlet boundary potential is for convenience; any constant value may be added without affecting the gradients. Using function composition, the potential values along surface of the cell is given by

$$\gamma = \gamma_b \circ \varphi ,$$

where the cell has a unit potential along the inlet zone and zero potential along the outlet zone. To generate the potential in the interior of the cell, we first find the potential in the unit ball using the solution to Laplace's equation as described in the next section.

4.2 Harmonic Potential Functions

Laplace's equation, the partial differential equation (PDE) given by

$$\nabla^2 u = \frac{\partial^2 u}{\partial {x_1}^2} + \cdots + \frac{\partial^2 u}{\partial {x_n}^2} = 0 , \tag{3}$$

is widely used in modeling physical phenomena such as heat conduction or chemical diffusion [24, 60]. In (3), using the standard notation of PDE literature, $u : \bar{U} \to \mathbb{R}$ where $U \subset \mathbb{R}^n$ is a given open set, $x \in U$, and $u = u(x)$. In modeling heat conduction, u is the temperature, and the heat flux $\mathbf{F}$ is proportional to the temperature gradient, $F = -aD_x u$, where $a > 0$ is some constant. For steady state heat conduction, the net flux through any smooth subregion $V \subset U$ is zero, or formally

$$\int_{\partial V} \mathbf{F} \cdot \nu \, dS = 0 ,$$

where ν is the outward pointing normal vector along the boundary ∂V. By the Gauss-Green theorem, we may write

$$\int_{\partial V} \mathbf{F} \cdot \nu \, dS = \int_V \mathbf{div}\, \mathbf{F} \, dx = 0 ,$$

where $\mathbf{div}$ is the divergence operator $\mathbf{div} = \nabla \cdot = \left[\frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_n}\right]$.

Because V is arbitrary, this implies $\mathbf{div}\ \mathbf{F} = 0$. Because $a > 0$, this implies that $\nabla^2 u = 0$, which is Laplace's equation. If $u \in C^2(U)$ and satisfies (3), then u is called a *harmonic* function [24]. Harmonic functions have the following "nice" properties [24, 60]:

Mean Value Formula: If $u \in C^2(U)$ is harmonic, then

$$u(x) = \frac{1}{\omega_n} \int_{\partial \mathcal{B}(x,\rho)} u \, dS = \frac{1}{\Omega_n} \int_{\mathcal{B}(x,\rho)} u \, dx ,$$

where ω_n is the measure of the $(n-1)$-sphere, and Ω_n is the measure of the $(n-1)$-ball,

Maximum (Minimum) Principle: Suppose $u \in C^2(U) \cap C(\bar{U})$ is harmonic within U, then

$$\max_{\bar{U}} u = \max_{\partial U} u ,$$

and

$$\min_{\bar{U}} u = \min_{\partial U} u ,$$

Uniqueness: Given a boundary solution $u = g$ on ∂U, there exists at most one solution $u \in C^2(U) \cap C(\bar{U})$,

Smoothness: Assume u is harmonic in U, then $u \in C^\infty(U)$ and u is analytic in U.

As a consequence of these properties, the potential function u is free of local minima.

For our system, the open set U is given by the unit ball $\mathcal{B}$, while u is given by our potential function γ_b. For now, assume that γ_b is a harmonic function that uniquely solves the boundary value problem given $\gamma_b(\partial \mathcal{B})$. We can then find the potential on the $(n-1)$-cell $\mathcal{P}$ as $\gamma = \gamma_b \circ \varphi$.

4.2.1 Pull Back of Harmonic Potential Functions

The idea of mapping the solution in one space to a valid solution in another space is closely related to the idea of conformal mapping, which is predicated on preserving the properties of harmonic equations between spaces. In classical conformal mapping using the Schwarz-Christoffel map, planar shapes are mapping to regions of the complex plane that have well defined solutions to PDE [22]. Because conformal maps are defined as those that preserve angles, the harmonic properties are preserved during the pull back, resulting in a valid solution for the original planar shape. The homeomorphism φ between the arbitrary $(n-1)$-cell and the unit ball $\mathcal{B}$ is not necessarily an angle preserving mapping. Therefore, we must address the validity of the pull back in detail; including the preservation of the "nice" properties.

In order for γ to have a local minima, $D_q\gamma$ must disappear, where

$$D_q\gamma = D_{q_b}\gamma_b \, D_q\varphi , \tag{4}$$

$q \in \mathcal{P}$, and $q_b \in \mathcal{B}$ [68]. Because γ_b does not have a local minima in $\mathcal{B}$, $D_{q_b}\gamma_b$ is nowhere zero in $\mathcal{B}$. Therefore, $D_q\varphi$ must be rank deficient for γ to have a local minima. We require that φ is full rank over $\mathcal{P}$, therefore γ is free of local minima. Because φ maps the boundary of $\mathcal{P}$ to $\partial \mathcal{B}$, and γ is free of local minima, the maximum (minimum) principle is preserved by the pull back. Because the composition of continuous functions is continuous, $\varphi \in C^k(\mathcal{P}, \mathcal{B})$, and $\gamma_b \in C^\infty(\mathcal{B})$, the pull back γ is an element of $C^k(\mathcal{P})$. With the restriction that φ is full rank, $\gamma : \mathcal{P} \to \mathbb{R}$ is a C^k continuous potential function without local minima.

4.2.2 General Form: Unit Ball in $\mathbb{R}^n$

The solution to Laplace's equation in an unit $(n-1)$-ball in n-dimensions, given specification of the boundary potential $g(y)$, $y \in \partial \mathcal{B}$, is given as

$$u(x) = \frac{1 - \|x\|^2}{\omega_n} \int_{\partial \mathcal{B}(0,1)} \frac{g(y)}{\|x - y\|^n} \, dS(y) , \tag{5}$$

where ω_n is the measure ("surface area") of the $(n-1)$-sphere boundary, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$, and $g(y) \in C(\partial\mathcal{B})$ [24, 60]. This formula is known as the *representation* or *Poisson integral* formula. The given $u(x)$ is a harmonic function, and extends continuously to the boundary provided that g is continuous [24, 60]. The specification of the boundary potential is known as a Dirichlet boundary condition.

Because the boundary conditions given above are discontinuous, we approximate them with a continuous transition over some interval contained in the outlet zone. In this case, the numerator of (5) is bounded above by one. Because the constant function is integrable (with trivial solution $u = 1$), we conclude the approximation is integrable by Lebesgue's bounded convergence theorem, and the integral over the transition region goes to zero as the interval goes to zero [39]. The above representation can be solved in closed form given certain symmetries for the boundary conditions. At worst, the function is computable, with a computable derivative representation. This allows us to directly calculate derivatives using numeric quadrature.

4.2.3 Special Case: Unit Disk in $\mathbb{R}^2$

The examples presented in this report are for planar systems, so we detail the calculations for the planar ball, or *disk*, in this section. Let $\mathcal{D} = \mathcal{B}(0,1) \subset \mathbb{R}^2$, where

$$\mathcal{D} = \left\{ (x_d, y_d) \in \mathbb{R}^2 \mid \sqrt{x_d^2 + y_d^2} < 1 \right\}.$$

The boundary is $\partial\mathcal{D} = \partial\mathcal{B} = \left\{ (x_d, y_d) \in \mathbb{R}^2 \mid \sqrt{x_d^2 + y_d^2} = 1 \right\}$, which is equivalent to the unit circle $\mathbf{S^1}$. For the disk, the most natural representation is in polar coordinates, so we define

$$\begin{aligned} \rho &= \sqrt{x_d^2 + y_d^2}, \\ \theta &= \operatorname{atan2}(y_d, x_d). \end{aligned}$$

The boundary is parameterized by an angle ζ, such that $(\cos\zeta, \sin\zeta) \in \partial\mathcal{D}$. We construct the homeomorphism $\varphi : \mathcal{P} \to \mathcal{D}$ such that the inlet zone is in $[\alpha_0, \alpha_1]$, where $-\pi \le \alpha_0 < \alpha_1 \le \pi$. Figure 10 shows the parameters for the polar coordinates.

For the unit radius boundary, $dS = d\zeta$, and we may rewrite (5) as

$$\begin{aligned} u(x) = u(\rho,\theta) &= \frac{1-\rho^2}{2\pi} \int_{\alpha_0}^{\alpha_1} \frac{1}{(\rho\cos\theta - \cos\zeta)^2 + (\rho\sin\theta - \sin\zeta)^2}\, d\zeta \\ &= \frac{1-\rho^2}{2\pi} \int_{\alpha_0}^{\alpha_1} \frac{1}{1+\rho^2 - 2\rho\cos(\zeta-\theta)}\, d\zeta, \end{aligned} \tag{6}$$

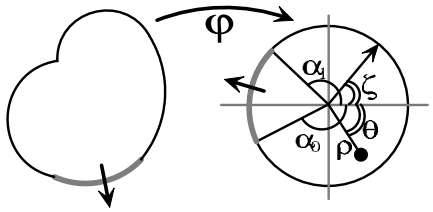


Figure 10: Variable specification for the unit disk.

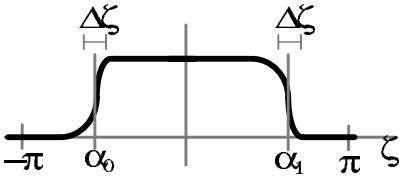


Figure 11: Discontinuous boundary conditions approximated by continuous functions as $\Delta\zeta \to 0$.

where $x = (\rho\cos\theta, \rho\sin\theta)$. Note, that we have modeled the discontinuities as shown in Figure 11, and taken the limit as $\Delta\zeta \to 0$.

The integral in (6) has a closed form solution, although some bookkeeping is required. Consider the indefinite integral

$$\int \frac{1}{1+\rho^2-2\rho\cos(\zeta-\theta)}\,d\zeta = \frac{2\tan^{-1}\left(\frac{(1+\rho)\tan\left(\frac{\zeta-\theta}{2}\right)}{-1+\rho}\right)}{-1+\rho^2}, \tag{7}$$

where $\tan^{-1} = \arctan$. This leads to the following solution via the fundamental theorem of calculus

$$\begin{aligned}\frac{1-\rho^2}{2\pi}\int_{\alpha_0}^{\alpha_1} \frac{1}{1+\rho^2-2\rho\cos(\zeta-\theta)}\,d\zeta &= -\frac{\tan^{-1}\left(\frac{(1+\rho)\tan\left(\frac{\alpha_1-\theta}{2}\right)}{-1+\rho}\right)}{\pi} \\ &\quad + \frac{\tan^{-1}\left(\frac{(1+\rho)\tan\left(\frac{\alpha_0-\theta}{2}\right)}{-1+\rho}\right)}{\pi}.\end{aligned} \tag{8}$$

If, in the range $[\alpha_0, \alpha_1]$, the argument for the tangent function crosses a discontinuity at $\pm\frac{\pi}{2} \pm m\pi$, where m is an integer, then the evaluation of (7) needs to be broken at the discontinuity. In this case, the arctan function will yield a total difference of π across the discontinuity, which leads to a difference of $+1$ with respect to the naive evaluation given in (8). Once derivatives are taken to evaluate the gradient, the constant $\pm m\pi$ term vanishes. By calculation, it can be shown that $\nabla^2 u = 0$ everywhere.

Another equally valid method of solving Laplace's equation, known as *separation of variables*, is applicable if the problem has natural symmetries [60]. As the name suggests, the technique involves separating the influence of the variables and solving using a Fourier series. The calculations, details of which can be seen in Appendix A, lead to

$$u(\rho,\theta) = a_0 + \sum_{m=1}^{\infty} \rho^m \left(a_m \cos(m\theta) + b_m \sin(m\theta)\right), \tag{9}$$

where a_m and b_m are the Fourier coefficients. Solving for the coefficients, and simplifying as shown in Section A, gives the final solution

$$\begin{aligned}u(\rho,\theta) = \frac{\alpha_1-\alpha_0}{2\pi} &+ \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin(\alpha_1-\theta)}{1-\rho\cos(\alpha_1-\theta)}\right) \\ &- \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin(\alpha_0-\theta)}{1-\rho\cos(\alpha_0-\theta)}\right).\end{aligned} \tag{10}$$

The solution has two singularities at $\theta = \alpha_0$ and $\theta = \alpha_1$ when $\rho = 1$, but is otherwise continuous. Again, a straightforward calculation shows that $\nabla^2 u = 0$ everywhere.

With appropriate bookkeeping, $\gamma_b(q_d) = u(\rho,\theta)$, where u is given in either (8) or (10), and $\rho = \sqrt{x_d^2+y_d^2}$ and $\theta = \text{atan2}(y_d, x_d)$ is the polar coordinate representation of $q_d \in \mathcal{D}$. Although the forms of the equations are quite different, they evaluate to the same potential and derivatives, as expected given the uniqueness of solutions to PDE with Dirichlet boundary conditions.

4.3 Component Control Policies

Given the potential field γ for the $n-$cell $\subset \mathbb{R}^n$, generated as described above, we wish to generate control policies that follow integral curves of the negative gradient vector field. By construction, the gradient vector

field induced by the pullback of γ_b is orthogonal to the boundary of the cell. This is trivial to show, given that the potential along the boundary of the cell is constant by virtue of the pull-back. The negative gradient vector field is inward pointing along the inlet zone of the cell, and outward pointing along the outlet zone of the cell. Orthogonality allows us to construct control policies that have continuity of orientation across the cell boundaries, and facilitates some of our later proofs. Because the gradient field is nowhere zero on the cell, and the field is C^k smooth, we are guaranteed that any trajectory that follows the integral curves of the negative gradient field will exit the cell through the designated outlet zone.

Although following the integral curves has the desired effect with regard to exiting the cell, the dynamic behavior may not be desired. For example, the potential function tends to be flat near the boundary at the maximal distance from the outlet zone, which leads to a small gradient. The gradient tends to be largest near the boundary condition discontinuities. Our approach is to make use of the negative *normalized* gradient of the pulled-back potential function to generate a vector field $\hat{X} : \mathcal{P} \rightarrow \mathcal{TP}$, given as

$$\hat{X}(q) = -\frac{D_q\gamma^T}{||D_q\gamma||} = -\frac{D_q\varphi^T\, D_{\varphi(q)}\gamma_b^T}{\left\|D_{\varphi(q)}\gamma_b\, D_q\varphi\right\|}\,. \tag{11}$$

where $q \in \mathcal{P}$ and $q_b = \varphi(q) \in \mathcal{B}$. The integral curves of $\hat{X}(q)$ correspond to the integral curves of the negative gradient field. Figure 12 shows an example of the vector field $\hat{X}(q)$. We are free to scale the vector field by any non-zero scalar to generate the desired speed profile along the integral curves. Given a scalar field, $s(q)$, defining the desired speed at each point, we define the desired velocity vector field as

$$X(q) = s(q)\,\hat{X}(q)\ . \tag{12}$$

In the subsections that follow, we construct control policies that follow the integral curves of $X(q)$ for both kinematic and dynamical systems.

4.3.1 Kinematic Control Policy

For kinematic systems, where velocity is controlled directly, integral curves of the vector field $X(q)$ can be exactly followed.

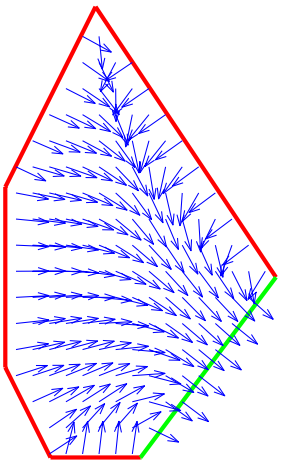


Figure 12: Negative normalized gradient vector field for a polygonal cell.

Lemma 4.1 *For a kinematic system of the form $\dot{q} = u$, where $q, u \in \mathbb{R}^n$, the integral curves of the system under the influence of $u = X(q) = s(q)\,\hat{X}(q)$, where $s(q) \in (0, \mathrm{V}_{\max}]$ and $\hat{X}(q)$ is defined in (11), determine a path from any point on the inlet zone, or interior of the cell, to a point on the outlet zone such that the trajectory is completely contained in the cell until it crosses the outlet zone.*

Proof: By construction, the potential over the cell is maximum on the inlet boundary, minimum on the outlet boundary, and free of local minima. The system is performing gradient descent by moving in the $\hat{X}(q)$ direction, which moves the configuration from a point on the inlet zone or interior of the cell through the outlet zone.

□

This is exactly the behavior required of the component control policy for use in the control policy deployment scheme from Section 3.

Combined with the automated control policy deployment scheme described in Section 3.4, the component control policy $u = X(q)$ yields a (resolution) complete motion planning method for the kinematic system via the induced overall control policy. Because the vector field is orthogonal to the boundaries, the velocity orientation is continuous across the boundaries. Speed continuity is dependent on specification of $s(q)$ in each cell. For the kinematic system, the most natural speed specification is $s(q) = s^* < \mathrm{V}_{\max}$, where s^* is some constant. It might be desirable to define lower speeds in regions where sharp turns are being made, but we defer discussion of how to do this until Section 4.3.3.

4.3.2 Unconstrained Dynamics Control Policy

While the kinematic control policy deployment scheme, with the component control policy design presented above, provides a novel solution to the classical mobile robot navigation problem, it does not guarantee that a real robot, subject to dynamical constraints, could follow the prescribed path.

Given a second order system of the form

$$\ddot{q} = u\,, \tag{13}$$

we wish to design a control law that converges to an integral curve of the vector field $X(q)$, without departing the defined cell $\mathcal{P}$. We begin by developing the control policy assuming the system allows arbitrary, but still finite, accelerations. Later, we consider acceleration and velocity constraints.

Using $X(q)$ as a position dependent velocity reference leads to a natural velocity reference control policy of the form

$$u = K\,(X(q) - \dot{q}) + \dot{X}(q)\,, \tag{14}$$

where $K > 0$ is the "velocity regulation" gain, which acts to decrease the error [59]. The feed-forward term, $\dot{X}(q) = D_q X\,\dot{q}$, accounts for the change in the vector field as we move in the $\dot{q}$ direction, and allows the system to exactly track the integral curves of $X(q)$.

Lemma 4.2 *In the absence of constraints, including those of the cell boundary, the integral curves of the vector field $X(q)$ are attractive to the trajectories of the closed loop system defined by (13) under the influence of (14).*

Proof: Define the velocity error, $e = X(q) - \dot{q}$, and consider the set

$$\mathcal{V} := \{(q, \dot{q}) \mid ||e|| = 0\}\,.$$

Define a Lyapunov-like function of the form

$$\begin{aligned} \eta_v &= \frac{1}{2} e^T e \\ &= \frac{1}{2} \left(X(q) - \dot{q}\right)^T \left(X(q) - \dot{q}\right) \end{aligned} \tag{15}$$

Evaluating the time derivative of (15) along the trajectories of the closed loop system, and substituting (14) yields

$$\begin{aligned} \dot{\eta_v} &= \left(X(q) - \dot{q}\right)^T \left(\dot{X}(q) - \ddot{q}\right) \\ &= \left(X(q) - \dot{q}\right)^T \\ &\quad \left(\dot{X}(q) - K \left(X(q) - \dot{q}\right) - \dot{X}(q)\right) \\ &= -K \left(X(q) - \dot{q}\right)^T \left(X(q) - \dot{q}\right) . \end{aligned} \tag{16}$$

For $K > 0$, $\dot{\eta_v} < 0$ for all non-zero velocity error, and we conclude that the set $\mathcal{V}$ is both attractive and invariant, implying that the velocity error asymptotically approaches zero [59, 62].

□

The orientation error, defined as the angle between the desired velocity, $X(q)$, and the current velocity, $\dot{q}$, is given by

$$\vartheta = \cos^{-1} \frac{\dot{q}^T X}{\sqrt{\dot{q}^T \dot{q} X^T X}} , \tag{17}$$

where $X = X(q)$.

Lemma 4.3 *In the absence of acceleration constraints, and for initial velocities such that $\dot{q}^T X > 0$, there exists a lower bound on K such that the orientation error, ϑ, monotonically decreases.*

Proof: First, consider the isolated case where $\dot{q} = 0$. We define $\vartheta \mid_{\dot{q}=0} = 0$, since differentially the acceleration will be in the direction of the desired velocity and the orientation error will be zero. As we show below, the orientation error will remain zero for all time.

To show that the orientation error ϑ monotonically decreases, consider the set

$$\mathcal{U} := \{(q, \dot{q}) \mid \vartheta = 0\} ,$$

and define a Lyapunov-like function of the form

$$\begin{aligned} \eta_u &= \sin^2 \vartheta = 1 - \cos^2 \vartheta \\ &= \frac{\dot{q}^T \dot{q} X^T X - \dot{q}^T X \dot{q}^T X}{\dot{q}^T \dot{q} X^T X} , \end{aligned} \tag{18}$$

where $X = X(q)$.

Evaluating the time derivative of (18) along the trajectories of the closed loop system, and simplifying yields

$$\begin{aligned}
\dot{\eta_u} &= \frac{2\left(X^T X\,\dot{q}^T\ddot{q} + \dot{q}^T\dot{q}\,X^T\dot{X} - \dot{q}^T X\,\dot{q}^T\dot{X} - \dot{q}^T X\,X^T\ddot{q}\right)}{\dot{q}^T\dot{q}\,X^T X} \\
&\quad - \frac{2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(X^T X\,\dot{q}^T\ddot{q} + \dot{q}^T\dot{q}\,X^T\dot{X}\right)\left(\dot{q}^T\dot{q}\,X^T X - \dot{q}^T X\,\dot{q}^T X\right) \\
&= \frac{-2}{\dot{q}^T\dot{q}\,X^T X}\left(\dot{q}^T X\,\dot{q}^T\dot{X} + \dot{q}^T X\,X^T\ddot{q}\right) + \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(X^T X\,\dot{q}^T\ddot{q} + \dot{q}^T\dot{q}\,X^T\dot{X}\right) \qquad (19)
\end{aligned}$$

Substituting (14) into (19), we obtain

$$\begin{aligned}
\dot{\eta_u} &= \frac{-2}{\dot{q}^T\dot{q}\,X^T X}\left(\dot{q}^T X\,\dot{q}^T\dot{X} + K\dot{q}^T X\,X^T X - K\dot{q}^T X\,X^T\dot{q} + \dot{q}^T X\,X^T\dot{X}\right) \\
&\quad + \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(KX^T X\,\dot{q}^T X - KX^T X\,\dot{q}^T\dot{q} + X^T X\,\dot{q}^T\dot{X} + \dot{q}^T\dot{q}\,X^T\dot{X}\right) \\
&= -\frac{2}{\dot{q}^T\dot{q}\,X^T X}\left(\dot{q}^T X\,\dot{q}^T\dot{X} + \dot{q}^T X\,X^T\dot{X}\right) \\
&\quad + \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(X^T X\,\dot{q}^T\dot{X} + \dot{q}^T\dot{q}\,X^T\dot{X}\right) \\
&\quad - \frac{2}{\dot{q}^T\dot{q}\,X^T X}\left(K\dot{q}^T X\,X^T X - K\left(\dot{q}^T X\right)^2\right) \\
&\quad + \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(KX^T X\,\dot{q}^T X - KX^T X\,\dot{q}^T\dot{q}\right) \\
&= \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(X^T X\,\dot{q}^T\dot{X} + \dot{q}^T\dot{q}\,X^T\dot{X} - \frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}\dot{q}^T\dot{X} - \frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}X^T\dot{X}\right) \\
&\quad - K\frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left(\frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}X^T X - \dot{q}^T\dot{q}\,X^T X - X^T X\,\dot{q}^T X + X^T X\,\dot{q}^T\dot{q}\right) \\
&= \frac{2\left(\dot{q}^T X\right)^2}{\left(\dot{q}^T\dot{q}\,X^T X\right)^2}\left[\left(X^T X\,\dot{q}^T + \dot{q}^T\dot{q}\,X^T - \frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}\dot{q}^T - \frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}X^T\right)\dot{X}\right. \\
&\qquad \left. - K\left(\frac{\dot{q}^T\dot{q}\,X^T X}{\dot{q}^T X}X^T X - X^T X\,\dot{q}^T X\right)\right] \qquad (20)
\end{aligned}$$

Now consider the case where $\dot{q}$ is aligned with $X(q)$ and we have $\dot{q}^T X = ||\dot{q}||\,||X||$. We assume that $||\dot{q}|| > 0$, and $||X|| > 0$ by construction, so the leading term is a finite positive number. Because $\dot{q}^T\dot{q} = ||\dot{q}||^2$ and $X^T X = ||X||^2$, both parenthetical terms inside the brackets of (20) are zero, and $\dot{\eta_u} = 0$. Therefore, we conclude that the set $\mathcal{U}$ is invariant. In other words, if the orientation error is zero, it remains zero.

Away from $\mathcal{U}$, the leading term of (20) is positive and bounded, because we assume that initially $\dot{q}^T X > 0$. Assuming the system has finite initial velocity and that $||X(q)||$ is finite, if follows that the velocity error is finite; then by Lemma 4.2, the error magnitude decreases, and we conclude that $||\dot{q}||$

remains finite for all time. The first parenthetical term in brackets for (20) has an indeterminate sign, but is finite since all the terms are bounded.

We may rewrite the parenthetical portion of the second term as

$$\begin{aligned}\left(\frac{\dot{q}^T\dot{q}\,X^TX}{\dot{q}^TX}X^TX - X^TX\,\dot{q}^TX\right) &= \frac{\dot{q}^T\dot{q}\,\left(X^TX\right)^2}{\dot{q}^TX}\left(1-\frac{\left(\dot{q}^TX\right)^2}{\dot{q}^T\dot{q}\,X^TX}\right)\\ &= \frac{\dot{q}^T\dot{q}\,\left(X^TX\right)^2}{\dot{q}^TX}\left(1-\cos^2\vartheta\right)\ . \qquad (21)\end{aligned}$$

Because (21) is non-negative for all $\dot{q}$ such that $\dot{q}^TX > 0$, the overall impact of the second term in brackets for (20) is to decrease $\dot{\eta}_u$. Therefore, for *sufficiently large* K, $\dot{\eta}_u$ can be made negative definite. This implies that $\dot{q}^TX$ remains positive, and therefore $\dot{\eta}_u$ is always negative for sufficiently large K. Since $\dot{\eta}_u < 0$, we conclude that $\mathcal{U}$ is attractive and invariant, and that ϑ monotonically decreases under the influence of (14).

□

Intuitively, making K sufficiently large ensures that the control policy is correcting more quickly than the vector field is changing. Formally, the sufficiently large K is determined such that

$$K > \max_{q,\dot{q}} \frac{\left(\dot{q}^TX\,X^TX\,\dot{q}^T + \dot{q}^TX\,\dot{q}^T\dot{q}\,X^T - \dot{q}^T\dot{q}\,X^TX\dot{q}^T - \dot{q}^T\dot{q}\,X^TX\,X^T\right)\dot{X}}{\left(\dot{q}^T\dot{q}\,\left(X^TX\right)^2 - X^TX\,\left(\dot{q}^TX\right)^2\right)}\ .$$

This is a worst case limit based on the vector field derivative.

Lemma 4.4 *In the absence of acceleration constraints, with sufficiently large K and initial velocities such that $\dot{q}^TX > 0$, the trajectories of the closed loop system defined by (13) under the influence of (14), converge to the integral curves of the vector field $X(q)$ in such a way that the trajectory never exits the cell except by the outlet zone, and in fact exits the cell via the outlet zone.*

Proof: For initial velocities such that $\dot{q}^TX > 0$, we know the orientation error is initially less than $\frac{\pi}{2}$. By Lemma 4.3, for sufficiently large K the orientation error is monotonically decreasing.

Assume the trajectory exits the cell in the *inlet* zone. At the point of departure, $\dot{q}^TX < 0$ given the inward pointing vector field orthogonal to the cell boundary. This implies that $\vartheta > \frac{\pi}{2}$, requiring that the orientation error increased along its trajectory. This contradicts Lemma 4.3.

Since the vector field $X(q)$ is nowhere zero over the cell, the system cannot come to rest and remain stationary, because the system experiences an acceleration along the vector field. Therefore, we conclude that the trajectory must leave the cell via the outlet zone under the influence of (14) for the given conditions.

□

The utility of lemmas 4.3 and 4.4 is limited by two factors. First, a large value for K can lead to an overly aggressive policy over the cell that may prove troublesome for implementation. Secondly, and most importantly, all real world systems have acceleration limits, which may very well be violated by the feedforward term of (14), regardless of the value of K and the velocity error.

4.3.3 Constrained Dynamics Control Policy

To extend our ideas to real world systems, we now consider the following dynamic constraints,

$$||\dot{q}|| \leq \mathrm{V}_{\max} \tag{22}$$
$$||u|| = ||\ddot{q}|| \leq \mathrm{A}_{\max}\,. \tag{23}$$

The velocity limit is taken to be a safety limit, and it is assumed that $||X(q)|| \leq \mathrm{V}_{\max}$ for all $q \in \mathcal{P}$. However, if the change in the vector field $X(q)$ given by $D_q X\,\dot{q}$ is aligned with the current velocity, then the feed-forward term of (14) may act to increase the velocity magnitude (speed), causing a violation of the maximum speed. We will modify the velocity reference control law in (14) to prevent this violation from occurring, but first we consider the acceleration limit.

The acceleration limit represents a physical limitation of the dynamical system. Since the acceleration cannot be exceeded, the inability to produce the desired acceleration will invalidate the lemmas given above. To accommodate both the velocity and acceleration constraints, we have modified our basic control policies, and developed hybrid control policies to address the dynamical constraints. We begin with a discussion of the acceleration constraints, and then discuss the hybrid strategies in Section 4.4.

As discussed in Section 4.3.2, the feed-forward term of (14) may cause a violation of the acceleration constraint regardless of the value of K and the velocity error. Consider the case where we are exactly tracking the integral curves of $X(q)$, and $V_e = X(q) - \dot{q} = 0$. Then, by (14), we have

$$u_0 = D_q X\,\dot{q} = D_q X\; X(q)\;,$$

where u_0 is the zero error input. To satisfy our constraint,

$$||u_0|| = ||D_q X\; X(q)|| \leq \mathrm{A}_{\max}\,. \tag{24}$$

This norm depends on both $D_q X$ and $\dot{q}$. If the system velocity is high, then large accelerations are needed to follow even modest curves in the vector field. If the vector field is changing significantly over small distances, then even modest velocities can require significant accelerations.

First consider $X(q) = s^*\hat{X}(q)$, where $s(q) = s^*$ is a constant (i.e., $X(q)$ is a constant speed field). In order to trace the integral curve given the velocity reference control policy from above,

$$||D_q X\; X(q)|| \leq \mathrm{A}_{\max}\,,$$
$$(s^*)^2 \left\|D_q\hat{X}\; \hat{X}(q)\right\| \leq \mathrm{A}_{\max}\,,$$

where $D_q X = s^*\, D_q\hat{X}$. By definition, $\left\|D_q\hat{X}\; \hat{X}(q)\right\| \leq \left\|\!\left| D_q\hat{X} \right|\!\right\|$, where $\left\|\!\left| D_q\hat{X} \right|\!\right\|$ is the spectral norm. The spectral norm of a matrix M, denoted $\|\!|M|\!\|$, is defined as

$$\|\!|M|\!\| = \max_{||x||=1} ||M\,x||\;.$$

Figure 13 shows a plot of $\left\|\!\left| D_q\hat{X} \right|\!\right\|$ for an example polygonal cell. Conservatively, if

$$s^* \leq \min_q \sqrt{\frac{\mathrm{A}_{\max}}{\left\|\!\left| D_q\hat{X} \right|\!\right\|}}\,,$$

then the system will not exceed the acceleration bound so long as the velocity magnitude at q does not exceed s^*. Unfortunately, this is overly conservative.

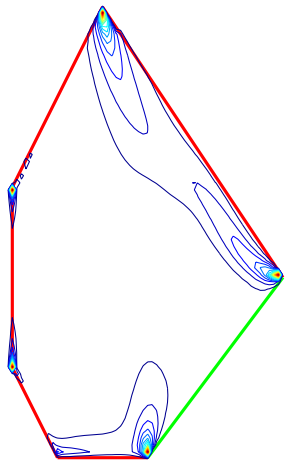


Figure 13: Spectral norm of the derivative of the negative normalized gradient vector field ($\left\| D_q \hat{X} \right\|$) for a polygonal cell, with the cell boundary shown. The contours corresponding to the largest norms are located near the polygon vertices.

We now use the spectral norm of $D_q \hat{X}$ to encode the idea of slowing down while turning. If we let $X(q) = s(q)\ \hat{X}(q)$, where

$$s(q) = \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda}, \tag{25}$$

with λ a constant, then $X(q)$ becomes a variable speed vector field. The vector field derivative $D_q X$ is

$$\begin{aligned} D_q X &= \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} D_q \hat{X} - \frac{s^*}{\left(\left\| D_q \hat{X} \right\| + \lambda \right)^2} \hat{X}\, D_q \left\| D_q \hat{X} \right\| \\ &= \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} \left(D_q \hat{X} - \frac{1}{\left\| D_q \hat{X} \right\| + \lambda} \hat{X}\, D_q \left\| D_q \hat{X} \right\| \right) \\ &= \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} G(q)\ , \end{aligned}$$

where $G(q) := \left(D_q \hat{X} - \frac{1}{\left\| D_q \hat{X} \right\| + \lambda} \hat{X}\, D_q \left\| D_q \hat{X} \right\| \right)$.

We now have the constraint that

$$\begin{aligned} \|D_q X \; X(q)\| &\leq \mathrm{A}_{\max}, \\ \left\| \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} G(q) \;\; \frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} \hat{X}(q) \right\| &\leq \mathrm{A}_{\max}, \\ \left(\frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda} \right)^2 \left\| G(q) \;\; \hat{X}(q) \right\| &\leq \mathrm{A}_{\max}. \end{aligned}$$

Again, being somewhat conservative, if we specify

$$s^* \leq \min_q \frac{\sqrt{\mathrm{A}_{\max}} \left(\left\| D_q \hat{X} \right\| + \lambda \right)}{\sqrt{\left\| D_q \hat{X} - \frac{\hat{X} \, D_q \left\| D_q \hat{X} \right\|}{\left\| D_q \hat{X} \right\| + \lambda} \right\|}}, \tag{26}$$

then the system will not exceed the acceleration bound so long as the velocity magnitude at q does not exceed $\frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda}$. Since $\left\| D_q \hat{X} \right\|$ is maximal where the normalized vector field is changing most, this form encodes the idea of slowing down while turning, as desired.

This form is still somewhat conservative, but is available in closed form. One can imagine solving a non-linear PDE to find a more aggressive speed scaling for $s(q)$. However, this would require numerical techniques, and would lose the analytic proofs, without giving much insight into the problem.

We make one final change to incorporate the velocity constraints, and define

$$s(q) = \min \left(\frac{s^*}{\left\| D_q \hat{X} \right\| + \lambda}, \mathrm{V}_{\max} \right). \tag{27}$$

Although this form allows the reference velocity control policy to make use of the dynamical capabilities of the system, the form is still not sufficient to prevent constraint violations. In the case where initial velocity is not aligned with the vector field, the proportional term may cause the acceleration constraint to be violated. To prevent this, we construct our component control policy from a set of hybrid control policies designed to cause the system to converge to the integral curves without violating the constraints.

4.4 Hybrid Control Policies for Dynamical Systems

To understand the need for hybrid control strategies over each cell for dynamical systems of the form given in (13), it is helpful to understand the interaction between level sets of the Lyapunov-like functions defined in Lemmas 4.2 and 4.3 and the constraint surfaces defined by (22) and (23). Consider an element in the $2n$-dimensional state space $(q, \dot{q})^T$, where $q \in \mathcal{P}$ is a configuration in the cell $\mathcal{P}$, and $\dot{q} \in \mathcal{T}_q \mathcal{P}$ is an element of the tangent space at q. Level sets of the Lyapunov-like functions from Section 4.3.2 define surfaces in the state space. The velocity constraint given in (22) defines a compact set of permissible velocities $\mathcal{V}$ in the state space at each point in the region such that

$$\mathcal{V} := \{(q, \dot{q}) \mid \|\dot{q}\| \leq \mathrm{V}_{\max}\}. \tag{28}$$

The acceleration constraint given in (23) defines a set of permissible states $\mathcal{A}$ such that

$$\mathcal{A} := \{(q, \dot{q}) \mid \|\ddot{q}\| = \|u\| \leq \mathrm{A}_{\max}\}, \tag{29}$$

where u is defined in (14), and depends on q, $\dot{q}$, and K. Furthermore, define the set $\mathcal{Z}$ of velocities that are somewhat aligned with the desired velocity $\left(\vartheta < \frac{\pi}{2}\right)$ as

$$\mathcal{Z} := \left\{ (q, \dot{q}) \mid X(q)^T \dot{q} > 0 \right\} . \tag{30}$$

All of these sets, $\mathcal{V}$, $\mathcal{A}$, and $\mathcal{Z}$, are defined purely by functions of the state given the vector field $X(q)$.

If a level set $\eta_v^{-1}(c)$ for some $c \in \mathbb{R}$, where $\eta_v : \mathcal{T}_q\mathcal{P} \to \mathbb{R}$ was defined in Lemma 4.3, is completely contained in $\mathcal{V} \cap \mathcal{A} \cap \mathcal{Z}$ for a sufficiently large K, then by Lemmas 4.3 and 4.4 we may invoke the controller specified by (14) for any state with $\eta_v < c$ with guarantees that the system will exit the region properly. Define c as

$$c = \arg\max_{\lambda} | \eta_v^{-1}(\lambda) \subset \mathcal{V} \cap \mathcal{A} \cap \mathcal{Z} .$$

If $\eta_v > c$, or if c does not exist, then it is possible that a trajectory will violate the either the maximum velocity or acceleration constraints. In addition to the difficulty of actually determining c, the set of states with $\eta_v \leq c$ is likely to be overly restrictive.

Our approach, again taking inspiration from the sequential composition methods of [11, 54, 59], is to define a set of controllers that cover the region of the free state space, and whose composition guarantees that the system exits the cell via the outlet zone with speed less than $||X(q)||$. For our system, we define three control policies: Φ_S, Φ_A, and Φ_T. Where the subscripts S, A, and T refer to "Save," "Align," and "Track" respectively. The control policies are designed such that

$$\Phi_S \succeq \Phi_A \succeq \Phi_T .$$

4.4.1 Save Control Policy

We begin by considering the case where the best the system can do, using all available acceleration, is prevent collision with the cell boundary. The Save control policy, Φ_S, is used to apply all available acceleration in way that prevents collision with the cell boundaries *if it is at all possible*.

The exact form of the Save control policy is dependent on the structure of the cell. In Section 5.1.3, we present the formulation for cells of arbitrary convex polytopes. For now, we assume that Φ_S is capable of bringing to rest any condition in $\mathcal{TP}$ that could be brought to rest without violating the given constraints [59].

By our construction and composition methods outlined below, collision with cell boundaries will only occur for trajectories entering too fast from outside the cell via the inlet zone, or for a non-zero initial condition that is too fast. The domain of the emergency stopping controller $\mathscr{D}(\Phi_S)$, which we term the *savable set* $\mathcal{S}$, is the set of all $(q, \dot{q}) \in \mathcal{TP}$ such that Φ_S can bring the system to rest without colliding with the cell boundary in the inlet zone.

Define the the collision ratio, ζ_c to be the ratio of the distance to collision, d_c, to the distance required for braking, d_b; that is

$$\zeta_c = \frac{d_c}{d_b} .$$

This definition is made concrete in Section 5.1.3 for cells defined as convex polytopes. Note, that if $\zeta_c < 1$ then collision can be avoided, while $\zeta_c > 1$ implies that collision is inevitable. For $\zeta_c < 1$, Φ_S is designed to reduce ζ_c further. We formally define the domain of Φ_S as

$$\mathscr{D}(\Phi_S) = \{ (q, \dot{q}) \mid q \in \mathcal{P} , \zeta_c < 1 \} .$$

The goal set of the Save control policy, $\mathscr{G}(\Phi_S)$, is any rest condition within the cell, or more formally

$$\mathscr{G}(\Phi_S) = \{ (q, \dot{q}) \mid q \in \mathcal{P} , \dot{q} = 0 \} .$$

The goal set of the Save control policy is in the domain of the Align or Track control policies because starting from rest we can follow the integral curve passing through q by Lemma 4.3, provided the vector field is defined as in (25). The savable set is positive invariant under the Save control policy because Save does not increase the collision ratio, ζ_c, for any state in the savable set.

4.4.2 Align Control Policy

The Align control policy is designed apply maximum acceleration to system in order to bring the velocity into the domain of the Track control policy as quickly as possible, in the case where collision with the cell boundaries is not imminent.

The Align control policy is designed to continuously transition from the Save control policy to a condition where maximum acceleration is applied along the velocity error vector. This later condition acts to decelerate the system and turn the velocity toward the desired velocity vector, $X(q)$. The domain of the Align control, given the collision ratio defined above, is

$$\mathscr{D}(\Phi_A) = \{(q\,,\,\dot{q}) \mid q \in \mathcal{P}\,,\, \zeta_c \leq \mu\}\ ,$$

where $\mu \in (0,1)$ is a user defined parameter defining the collision avoidance margin.

Let

$$v = \frac{\mu - \zeta_c}{\mu}\,,$$

and define the Align control policy as

$$\Phi_A\ :\ u = \begin{cases} \mathrm{A_{max}}\ \frac{(1-\sigma(v))\,\Phi_S + \sigma(v)\,\hat{e}}{\|(1-\sigma(v))\,\Phi_S + \sigma(v)\,\hat{e}\|} & \dot{q}^T X \leq \dot{q}^T\dot{q} \\ \mathrm{A_{max}}\ \frac{(1-\sigma(v))\,\Phi_S - \sigma(v)\,\hat{\dot{q}}}{\|(1-\sigma(v))\,\Phi_S - \sigma(v)\,\hat{\dot{q}}\|} & \text{otherwise} \end{cases}\ , \tag{31}$$

where $\hat{e} = \frac{X(q)-\dot{q}}{\|X(q)-\dot{q}\|}$, $\hat{\dot{q}} = \frac{\dot{q}}{\|\dot{q}\|}$, and $\sigma : v \to [0,1]$ is a transition function with $\sigma(0) = 0$ and $\sigma(1) = 1$. For demonstrations in this report, $\sigma(v) = \sqrt{v}$ has been used. The Align control policy is guaranteed to keep the system in its domain, as the policy transitions to the Save control policy at the domain boundary. Recall that under the Save control policy, the collision ratio ζ_c is guaranteed to not increase (see Section 5.1.3 for proof). Therefore, the system will not exit the $\zeta_c \leq \mu$ domain. Figure 14 shows an example of the vector relationships used by the Align control policy.

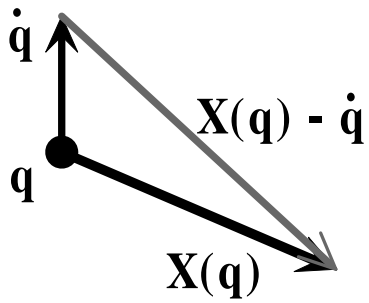


Figure 14: Velocity vector relationships for the Align control policy.

Because the domain, $\mathscr{D}(\Phi_A) \subset \mathcal{S}$, the worst the Align control policy will do is bring the system to rest, which is within the domain of the Track control policy. In the normal case the Align control policy will bring the system velocity orientation towards the desired velocity orientation, while at the same time reducing the speed of the system. If acceleration along the unit error vector would tend to increase the velocity, *i.e.* when $\dot{q}^T\dot{X} > \dot{q}^T\dot{q}$, the system switches to accelerate against the current velocity. In all regions, the Align control policy acts to decrease the system speed. The goal set of the Align control policy, $\mathscr{G}(\Phi_A)$, is

$$\mathscr{G}(\Phi_A) = \{(q, \dot{q}) \mid q \in \mathcal{P}, \dot{q} = 0\} ,$$

which prepares the Track control policy.

4.4.3 Track Control Policy

The Track control policy is designed to bring the system velocity into alignment with the vector field $X(q)$ using maximum available acceleration and transition continuously to the velocity reference control law. The domain of the Track control policy is

$$\mathscr{D}(\Phi_T) = \left\{(q, \dot{q}) \mid \dot{q}^T X > 0, \|\dot{q}\| \leq \|X(q)\|\right\} .$$

At the same time, the Track control policy must guarantee that the system trajectory does not exit the cell, other than by the outlet zone. The goal of the Track control policy is

$$\mathscr{G}(\Phi_T) = \{(q, \dot{q}) \mid \|\dot{q}\| \leq \|X(q)\|, q \in \partial\mathcal{P}_{\text{outlet}}\} ,$$

i.e. the system exits via the outlet zone.

To accomplish this goal, the Track control policy is designed to monotonically decrease the orientation error, ϑ, between the current velocity and the desired velocity. The approach is to use some of the available acceleration to keep the orientation error constant as the trajectory evolves, and use the remainder of the available acceleration to decrease the error. The vector field derivative, $\dot{X} = D_q X \dot{q}$, defines the amount the desired velocity, $X(q)$, changes as the system moves by $\dot{q}$. Let dQ be the acceleration vector applied to the system such that the change in orientation error is zero. Essentially, dQ, shown in Figure 15, is a scaled version of $\dot{X}$ that has been rotated by ϑ.

Consider the plane defined by the current velocity, $\dot{q}$, and the desired velocity, $X(q)$, which we will term the *velocity plane*. We will decompose the vector field derivative vector into three components: the component along the desired velocity, the amount orthogonal to the desired velocity in the velocity plane, and the remainder. The component along the desired velocity is the differential speed change. The second component encodes how the desired velocity vector differentially rotates in the velocity plane. The remainder encodes how the velocity plane differentially rotates in space. If the system is accelerated such that the current velocity differentially rotates in the velocity plane the same as the desired velocity, and rotates with the velocity plane, then the change in the orientation error will be zero. Define the following unit vectors: $\hat{\dot{q}}$, $\hat{M}$, $\hat{N}$, and $\hat{P}$, where $\hat{\dot{q}}$ is the unit vector along the current velocity, $\hat{M}$ is the orthogonal to the desired velocity in the direction given by the error vector $e = X(q) - \dot{q}$, $\hat{N}$ is the unit vector orthogonal to the current velocity in the direction of $\hat{M}$, and $\hat{P}$ is the unit vector orthogonal to the velocity plane. These vectors are shown in Figure 15. Note, that $\hat{M}$ and $\hat{N}$ are both in the velocity plane. If the current and desired velocities are aligned, we define $\hat{M} = \hat{N} = \mathbf{0}$.

Let $x = \hat{X}^T\dot{X}$ and $m = \hat{M}^T\dot{X}$, and define

$$P = \dot{X} - x\hat{X} - m\hat{M} .$$

The vector P, orthogonal to both $\hat{X}$ and $\hat{M}$, defines how the desired velocity vector rotates out of the velocity plane. The scalar x defines the differential speed change, and the scalar m defines how the desired velocity

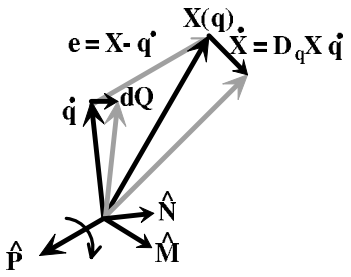


Figure 15: Vector relationships for the Track control policy.

vector rotates in the velocity plane. Considering the different magnitudes of $\dot{q}$ and $X(q)$,

$$dQ = \frac{||\dot{q}||}{||X(q)||}\left(x\,\hat{\dot{q}} + m\,\hat{N} + P\right).$$

This is equivalent to scaling $\dot{X}$, and rotating in the velocity plane by ϑ. Given the scaling from Section 4.3.3, $||dQ|| \leq \mathrm{A_{max}}$ because $||\dot{q}|| \leq ||X(q)||$ in $\mathscr{D}(\Phi_T)$, and $\left\|\dot{X}\right\| \leq \mathrm{A_{max}}$. Letting $u = dQ$ will hold the orientation error constant, while allowing the speed to change proportionally. Note that the speed can never exceed the desired speed under this control, because the speed change will be equivalent when the current speed equals the desired speed. In general, because $||dQ|| \leq \mathrm{A_{max}}$, there will be some acceleration capacity left over to decrease the orientation error. Let $u = dQ + K^*\left(X(q) - \dot{q}\right)$, where K^* is calculated to use the remaining acceleration capacity. This control will decrease the orientation error, or at worst keep the error constant. However, the available control can be used more efficiently.

Consider, if $m < 0$, then the vector field is changing in a way that is already decreasing the orientation error. Also, if the speed change given by x is positive, then we can safely ignore this component, assuming we prefer alignment over speed matching. We redefine dQ such that

$$dQ = \frac{||\dot{q}||}{||X(q)||}\left(\min(0,x)\;\hat{\dot{q}} + \max(0,m)\;\hat{N} + P\right),. \tag{32}$$

and preferentially use the available acceleration for steering, then use any remaining acceleration for speed regulation. The Track control policy is defined as

$$\Phi_T\,:\,u = dQ + s\,\hat{N} + a\,\hat{\dot{q}}, \tag{33}$$

where

$$s = \min\left(K\,\hat{N}^T e,\,\sqrt{{\mathrm{A_{max}}}^2 - \frac{\dot{q}^T\dot{q}}{X^T X}\left(\mathrm{P}^T\mathrm{P} + \min(0,x)^2\right)} - \frac{||\dot{q}||}{||X||}\max(0,m)\right) \text{ and}$$

$$a = \min\left(K\hat{\dot{q}}^T e,\sqrt{{\mathrm{A_{max}}}^2 - \frac{\dot{q}^T\dot{q}}{X^T X}\mathrm{P}^T\mathrm{P} - \left(\frac{||\dot{q}||}{||X||}\max(0,m) + s\right)^2} - \frac{||\dot{q}||}{||X||}\min(0,x)\right).$$

Lemma 4.5 *Under the influence of (33), the system of (13) with constraints given in (22) and (23) converges to the integral curves of $X(q)$, defined as in (25), in such a way that the trajectory never exits the cell except by the outlet zone, and in fact exits the cell via the outlet zone.*

The proof directly follows that of Lemma 4.4.

In the limit, as the velocity error approaches zero, the Track control policy is identical to the velocity reference control policy given in (14). We assume that the vector field $X(q)$ is defined as in (25), and as such the velocity reference control policy is able to follow the integral curves without violating the constraints. Therefore, once the velocity $\dot{q}$ has converged to an integral curve such that $||\dot{q}|| \leq s(q)$, lemmas 4.3 and 4.4 are applicable. Note, that since the orientation error is zero, the value of K only needs to be greater than zero for the lemmas to hold, and not *sufficiently large*.

4.4.4 Composite Policy

Given Φ_S, Φ_A, and Φ_T, any initial state in the savable set will be brought into the domain of the Track control policy and exit the cell via the outlet zone. Given the prepares relationship,

$$\Phi_S \succeq \Phi_A \succeq \Phi_T \,,$$

the highest priority control policy takes control until the system exits the cell. As the Align policy transitions to the Save policy, the Save policy is superfluous over a cell where the Align policy is deployed.

The Align and Track policies may be thought of as separate policies over the same cell, to be deployed individually, or as part of a single switched policy over a given cell. We define the composite control policy, which we term the Flow policy, as $\Phi_F = \bigvee \{\Phi_T, \Phi_A\}$. Thus, we can think of deploying individual Align and Track policies, or one Flow policy, over a given cell.

Since the vector fields used with the Align/Track/Flow policies are orthogonal to the boundaries, the trajectory will enter the adjacent cell in the domain of Φ_T, with one caveat. The orientation will be in alignment, but the speed along the curve may be too fast, and require the Align policy to be activated. In order to prevent this, we want the exit speed from one cell to be within the domain of the Track control policy of the next cell.

Beginning at the goal cell, we may impose the restriction on (27) in the adjacent cell be less than (27) in the current cell at the shared boundary. This guarantees the desired behavior, but is overly conservative as the sharpest turn dominates the speed profile. We desire to find a more relaxed, yet computationally tractable, method of finding $s(q)$ such that the constraints are satisfied and speeds are consistent across cell boundaries.

4.5 Goal Control Policy

The Flow control policy constructed above is useful for transferring a system from some configuration to the cell containing the goal, however, a different type of policy is needed to stabilize the system at the goal. The policy constructed in [59] is appropriate, and can be deployed along side our Flow policies. However, we make an extension that aligns the vector fields at the boundaries of the Goal and Flow cells.

To construct our component control policy for the goal cell, we follow a procedure similar to that used to construct the component control policies described above. We generate a potential field with uniform potential along the boundary of the goal cell, and zero potential at the goal, and generate a configuration based velocity reference as before.

We first map the goal cell to the unit ball using the map φ as before, and let

$$q_b^g = \varphi(q_g) \,,$$

where q_g is the goal configuration, and q_b^g is the mapped goal point. We then define a diffeomorphism $\psi : \mathcal{B} \to \mathcal{B}$ such that $\psi(\partial\mathcal{B}) = \partial\mathcal{B}$ and $\psi(q_b^g) = 0$. Define the potential function $\gamma_g : \mathcal{P} \to \mathbb{R}$ such that

$$\gamma_g = \frac{1}{2} ||\psi \circ \varphi||^2 \,.$$

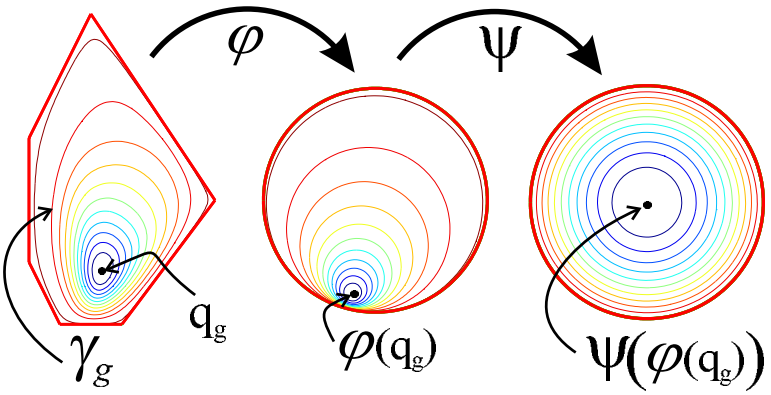


Figure 16: Equipotential contours for potential field found by mapping from goal cell to ball to ball centered at goal, and letting $\gamma_g = \frac{1}{2} \|\psi \circ \varphi\|^2$.

Figure 16 shows an example of this mapping for a polygonal cell in the plane.

We define the negative normalized gradient vector field, $\hat{X}_g$, as

$$\hat{X}_g(q) = -\frac{D_q \gamma_g^T}{\|D_q \gamma_g\|} .$$

We define the position dependent velocity vector field as

$$X_g(q) = s_g(q)\ \hat{X}_g ,$$

where $s_g(q)$ is given by

$$s_g(q) = \frac{\|q - q_g\|^2}{\|q - q_g\|^2 + \alpha} , \tag{34}$$

and α is a scalar parameter that regulates the rate of deceleration near the goal. Figure 17 shows how the α parameter affects the speed profile as the system approaches the goal.

We now wish to develop control policies to follow the vector field $X_g(q)$, while respecting the system constraints. We will leverage our prior results by letting $\hat{X} = X_g(q)$, and using the control laws developed above to follow the vector field.

For the kinematic system, we scale $X_g(q)$ as in (12) with $s(q) < \mathrm{V_{max}}$. The value of $\alpha > 0$ is only useful from a convergence time standpoint, as its impact on acceleration is irrelevant for kinematic systems. Smaller values of α result in faster convergence.

For dynamical systems, the velocity reference control policy of (14) is still valid; however, care must be taken in defining the scaling function for constrained systems. Consider the vector field derivative,

$$D_q X_g(q) = s_g(q)\ D_q \hat{X}_g + \hat{X}_g\, D_q s_g(q) , \tag{35}$$

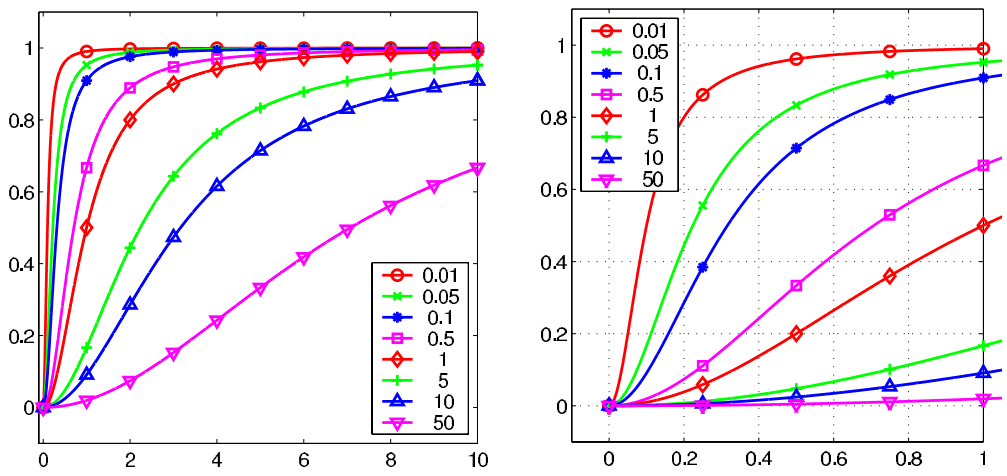


Figure 17: Curves of $s_g(q)$ vs. $||q - q_g||$ for various α. Figure on right is close up of origin.

At the goal point, the $D_q\hat{X}_g$ component has an infinite norm due to

$$\begin{aligned} D_q\hat{X}_g &= D_q\left(-\frac{D_q\gamma_g^T}{||D_q\gamma_g||}\right) \\ &= -\frac{D_q\left(D_q\gamma_g^T\right)}{||D_q\gamma_g||} + \frac{D_q\gamma_g^T D_q||D_q\gamma_g||}{||D_q\gamma_g||^2}, \end{aligned}$$

which goes to infinity as $\frac{1}{||D_q\gamma_g||}$. However, by multiplying by $s_g(q)$, which goes to zero at the goal point, we negate this impact. The last term in (35) involves a unit vector and a bounded gradient,

$$D_q s_g(q) = \frac{2\alpha\,(q-q_g)^T}{\left(||q-q_g||^2+\alpha\right)^2}, \tag{36}$$

so it results in a bounded vector field derivative.

Now consider the value of α. Assume the system is tracking an integral curve of $X(q) = s^*\,X_g(q)$ such that $\dot{q} = X(q)$, where s^* is a constant. Further assume that the vector field has begun to approach the goal point radially, so that $D_q\hat{X}_g\,\dot{q} = 0$. Under these conditions, the feed-forward acceleration is due only to the speed changes due to the change in the scaling factor,

$$\begin{aligned} \dot{X} = D_qX(q)\;\dot{q} &= s^*\,D_qX_g(q)\;s^*X_g(q) \\ &= (s^*)^2\,\hat{X}_g\,D_qs_g(q)\;s_g(q)\;\hat{X}_g \\ &= (s^*)^2\,s_g(q)\;\hat{X}_g\,D_qs_g(q)\;\hat{X}_g \\ &= (s^*)^2\,s_g(q)\;||D_qs_g(q)||\cos\xi\,\hat{X}_g, \end{aligned}$$

where ξ is the angle between $D_q s_g(q)$ and $\hat{X}_g$. For the radial approach, ξ is 0. We note that the norm of $\dot{X}$ is maximal when $s_g(q)\ ||D_q s_g(q)||$ is maximal. By calculation,

$$\begin{aligned} s_g(q)\ ||D_q s_g(q)|| &= \frac{||q-q_g||^2}{||q-q_g||^2+\alpha}\,\frac{2\alpha\ ||q-q_g||}{\left(||q-q_g||^2+\alpha\right)^2} \\ &= 2\alpha\,\frac{||q-q_g||^3}{\left(||q-q_g||^2+\alpha\right)^3}, \end{aligned}$$

therefore $s_g(q)\ ||D_q s_g(q)||$ is maximal when $||q-q_g|| = \sqrt{\alpha}$. We now have the constraint that

$$\begin{aligned} \left\|\dot{X}\right\| = (s^*)^2\ s_g(q)\ ||D_q s_g(q)|| &\leq \mathrm{A_{max}} \\ 2\alpha\ (s^*)^2\ \frac{||q-q_g||^3}{\left(||q-q_g||^2+\alpha\right)^3} &\leq \mathrm{A_{max}} \\ 2\alpha\ (s^*)^2\ \frac{\alpha^{3/2}}{(2\alpha)^3} &\leq \mathrm{A_{max}} \\ (s^*)^2\ \frac{\alpha^{3/2}}{4\alpha^2} &\leq \mathrm{A_{max}} \\ \frac{(s^*)^2}{4\sqrt{\alpha}} &\leq \mathrm{A_{max}}, \end{aligned}$$

which implies

$$\alpha \geq \frac{(s^*)^4}{16{\mathrm{A_{max}}}^2}. \tag{37}$$

Defining α as in (37) allows $X_g(q)$ to decelerate the system to the goal point in a way that respects the acceleration constraint on final approach, assuming the approach is radial. However, this does not take into account curvature in the vector field. For this we let $\hat{X} = X_g(q)$, and use the speed scaling factor as in (26) to ensure that the acceleration constraint is not violated where the vector field is changing. The α parameter takes most of the burden during final approach, while (26) guarantees that the overall control policy does not violate the constraints in the goal cell.

Given the vector field $X_g(q)$, the hybrid control polcies, Align and Track, can be used to bring the system to rest at the goal point. Thus, we define the Goal control policy $\Phi_G = \bigvee\{\Phi_T, \Phi_A\}$ using the vector field $X_g(q)$.

5 Specific Example

In this section we make our control policies concrete, given a specific type of decomposition. We begin by describing the use of *convex polytopes* to specify our cells. We construct a mapping that takes the cells to the unit ball such that the mapping of the interior of the cell is C^∞ almost everywhere. We then develop the specifics of the Save control policy, which is used in the hybrid control policies that form the switched component control policy for each cell in the constrained dynamical system. We conclude with simulations of navigation problems for planar systems.

5.1 Cellular Decomposition: Convex Polytopes

The use of convex polytopes is one of the most basic types of cellular decompositions. In lower dimensions, the polytopes are the familiar *polygons* in $\mathbb{R}^2$and *polyhedra* in $\mathbb{R}^3$. The cells can be easily described as the intersection of a set of half space constraints, which make the boundaries easy to specify, and involves trivial calculations to check a point for inclusion. It takes a minimum of $n+1$ half space constraints to bound n-dimensional space.

The decomposition of arbitrary space into convex polytopes is dependent on resolution, as curves are approximated by linear segments, and surfaces by planes. This implies that our methods are only resolution complete for arbitrary environments. For environments with obstacles defined by polytopes, this method can be exact. The component control policies are designed to take the configuration through a designated *outlet face* into the adjoining polytope.

We represent each half space constraint with a point, $p \in \mathbb{R}^n$, and a unit normal, $n \in \mathbb{R}^n$. The normal direction is assumed to be inward pointing with respect to the polytope being defined. Note, that the normal direction changes as the system goes from cell to cell across a common face. We choose p such that p is the center of the polytope face being specified by the half space. Note, this is an over parameterization of the half-space constraint. All that is required is the distance from the origin, and not a point on the hyper plane. We choose to carry p_i for use in later transformations.

Let $\beta_i(q) = n_i \cdot (q - p_i)$, the distance from a point q to the hyperplane defining the i^{th} half space constraint. If q is an interior point of the cell, then $\beta_i(q) > 0$ for all $i \in 1 \ldots m$. This allows us to compactly specify a convex polytope $\mathcal{P}$ as

$$\mathcal{P} = \{q \in \mathbb{R}^n \mid \forall i = 1 \ldots m, \beta_i(q) > 0\} .$$

If $\beta_i(q) = 0$ for some, but not all, of the half space constraints, then q is on the boundary of the cell. A necessary condition for the set $\{(p_i, n_i) \mid i = 1, \ldots, m\}$ of half-space constraints to specify a valid polytope is that the face normals n_i positively span the free space. Another necessary condition is that the center point of each face, p_i, is contained in the intersection of the other half space constraints. In other words,

$$\forall i = 1 \ldots m, \forall j = 1 \ldots m, i \neq j, \beta_j(p_i) > 0 .$$

Define q_β such that

$$q_\beta = \arg\max_q \prod_{i=1}^{m} \beta_i(q) ,$$

and let

$$\beta_{\max} = \prod_{i=1}^{m} \beta_i(q_\beta) ,$$

where $\beta_{\max}$ is the maximum value of the product of distances to each face on the interior of the cell. We define the scaled distance product function $\beta(q)$ as

$$\beta(q) = \beta_{\max}^{\frac{1-m}{m}} \prod_{i=1}^{m} \beta_i(q) \ . \tag{38}$$

Lemma 5.1 *The set of local maxima of $\beta(q)$ on the interior of $\mathcal{P}$ is a singleton. Furthermore, $\beta(q)$ is free of local minima on the interior of $\mathcal{P}$.*

Proof: By construction, $\beta(q)$ is positive over the interior of $\mathcal{P}$ and zero along the boundary of $\mathcal{P}$. Therefore $\beta(q)$ has at least on point corresponding to a global maximum on the interior of $\mathcal{P}$. Let such a point be given, which we denote q_β. Without loss of generality, transform the polytope such that q_β is at the origin.

Assume there exists another local maxima on the interior, and denote such point as q_m.

Define the line segment between the origin and q_m as $\mathbf{l}(t) = q_m t$, and note that β restricted to the line segment is given by

$$\begin{aligned}
\beta(l(t)) &= \beta_{\max}^{\frac{1-m}{m}} \prod_{i=1}^{m} n_i \cdot (\mathbf{l}(t) - p_i) \\
&= \beta_{\max}^{\frac{1-m}{m}} \prod_{i=1}^{m} \left(n_i^T q_m t - n_i^T p_i\right) \\
&= \beta_{\max}^{\frac{1-m}{m}} \prod_{i=1}^{m} \left(a_i t + d_i^0\right)
\end{aligned} \tag{39}$$

where $a_i = n_i^T q_m$, $d_i^0 = -n_i^T p_i$, and $t \in [0, 1]$.

The derivative along the line segment parameterized by t is given by

$$D_t\beta(t) = \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(a_i \prod_{\substack{j=1 \\ j\neq i}}^{m} \left(a_j t + d_j^0\right) \right) , \tag{40}$$

while the second derivative is given by

$$\begin{aligned}
D_{tt}\beta(t) &= \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(a_i \sum_{\substack{j=1 \\ j\neq i}}^{m} \left(a_j \prod_{\substack{k=1 \\ k\neq i \\ k\neq j}}^{m} \left(a_k t + d_k^0\right) \right) \right) \\
&= \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(a_i \sum_{\substack{j=1 \\ j\neq i}}^{m} \left(a_j \frac{1}{a_i t + d_i^0} \prod_{\substack{k=1 \\ k\neq j}}^{m} \left(a_k t + d_k^0\right) \right) \right) \\
&= \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(\frac{a_i}{a_i t + d_i^0} \sum_{\substack{j=1 \\ j\neq i}}^{m} \left(a_j \prod_{\substack{k=1 \\ k\neq j}}^{m} \left(a_k t + d_k^0\right) \right) \right) \\
&= \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(\frac{a_i}{a_i t + d_i^0} A_i \right) ,
\end{aligned} \tag{41}$$

where

$$A_i = \sum_{\substack{j=1 \\ j \neq i}}^{m} \left(a_j \prod_{\substack{k=1 \\ k \neq j}}^{m} \left(a_k t + d_k^0 \right) \right) .$$

Because we assumed that $q_\beta$and q_m are local maximum points, then $\beta(t)$ must be constant or there must be a local minima in $t \in (0,1)$. In either case, $D_t\beta(t^*) = 0$ for some $t^* \in (0,1)$. Note, that

$$D_t\beta(t) \mid_{t^*} = A_i + a_i \prod_{\substack{k=1 \\ k \neq i}}^{m} \left(a_k t^* + d_k^0 \right) ,$$

which implies that

$$A_i = -a_i \prod_{\substack{k=1 \\ k \neq i}}^{m} \left(a_k t^* + d_k^0 \right) . \tag{42}$$

Substituting (42) into (41), we obtain

$$D_{tt}\beta(t^*) = -\beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(\frac{a_i^2}{a_i t + d_i^0} \prod_{\substack{k=1 \\ k \neq i}}^{m} \left(a_k t^* + d_k^0 \right) \right) .$$

Since each term in the summation is positive, we have $D_{tt}\beta(t^*) < 0$, which implies $\mathbf{1}(t^*)$ is a local maximum. This contradicts our assumption that β is constant or has a local minimum along the line segment. Therefore, q_m must equal q_β, and we have a single local maximum on the interior of $\mathcal{P}$. Since β is positive over the interior of $\mathcal{P}$, and there is only one local maximum on the interior, there cannot be a local minimum in the interior.

□

From Lemma 5.1, we conclude that $\beta(q)$ monotonically decreases as q approaches the boundary of $\mathcal{P}$ along a ray from q_β in all directions.

5.1.1 Mapping to Unit Ball

Given the specification of a valid convex polytope, we construct a mapping to the unit ball using the scaled distance product, $\beta(q)$. First, note that the desirable properties of the constructed vector fields are invariant under rigid body transformation. Let T be the rigid body motion that transforms the cell so that the point q_β is at the origin, and the center point, p, of the designated outlet face lies on the negative x_1 axis. Such an operator maps face points to face points, and face normals to face normals. Unless otherwise noted, we will henceforth assume that the cell is transformed such that $p_i = T(p_i)$, $n_i = T(n_i)$, and $q = T(q) \in T(\mathcal{P})$.

Given a convex polytope $\mathcal{P}$ and transformation T defined above, $\varphi : T(\mathcal{P}) \to \mathcal{B}$ is defined as

$$\varphi(q) = \frac{q}{\|q\| + \beta(q)} , \tag{43}$$

where $\beta(q) : T(\mathcal{P}) \to \mathbb{R}$ was defined above. The $\beta(q)$ term is scaled such that the mapping is scale invariant. The mapping in (43) maps the origin to the origin, and any point on the boundary to the $(n-1)$-sphere.

The Jacobian matrix, $D_q\varphi$, is given by $D_q\varphi = \left[D_{x_j}\varphi_i\right]$, where x_j is the j^{th} component of q, and φ_i is the i^{th} component of $\varphi(q)$. Each element of the Jacobian is

$$\begin{aligned} D_{x_j}\varphi_i &= \frac{\delta_{i,j}}{||q|| + \beta(q)} - \frac{\varphi_i}{(||q|| + \beta(q))^2}\left(D_{x_j}||q|| + D_{x_j}\beta(q)\right) \\ &= \frac{\delta_{i,j}}{||q|| + \beta(q)} - \frac{\varphi_i}{(||q|| + \beta(q))^2}\left(\frac{\varphi_j}{||q||} + D_{x_j}\beta(q)\right). \end{aligned} \tag{44}$$

where

$$\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}.$$

Note, when $||q|| = 0$, $\varphi_i = \varphi_j = 0$, and

$$\lim_{||q|| \to 0} D_{x_j}\varphi_i = \frac{\delta_{i,j}}{\beta_{\max}^{\frac{1}{m}}}.$$

The distance function partial derivative is

$$D_{x_j}\beta(q) = \beta_{\max}^{\frac{1-m}{m}} \sum_{i=1}^{m} \left(n_{i,j} \prod_{\substack{k=1 \\ k \neq i}}^{m} \beta_k(q) \right), \tag{45}$$

where $n_{i,j}$ is the j^{th} component of the i^{th} normal vector. We claim that the Jacobian is full rank everywhere on the interior, so that the mapping φ is full rank on the interior as required from Section 4.1 [5]. We sketch the proof for arbitrary dimensions, and give a detailed proof for 2D.

Lemma 5.2 *The mapping φ is full rank on the interior of the cell $\mathcal{P}$.*

Proof: (Sketch) The mapping φ is designed to preserve angles relative to the origin, and scale the radial component. Consider the spherical coordinate representation of the mapping. The Jacobian has diagonal 1's for the angle coordinates. In order for the mapping to be singular, the partial derivative of the radial scaling with respect to the radius must be zero.

Let q_I be the point of intersection with the boundary, along the ray from the origin through the arbitrary point q in the interior of $\mathcal{P}$. This relationship is shown in Figure 18. The radial scaling is given by

$$r(t) = \frac{\rho t}{\rho t + \beta(q_I t)},$$

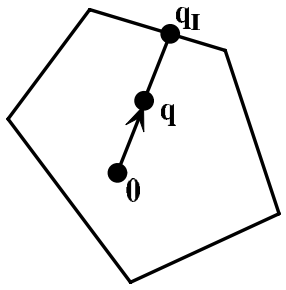


Figure 18: The point q and origin determine a point of intersection q_I in the linear retraction mapping.

where $\rho = ||q_I||$, and $t \in [0,1]$. Then,

$$\begin{aligned} D_t r(t) &= \frac{\rho}{(\rho t+\beta)} - \frac{\rho t(\rho + D_t\beta)}{(\rho t+\beta)^2} \\ &= \frac{\rho(\rho t+\beta)}{(\rho t+\beta)^2} - \frac{\rho t(\rho + D_t\beta)}{(\rho t+\beta)^2} \\ &= \frac{\rho(\beta - t D_t\beta)}{(\rho t+\beta)^2}, \end{aligned}$$

which only equals zero when

$$\beta - t D_t\beta = 0.$$

However, by Lemma 5.1, β is monotonically decreasing as we move along the ray parameterized by t, so that $D_t\beta < 0$. Since $t \geq 0$ and $\beta > 0$ over the interior of $\mathcal{P}$,

$$\beta - t D_t\beta > 0$$

for all $q \in \mathcal{P}$. We conclude that the Jacobian in spherical coordinates is full rank.

The mapping from Cartesian to spherical coordinates is full rank, except at the origin, where a proper limit exists. Therefore, we conclude the mapping φ is full rank over the interior of $\mathcal{P}$.

□

The mapping fails to be C^∞ at the origin due to the use of the radial retraction. However, this isolated discontinuity can be accomodated in the vector field calculations as it only impacts the calculation of the feed forward term in the control policies. Another option is to make use of blending C^∞ blending functions in a neighborhood of the origin. This is not necessary, as the system recovers from any minor perturbation on the interior due to the discontinuity.

5.1.2 Mappings to Unit Disk

The simulations presented in this report are for systems evolving on $\mathbb{R}^2$. For these systems, we decompose the free space into convex polygons. There are a number of algorithms for decomposing spaces into convex polygons. For some of the demonstration in this paper, we assume the free space is represented as a general polygon and use an algorithm from Keil [30]. This allows us to demonstrate automated deployment of our control policies. In general, our work assumes the decomposition is given.

From (43), with the scaling factor included in $\beta(q)$, we have

$$\varphi(q) = \left(\frac{x}{\sqrt{x^2+y^2}+\beta(q)}, \frac{y}{\sqrt{x^2+y^2}+\beta(q)} \right). \tag{46}$$

The Jacobian, $D_q\varphi$, is

$$D_q\varphi = \frac{1}{\left(\sqrt{x^2+y^2}+\beta(q)\right)^2} \begin{bmatrix} \frac{y^2}{\sqrt{x^2+y^2}} + \beta(q) - xD_x\beta(q) & -\frac{xy}{\sqrt{x^2+y^2}} - xD_y\beta(q) \\ -\frac{xy}{\sqrt{x^2+y^2}} - yD_x\beta(q) & \frac{x^2}{\sqrt{x^2+y^2}} + \beta(q) - yD_y\beta(q) \end{bmatrix} \tag{47}$$

Lemma 5.3 *The mapping φ given in (46) from an arbitrary polygon to the unit disk is full rank everywhere on the interior of the polygon.*

Proof: If $q = 0$, then a simple limit operation yields

$$D_q\varphi\,|_{q=0} = \begin{bmatrix} \frac{1}{\frac{1}{\beta_{\max}^m}} & 0 \\ 0 & \frac{1}{\frac{1}{\beta_{\max}^m}} \end{bmatrix},$$

which is full rank. Assume, $||q|| \neq 0$, and consider the determinant of $D_q\varphi$,

$$\mathrm{Det}\,(D_q\varphi) = -\frac{\left(x^2+y^2+\sqrt{x^2+y^2}\,\beta\,(q)\right)\left(-\beta\,(q)+x\,D_x\beta\,(q)+y\,D_y\beta\,(q)\right)}{\sqrt{x^2+y^2}\left(\sqrt{x^2+y^2}+\beta\,(q)\right)^4}. \tag{48}$$

The denominator and the first term in parenthesis in the numerator are positive, non-zero numbers for $||q|| > 0$. Therefore, to lose rank, the second term, $-\beta\,(q) + x\,D_x\beta\,(q) + y\,D_y\beta\,(q)$ must be zero. Assume this is true for some $q = (x, y) \in \mathcal{P}$. Then,

$$\beta\,(q) = D_q\beta\,(q)\;q. \tag{49}$$

However, by Lemma 5.1, $\beta\,(q)$ is monotonically decreasing as we move along the ray through q originating at the origin. Therefore, $D_q\beta\,(q)\;q < 0$ for all $q \in \mathcal{P}$, while $\beta\,(q) > 0$, contradicting (49). We conclude that the determinant is non-zero.

□

The mapping is singular on the boundaries, and in fact the Jacobian is the zero matrix at a vertex of the polygon. This necessitates an approximation of the cell near the polygon vertices, since the gradient vector disappears. We discuss approximation techniques in Appendix B.

Given the mapping φ from above, and the solution, γ_b, to Laplace's equation for the disk given in (8) or (10), we define the potential over the polygon as

$$\gamma = \gamma_b \circ \varphi.$$

This mapping is depicted in Figure 19.

By construction, the gradient vector field induced by the pullback of the heat equation solution is orthogonal to the boundary of the cell almost everywhere. This is trivial to show, given that the potential along the boundary of the cell is constant by virtue of the pull-back. At the vertices, where the gradient disappears, it is natural to define $\hat{X}$ as the norm of the average of the adjacent face normals. In Appendix B, we present a method for approximating the vertex region using a C^2 continuous curve.

To calculate the gradient, we must pick a solution to Laplace's equation, and differentiate. Choosing (10) to avoid the bookkeeping, we have

$$\begin{aligned}\gamma_b\,(\rho,\theta) = \frac{\alpha_1-\alpha_0}{2\pi} \;&+\; \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin\,(\alpha_1-\theta)}{1-\rho\cos\,(\alpha_1-\theta)}\right) \\ &-\; \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin\,(\alpha_0-\theta)}{1-\rho\cos\,(\alpha_0-\theta)}\right).\end{aligned} \tag{50}$$

With $\rho = \sqrt{x_d^2+y_d^2}$ and $\theta = \mathrm{atan2}\,(y_d, x_d)$, where $(x_d, y_d) = \varphi\,(x, y)$, we have

$$D_{q_b}\gamma_b = \begin{bmatrix} \frac{-\sin(\alpha)+\sin(\beta)+\rho\,(2\cos(\theta)\sin(\alpha-\beta)+\rho\,(-\sin(\alpha-2\,\theta)+\sin(\beta-2\,\theta)))}{\pi\,(1+\rho^2-2\,\rho\,\cos(\alpha-\theta))\,(1+\rho^2-2\,\rho\,\cos(\beta-\theta))} \\ \\ \frac{\cos(\alpha)-\cos(\beta)+\rho\,(-(\rho\,\cos(\alpha-2\,\theta))+\rho\,\cos(\beta-2\,\theta)+2\,\sin(\alpha-\beta)\,\sin(\theta))}{\pi\,(1+\rho^2-2\,\rho\,\cos(\alpha-\theta))\,(1+\rho^2-2\,\rho\,\cos(\beta-\theta))} \end{bmatrix}^T. \tag{51}$$

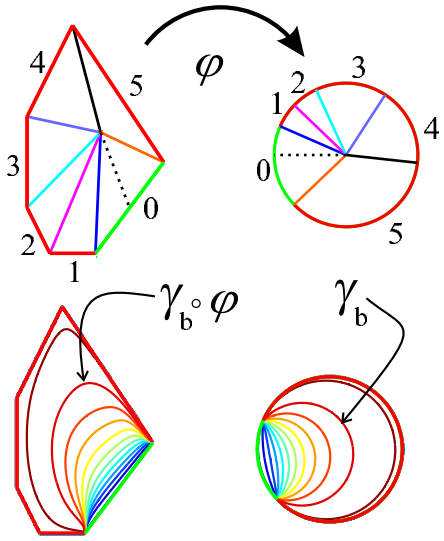


Figure 19: Mapping from polygon to unit disk. The contour plot on the left shows level sets of the pullback $\gamma_b \circ \varphi$ on the polygon; the contour plot on the right shows the corresponding level sets of γ_b on the unit disk.

The gradient in of the potential in the polygon is $D_q\gamma = D_{q_b}\gamma_b\, D_q\varphi$, which is all we need to calculate the negative normalized gradient vector field $\hat{X}$.

It is worth noting, that the mapping φ has a direct impact on the potential field, and therefore the gradient vector field. Figure 19 showed the results for the mapping given in this report. There are many other alternative mappings; consider the following three.

For the planar system, a *Schwarz-Christoffel* conformal map is a natural choice [22]. Unfortunately, the conformal map is in the opposite direction, that is from the disk to the polygon,

$$q = \varphi_{SC}^{-1}(q_b) = A + C\int^{z}\prod_{k=1}^{n}\left(1-\frac{\zeta}{z_k}\right)^{\alpha_k-1} d\zeta\,, \tag{52}$$

where A and C are complex constants, $z_k = \varphi(w_k)$ is the *prevertex* corresponding to the k^{th} vertex w_k of the polygon, and $\alpha_k\pi$ is the interior angle of the polygon. Note, z_k, w_k, and $q_b = \varphi_{SC}(q)$ are represented as complex numbers. Driscoll and Trefethen [22] present numerical algorithms for performing the calculations in Matlab [2]

As a second alternative, consider a variation on the mapping presented in this report. We term this the *Beta Root* mapping,

$$\varphi_{BR}(q) = \frac{q}{\sqrt[m]{\|q\|^m + \beta_1(q)}}\,, \tag{53}$$

where

$$\beta_1(q) = \prod_{i=1}^{m}\beta_i(q)$$

is the unscaled distance product. Like our mapping φ, φ_{BR} is scale invariant.

[2] MatLab is a trademark of the Mathworks, Inc.

For the final alternative, consider the simple *linear retraction*

$$\varphi_{LR}(q) = \frac{q}{||q_I||}, \tag{54}$$

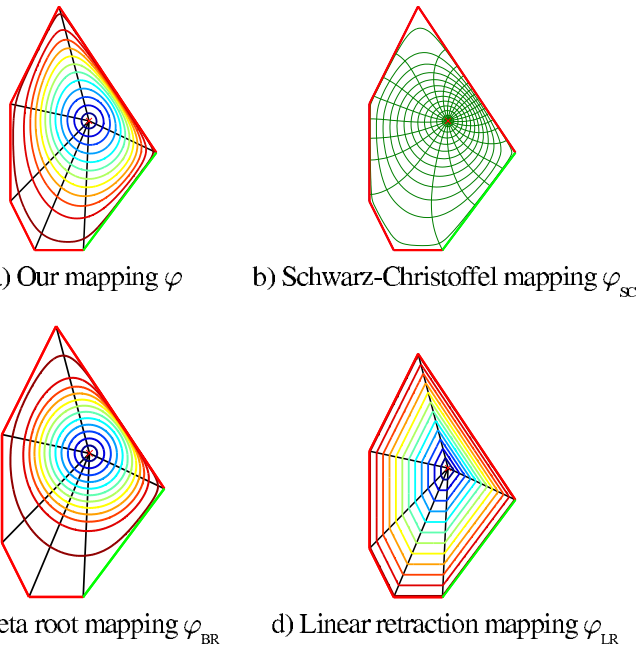
where q_I is the boundary intersection point found by traveling from the origin along the vector q, as shown in Figure 18. Note that the map origin may be chosen at any point in the interior of the polygon for this mapping, although this has a direct impact on the distortion. For the comparisons in this section, q_β was chosen as the common map origin.

Now, compare the mappings to one another. First consider the way in which the contours in the polygon corresponding to circles of constant radius inside the unit disk are distorted by the mapping. Figure 20 shows the comparison for the four mappings.

We note that except for the linear retraction, the mappings have a gradual transformation from a circle to the polygonal shape. In fact, the mappings involving $\beta(q)$ resemble the results from the Schwarz-Christoffel mapping with regards to the circular distortion. However, the Schwarz-Christoffel conformal map preserves angles between vectors when mapping from the disk and the polygon, which results in the bending of radial lines. The other mappings are scalings where the angles between points are preserved.

Because of the distortion patterns of the mappings, the equipotential contours of the pulled back potential field is likewise distorted. This causes different trajectories to occur when following the negative gradient. Figure 21 shows the resulting integral curves obtained by following the negative gradient for trajectories originating on the cell boundary. Note, the Schwarz-Christoffel mapping required the use of numerical approximations to calculate the Jacobian.

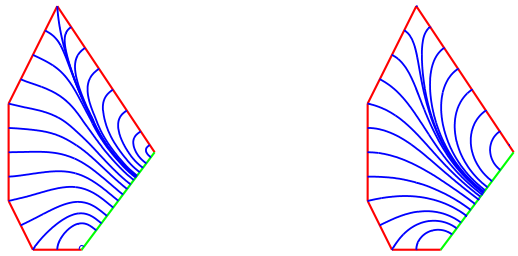
The resulting integral curves using φ, φ_{BR}, and φ_{SC} are very similar, with some subtle differences. Because the conformal mapping does not generalize beyond the planar systems, and given the lack of a closed form



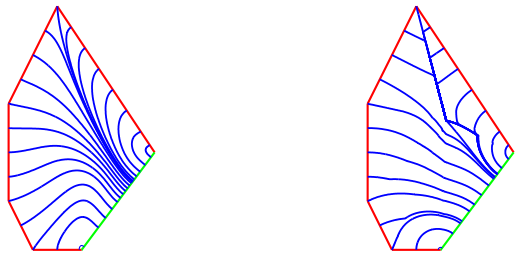
a) Our mapping φ b) Schwarz-Christoffel mapping φ_{SC}

c) Beta root mapping φ_{BR} d) Linear retraction mapping φ_{LR}

Figure 20: Comparison of distortion due to mappings of polygon to disk. Contours correspond to circles of constant radius on the disk.

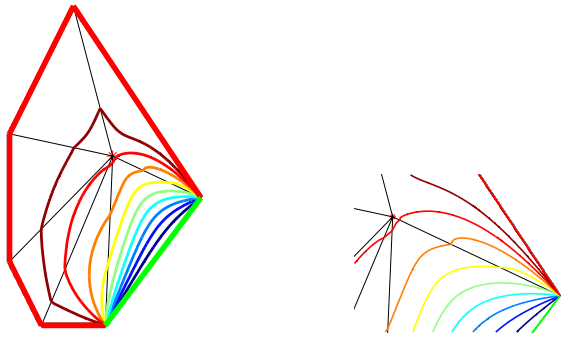


a) Our mapping φ b) Schwarz-Christoffel mapping φ_{SC}



c) Beta root mapping φ_{BR} d) Linear retraction mapping φ_{LR}

Figure 21: Integral curves of the negative gradient vector field for various mappings.



a) Equipotential contours for φ_{LR} b) Close up view

Figure 22: Distortion of the potential field due to linear retraction mapping. Notice the inflection points in the contours along the line connecting the map center to the vertices.

mapping and dependence on numerical solutions, the Schwarz-Christoffel mapping is not a good choice for our work. However, it is reassuring that the chosen mapping gives qualitatively similar results to the conformal mapping. On the other hand, the integral curves for the linear retraction are problematic. Although the mapping is continuous, the distortion of the potential field leads to unintuitive gradients. Figure 22 shows the distortion of the equipotential contours.

Figures 20 and 21 point to both the problem and promise of our approach; that is the dependence of the the path on the chosen mapping. We have presented a valid mapping; however, there are other equally valid

mappings that might be less costly for some metric. Our approach has been to define a generic mapping and control law applicable to any cell in the decomposition. However, any mapping that meets the criteria given in Section 4.1 would work with our generic control law. In this way, the mapping used could depend on some structure of the cell or other criteria, and the mapping could change as we move from cell to cell in the decomposition.

5.1.3 Save Control Policy

The design of the Save control policy, introduced in Section 4.4.1, is dependent upon the form of the cells used in the decomposition. For convex polytopes, the Save control policy given in [59] has maximal domain. This section presents the Save policy, and develops an expression for the savable set defining the domain of the policy. We begin by presenting the policy in its basic form, and then discuss the switched dynamics that the policy induces. We next develop an expression for the collision ratio, ζ_c, that takes into consideration the switched dynamics.

The Save control policy, Φ_S, is used to apply all acceleration normal to the boundary at the projected collision point, in order to slow the system trajectory and prevent collision if at all possible. The goal of the policy is to bring the system to rest within a given cell without violating the cell boundaries. Define q_c to be the point of intersection with the cell boundary along the direction of the current velocity, that is q_c is the collision point if no control input is applied (Figure 23). Let n_c, which we will term the *collision normal*, be the boundary normal at this collision point. For now, under the general position assumption, we assume the collision point is contained in one face of the polytope. That is the collision does not occur at the intersection of two or more faces of the polytope. We define the Save control policy, Φ_S, as

$$u = \mathrm{A}_{\max} n_c \,. \tag{55}$$

The effect of the Save control policy is to accelerate maximally away from the projected collision point, thereby decreasing the collision speed s_c.

The effect of Φ_S can be decomposed into a component along the current velocity, and a component orthogonal to the current velocity, as shown in Figure 24. In every case, the component along the current velocity acts to slow the system down, while the orthogonal component acts to steer the trajectory away from the closest boundary.

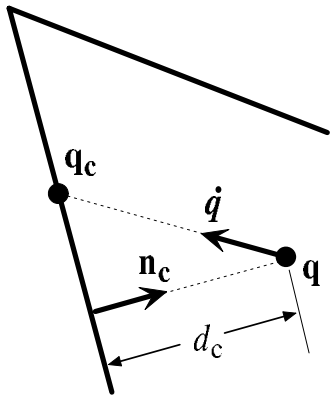


Figure 23: Collision projection based on current velocity.

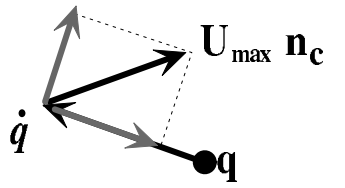


Figure 24: The acceleration components of the Save control policy always act to slow the overall speed and steer the trajectory away from the closest boundary.

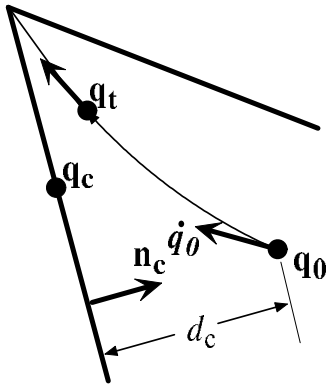


Figure 25: The action of the Save control policy Φ_S pushes the trajectory away from the point of imminent collision to a local maximum of the distance to collision. At the local maximum, the collision normal is aligned with the current velocity.

Note that the control policy always acts to maximally decrease the component of velocity towards the shortest collision distance. This pushes the trajectory away from the point of imminent collision, and locally increases the time to impact. The acceleration away from the point of first impact will continue until the velocity vector is oriented toward the intersection of two or more faces of the polytope, as shown in Figure 25. Thus, as the system is accelerating away from the point of immenient collision, it is accelerating towards another face, until the system velocity is oriented toward the intersection of two polytope faces. In this case, accelerating in the direction of either face's surface normal would decrease the time to impact of at least one of the faces. This introduces a discrete change in the required acceleration direction.

We redefine the collision normal to be in the positive linear space of the normals of the faces intersecting at the collision point on the intersection. In the planar case, where the current velocity is directed toward a vertex, the collision normal is aligned along the negative direction of the current velocity, which will act to bring the system to rest. The collision normal is oriented so that it and the current velocity vector form a co-dimension $(n-m)$ hyper-plane normal to the surface formed by the intersection of the polytope faces, where n is the dimension of the $\mathcal{FS}$ and m is the number of faces intersecting at the collision point. Figure 26 shows an example of this for 3-dimensional configuration space. The acceleration is contained in this co-dimension $(n-m)$ hyper-plane is by definition normal to the surface formed by the intersection if the polytope faces. The acceleration pushes the trajectory along the intersection surface towards a "corner", formed by intersection with another face. The process continues until the intersection surface is a point, and the collision normal is oriented directly opposite the current velocity, which drives the system to rest.

Lemma 5.4 (Rizzi [59]) *The Save control policy, Φ_S, is capable of bringing to rest any condition in $\mathcal{TP}$ that can be brought to rest without violating the given constraints.*

Proof: Based on [59].

Assume Φ_S cannot prevent collision with the boundary for some initial position and velocity, and further assume the existance of another control policy Φ'_S that can prevent collision.

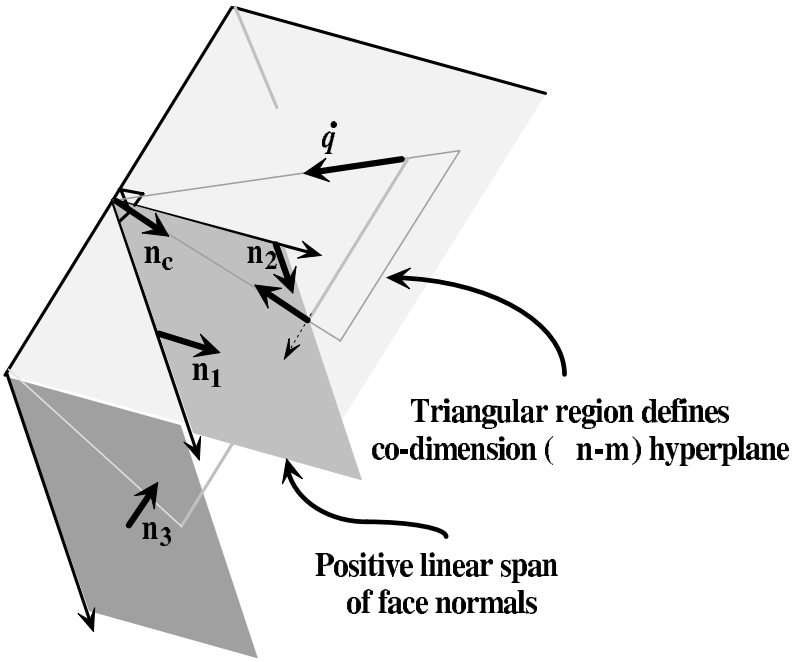


Figure 26: Collision with intersection of two faces with Save policy. Face 1 is transparent. With $m = 2$ and $n = 3$, the velocity is contained in a co-dimension 1 plane.

Any boundary violation under the influence of Φ_S involves collision with the "nearest" boundary component, that is the boundary component with the shortest time to impact. However, Φ_S acts to maximally increase the time to impact of the nearest boundary component. If $\Phi'_S \neq \Phi_S$, then Φ'_S must not act to maximally increase the shortest time to impact. But if Φ_S cannot prevent the collision, then neither can Φ'_S.

□

With this proof of correctness, we seek an expression for defining the savable set for convex polytopes. Given the collision normal, the distance to collision is given by

$$d_c = n_c^T (q - p_c) \ ,$$

where p_c is a point on the face, and d_c is the distance to the *collision* plane defined by the collision point and the collision normal. We define the collision speed as

$$s_c = -n_c^T \dot{q} \, ,$$

where s_c is the velocity component along the normal to the collision point, and represents how fast the system is approaching the boundary. The time to impact, t_c, is simply

$$t_c = \frac{d_c}{s_c} = -\frac{n_c^T (q - p_c)}{n_c^T \dot{q}} \, . \tag{56}$$

The time required to bring the collision speed to zero using maximum acceleration in the constant direction of the collision normal, is

$$t_b = \frac{s_c}{\mathrm{A}_{\max}}.$$

Krogh [40] defines the *reserve avoidance time*, $t_c - t_b$, and notes that collision can only be avoided if $t_c - t_b > 0$. The distance covered during the braking maneuver, given t_b, is

$$d_b = s_c\, t_b - \frac{1}{2}\mathrm{A}_{\max} t_b^2 = \frac{s_c^2}{2\mathrm{A}_{\max}}.$$

Using the definitions above, we define the collision avoidance ratio with the initial collision face, ζ_1, as

$$\zeta_1 = \frac{d_b}{d_c}.$$

If $\zeta_1 \geq 1$ then a collision cannot be avoided.

Now, consider the change in ζ_1 as time evolves. From the initial point, both the braking distance d_b and the collision distance d_c decrease by the distance traveled over some differential time period. We can write ζ_1 as a function of time as

$$\begin{aligned} \zeta_1(t) &= \frac{d_b - \int_0^t (s_c - \mathrm{A}_{\max}\tau)\, d\tau}{d_c - \int_0^t (s_c - \mathrm{A}_{\max}\tau)\, d\tau} \\ &= \frac{d_b - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2}{d_c - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2}. \end{aligned} \tag{57}$$

Here we assume that the cell is a convex polytope, with the collision normal constant over some finite range, hence the collision velocity is a one dimensional effect. Taking the time derivative of (57), we obtain

$$\begin{aligned} \dot{\zeta}_1(t) &= \frac{-s_c + \mathrm{A}_{\max} t}{d_c - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t} - \frac{d_b - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2}{\left(d_c - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2\right)^2}\left(-s_c + \mathrm{A}_{\max} t\right) \\ &= \frac{2\,(d_b - d_c)\,(s_c - \mathrm{A}_{\max} t)}{\left(d_c - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2\right)^2}. \end{aligned} \tag{58}$$

In the time period before the collision, $s_c - \mathrm{A}_{\max} t > 0$ and $d_c - s_c\, t + \frac{1}{2}\mathrm{A}_{\max} t^2 > 0$, therefore, the sign of $\dot{\zeta}_c$ depends on the relative values of d_b and d_c. If $d_c > d_b$, then the derivative of the collision ratio is negative, and the collision ratio never increases beyond the unity value signifying imminent collision. Intuitively, the remaining braking distance goes to zero before the collision distance, and $\zeta_1 \to 0$. On the other hand, if $d_b > d_c$, $\dot{\zeta}_1$ is positive, signifying no recovery. In this case, the collision distance goes to zero before the braking distance, and the collision ratio "blows up." While this proves that the system will not collide with the initial collision face, it fails to prove that there will not be a collision with any face on the polytope.

The discrete change in collision normal that occurs when the velocity is oriented towards the intersection of two polytope faces results in a change in the collision ratio calculation, as the acceleration is no longer orthogonal to either face. Therefore, although the Save policy may be able to avoid a first collision face in isolation, collision with the second face may be unavoidable. As the closed loop dynamics in response to the constant acceleration are easy to determine, we can determine the collision ratio when the system velocity is aligned with the intersection of two or more faces based on the new collision normal.

Let n_1 equal the original collision normal defined by projected collision with a single face, and let p_1 denote the associated face location. Using the Save policy defined in (55), the closed loop dynamics are

given by

$$q(t) = q(0) + \dot{q}(0)t + \frac{1}{2}\mathrm{A}_{\max} n_1 t^2 , \tag{59}$$

$$\dot{q}(t) = \dot{q}(0) + \mathrm{A}_{\max} n_1 t . \tag{60}$$

Repeating the analysis in (56), we may calculate the instantaneous time to collision for the closed loop system.

$$t_{c_1} = -\frac{n_1^T (q(t) - p_1)}{n_1^T \dot{q}(t)} .$$

Likewise, we may calculate the time to collision with the second face

$$t_{c_2} = -\frac{n_2^T (q(t) - p_2)}{n_2^T \dot{q}(t)} ,$$

where the second face is determined by checking the component of velocity orthogonal to the first face with respect to collision with other faces. Equating t_{c_1} and t_{c_2}, we can solve for the time at which the velocity is oriented toward the intersection of two faces, t_2. Let d_{c_2} denote the orthogonal distance to intersection of faces 1 and 2. With the collision normal n_c redefined as described above, we may define the secondary collision ratio as

$$\zeta_2 = \frac{\left(n_c^T \dot{q}(t_2)\right)^2}{2\mathrm{A}_{\max} d_{c_2}},.$$

For higher dimension systems, we continue these calculations beginning at t_2, and solving for the closed loop response given the new collision normal. The iterations continue until the intersections of additional faces results in a single vertex point. If at iteration i, the calculated value for ζ_i is greater than one, collision is inevitable and the iteration halts. Given the iterations, define the overall collision ratio ζ_c as

$$\zeta_c = \max_i \zeta_i .$$

In Section 4.4.1, we introduced the notion of the savable set, $\mathcal{S}$ which is the set of all $(q, \dot{q}) \in \mathcal{TP}$ such that Φ_S prevents collisions with the inlet zone. Given our definition of the overall collision ratio ζ_c, we may formally define $\mathcal{S}$ as

$$\mathcal{S} := \{(q, \dot{q}) \in \mathcal{TP} \mid \zeta_c \leq 1\} .$$

5.2 Simulation Results

The methods described in the previous sections have been validated in simulation for fully actuated systems in $\mathbb{R}^2$. In this section, we present these simulation results for kinematic and dynamical systems.

5.2.1 Kinematic System Simulation

For the kinematic system $\dot{q} = u$, and the control law $u = X(q)$ as described in Section 4.3.1, the deployment strategy is even more straightforward that than given in Section 3.4. Because the state space is just the configuration, the control policy deployment only needs to cover the configuration space, which is automatic for control policies defined over the cells in the disjoint cellular decomposition. For non-overlapping cells, the prepares relationship $\Phi_1 \succeq \Phi_2$ requires that cells $\mathcal{P}_1$ and $\mathcal{P}_2$ share a common portion of their boundary, and the velocity from cell $\mathcal{P}_1$ cause the system to enter cell $\mathcal{P}_2$.

The adjacency graph $\mathcal{G}$ of the disjoint cellular decomposition $\mathcal{K}$ provides the full information needed to deploy the control policies developed in Section 4.3.1. The control policies provide the means for navigating the adjacency graph, transforming a continuous problem into a discrete graph search. Once the configuration has traversed the cells and arrived at the cell containing the goal, the final attractive control policy is deployed, which brings the configuration to the goal.

Given a collection of control policies $\{\Phi_{ij}\}$ over each cell $\mathcal{P}_i$, such that Φ_{ij} drives the configuration $q \in \bar{\mathcal{P}}_i$ through the outlet zone given by the common boundary with the j^{th} adjacent cell in the decomposition, we construct a partial order of the control policies by converting the adjacency graph to a *spanning tree* representation. The spanning tree is constructed based on some metric used in the graph searching algorithm. Although this report does not address the development of such a metric, some measure of the average cost to traverse the cells of higher priority is appropriate. For an edge in the spanning tree connecting node i to node j, the control policy Φ_{ij} is chosen over cell i. A spanning tree is guaranteed to exist for the portion of the adjacency graph connected to the node corresponding to the cell $\mathcal{P}_g$ [71]. Figure 27 shows the spanning tree corresponding to the adjacency graph shown in Figure 7. For kinematic systems, the prepares relationship is guaranteed by the nature of the control policies described in this report.

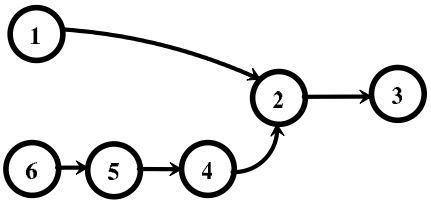


Figure 27: The adjacency graph is converted to a spanning tree representation, where the root node of the tree corresponds to the cell containing the goal point. The topology of the spanning tree, determined by some cost function, induces a partial order on the adjacency graph.

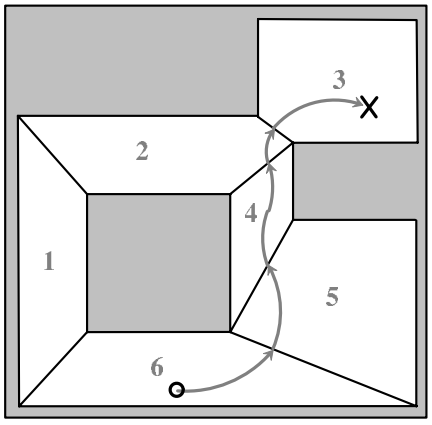


Figure 28: Given the cellular decomposition and goal point, a path to the goal can be determined by searching the spanning tree of the adjacency graph.

The topology of the spanning tree determines the partial ordering of the control policies over the cells, and we refer the the spanning tree representation as the *partial order* of the control policies for the kinematic system. Starting at a configuration within any cell, $\mathcal{P}_s \in \mathcal{K}$, the deployed component control policies induce a trajectory that determines a path through connected cells by traversing the spanning tree from the current node to its parent node until we reach the root node, which corresponds to the goal cell. Given the structure of the spanning tree, we are guaranteed to find a path to the goal cell for any cell in the connected sub-graph containing the goal vertex [71].

The existence of this connected path in the partial order implies the existence of a connected path in the free space by construction. The global control problem of how to get the system to the goal is reduced to that of getting the system to the next cell in the discrete graph path, and then finally getting from the interior of the goal cell $\mathcal{P}_g$ to the goal point $q_g \in \mathcal{P}_g$. In this way, the continuous control problem is abstracted to a discrete search problem. For an exact cellular decomposition, if a path in the spanning tree is not found, then a path in the free space does not exist [42]. An approximate cellular decomposition may require refinement of the approximation if a path is not found, but the search of the approximate cellular decomposition is *resolution complete* [42, 23].

The control policy deployment scheme, given component control policies that function as specified above, is (resolution) complete. In other words, the system can navigate a path from start to goal if and only if a path in the spanning tree corresponding to a chain of connected cells exists. Figure 28 shows a hypothetical trajectory under the influence of the component control policies specified by the partial order shown in Figure 27. When the configuration enters the cell containing the goal configuration, a simple converging control policy is used in place of the component control policy described above.

Figure 29 shows the results of this control policy deployment for a maze-like region of free space constructed in $\mathbb{R}^2$. The maze was built as a single polygon, and then decomposed into convex cells using the algorithm of Keil [30], which allowed the demonstration of automated deployment. Switching from policy to policy is automatic as the system configuration crosses the boundaries of the cells defined by the decomposition. The exact path taken by the system is dependent on the underlying decomposition.

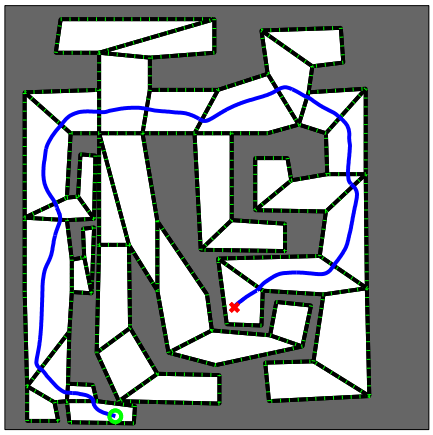


Figure 29: Simulation of kinematic system. The dark line shows the path taken, dark region denotes the boundary of the free space, and dotted lines show the decomposition into convex polygons.

5.2.2 Unconstrained Dynamics Simulation

For an unconstrained dynamical system $\ddot{q} = u$, the control law $u = K\ (X(q) - \dot{q}) + D_q X\, \dot{q}$ as described in Section 4.3.2 is used. For sufficiently high gain K, the system becomes essentially kinematic, and the deployment strategy presented in Section 5.2.1 is followed. The Align component control policy can be deployed to recover non-zero initial conditions, by specifying a large arbitrary acceleration limit.

Figure 30 shows a simulation of the dynamic system given in (13) under the influence of (14). A variety of initial conditions are shown, each converging to the goal configuration using the hybrid control strategy induced by the underlying decomposition. This demonstrates the global control policy that is induced by our policies and deployment method.

5.2.3 Constrained Dynamics Simulation

For an constrained dynamical system $\ddot{q} = u$, subject to $||\dot{q}|| \leq \mathrm{V}_{\max}$ and $||\ddot{q}|| \leq \mathrm{A}_{\max}$, the hybrid control policies developed in Sections 4.3.3 and 4.4 are deployed using the strategy presented in Section 3.4. Because the domain of the Save policy is limited by the cell, it will require multiple deployments of overlapping cells to fully fill the free state space.

The example shown in Figure 31 is only based on a single decomposition. The free workspace is decomposed into convex regions, and the policies are applied. The initial velocity points toward the upper right corner of the first cell. The Align policy is applied first, followed by the Track policy. The initial velocity was chosen to just miss the cell boundary. During execution of the Align policy, we differentiate between stopping ($\zeta_c > \mu$) and aligning ($\zeta_c < \mu$). During the full simulation, numerical errors (disturbances) cause the system to jump from the Track to the Align policies. The disturbances are handled, and the trajectory quickly returns to the influence of the Track policy, because the Align and Track policies are deployed to fill the saveable state space. The system eventually converges to the goal as desired, while avoiding the obstacles.

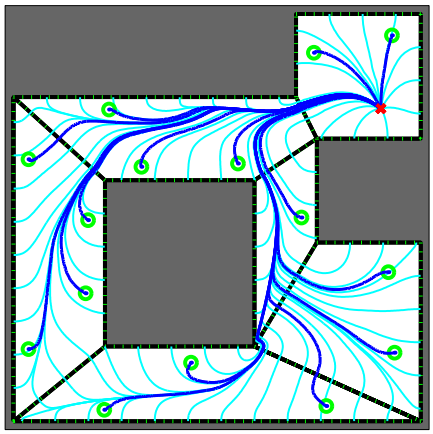
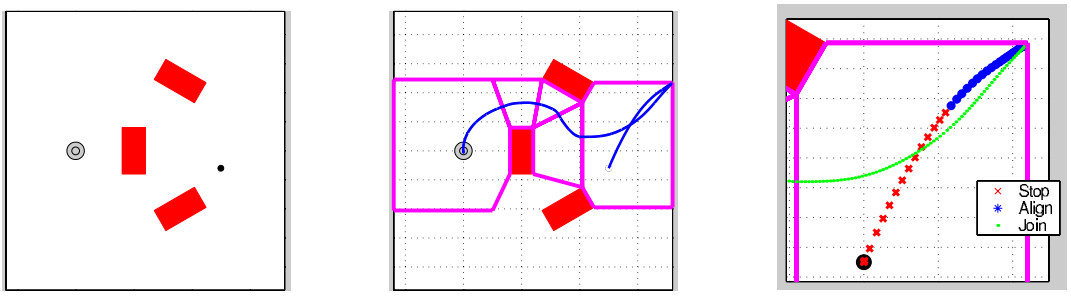


Figure 30: Simulation of the dynamical system using the hybrid control strategy introduced in this paper. Light colored lines represent integral curves of the $X(q)$, while the dark colored lines represent trajectories of the system for various initial conditions. The velocity regulation gain was $K = 20$ in this example.



a) Course layout b) Decomposition and path c) Close-up of switched policy behaviors. The initial velocity is to the upper right.

Figure 31: Simulation of a constrained dynamical system showing the result of hybrid switching policies with spatial decomposition. The Align policy is activated first, with stopping ($\zeta_c > \mu$) preceeding aligning. The Track policy the takes over and drives the system through the region by composing the control policies.

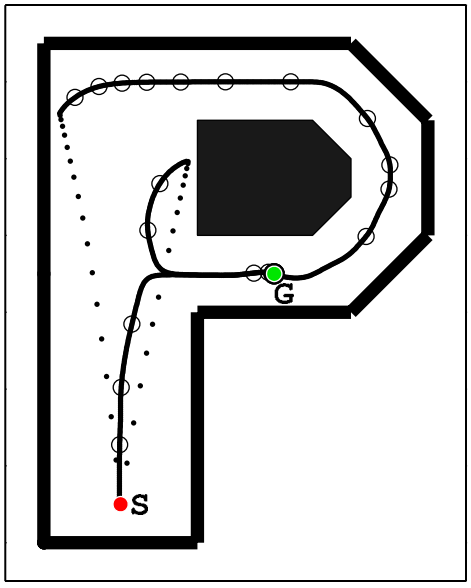


Figure 32: Dynamical "P" problem. The non-simply connected configuration space elicits different "choices" for the desired trajectory depending on the initial conditions. Behaviors for three different initial conditions are shown. The overall control policy is achieved with 16 local control policies deployed using our methods for extended sequential composition. The solid dots, which form the lines, are placed every 0.025 seconds; the circles are placed every 5 seconds of simulation time.

5.2.4 Dynamical "P" Problem

As a demonstration of the power and flexibility of our approach, we present simulation results of what we term the "dynamical P" problem, shown in Figure 32. The configuration space is not simply connected, and a decision about which way to travel around the loop to get to the goal must be made. One possible approach

for kinematic systems is to base the decision on path length, with the resulting decision surface defined by the set of points equidistant from the goal. Certainly other decision surfaces are possible as well. For a constrained dynamical system, specification of a comparable decision surface is complicated, based not only on configuration, but also on initial velocity. Instead of determining the decision surface in a optimal fashion, our method partitions the space into regions based on reasonably efficient local control policies.

The deployment of the policies is based on the geometry of the configuration space, and the constraints of the system. The system constraints determine the maximum extent of the domain in the free state space for a policy defined over a given region of configuration space. Thus the active policy changes based on the state of the system and the constraints of the system. As the active policies change, the induced behavior of the system changes.

The simulation assumes an idealized dynamical robot, $\ddot{q} = u$ with $q, u \in \mathbb{R}^2$, subject to both velocity and acceleration constraints. The system starts at a point near the bottom, and must navigate to a specified point in the lower portion of the "P" loop. The simulation was conducted for three separate initial velocities of varying magnitude and direction.

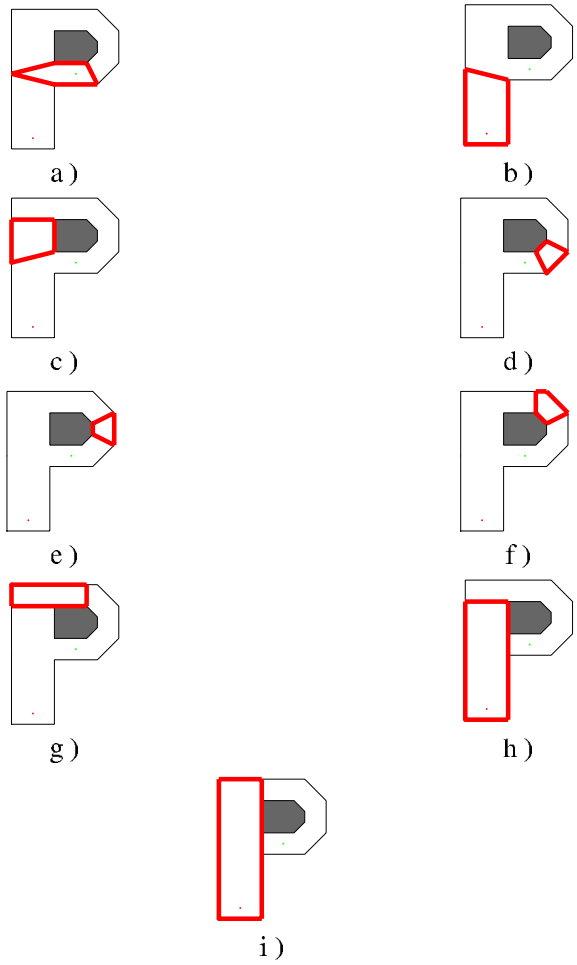


Figure 33: Configuration-cells used to define local control policies for dynamical P simulation.

For the configuration given in Figure 32, nine hybrid control policies were deployed. One Goal policy was used around the goal. Six Flow policies were deployed in polytopes that covered the remaining free configuration space. Two overlapping Save policies were deployed to capture adverse initial conditions. The ordering of the policies, $\mathcal{U}'$, was generated by hand based on the algorithm presented in Section 3.4. Let

$$\begin{aligned}\mathcal{U}' = \{ \quad & \Phi_{G_a}, \Phi_{F_b}, \Phi_{F_c}, \Phi_{F_d}, \Phi_{F_e}, \Phi_{F_f}, \\ & \Phi_{F_g}, \Phi_{S_h}, \Phi_{S_i} \} ,\end{aligned}$$

where the second subscript corresponds to the configuration-cells shown in Figure 33. Additional deployments and cells are possible, but this set allows us to demonstrate the required switching behavior in simulation. The Goal and Flow policies are each hybrid policies composed of Align and Track policies, so 16 local control policies were deployed in total.

The deployment of the Goal and Flow policies covers the free configuration space, and the Flow policies are configured to prepare the adjacent policy of higher priority. This induces a piecewise potential based navigation function for any state within the domains of the Goal or Flow policies. To capture large initial velocities, the two Save policies are deployed in the large corridor. The first Save policy covered the corridor from the bottom to the top of the obstacle forming the "P". Any state savable by this policy prepared the Flow or Goal policies in a way that caused the system to enter the lower half of the loop from the left. The second Save policy encompassed the entire large corridor, anything savable by this policy, but not by the first Save policy, entered the lower loop from the right by traveling around the obstacle.

Figure 32 shows three initial conditions, which demonstrate the change in behavior as a response to changes in the initial conditions. In the first case, which starts to the left, the activated policies are Φ_{S_i}, Φ_{F_g}, Φ_{F_f}, Φ_{F_e}, Φ_{F_d}, and Φ_{G_a}, in that order. The second case, which starts to the right activates Φ_{S_h}, Φ_{F_c}, and Φ_{G_a}, in that order. The final case, which starts near vertical, activates Φ_{F_b} and Φ_{G_a}. In all cases, the switching is automatic based on the state being within the domain of the highest priority control policy.

The "intelligent" decision making with regard to when to go around the obstacle is inherent in the deployment scheme. There is no replanning based on initial conditions. The change in behavior is induced by the switched local control policies and their respective domains. By allowing overlapping domains that prepare multiple policies, our extensions allow for an expressive set of policies to be deployed that cover a large region of the free state space.

6 Conclusion

The work presented in this report represents the initial steps in a program of research designed to bring about automatic methods of deploying robust low-level control policies for constrained systems. Our goal is to enable global behaviors through composition of low-level controls in a manner that guarantees performance. Our long term goal is to develop methods of encoding behavior design, and facilitating automated deployment of these component control policies to enable high-level goals to be accomplished, while leveraging the robustness and performance of low-level controls to accomplish the specified tasks.

Our work focuses on the development of control strategies based on a bottom-up design philosophy. The deployment of local control policies results in a task independent method of solving global problems based on the capabilities of a given system. Each local control policy results in a provably correct behavior for any state within its domain, with guaranteed safe switching between policies inducing global convergence for any state in the union of the domains of the deployed control policies. The policies are designed to be memoryless state feedback policies, making them suitable for deployment in reactive systems. However, by their design, the resulting stability properties guarantee safe behavior and predictable transitions.

Our extensions to the sequential composition method allow for the use of the broader class of control policies presented here, in addition to other policies. The ability to deploy policies defined over multiple overlapping configuration-cells increases our ability to deploy policies that cover a large portion of the free state space. The extended sequential composition method yields a constructive technique for generating a hybrid control automata with guaranteed transition properties, thereby enabling planning and analysis in the discrete space of behaviors. As our simulations demonstrate, this method yields a rich set of behaviors without the need for high-level planning, while guaranteeing the safety and global convergence.

7 Future Work

To enable the specification of truly globally convergent switched control policies, additional policies must be defined. The current method of basing policies on convex decompositions in free space, while effective, is not expressive enough to fully cover the free state space. Specifically, the configuration cell presents an artificial boundary with respect to what is savable. Policies that cannot save or align, but can direct the flow through a configuration-cell to a given face could be designed to address this problem. This would enable a cell that cannot save a given inlet velocity to pass the system to another downstream policy that can bring the system to rest. These flow-through policies would expand the available domains, and ease the restrictions on the velocity scaling $s(q)$. This, along with less conservative velocity scalings, could be used to expand the domains of the deployable policies, thereby allowing a greater portion of the free state space to be filled.

The next phase of our research will extend the control policy composition concepts to systems with nonholonomic constraints. Extensions to traditional wheeled mobile robots, as well as car-like vehicles, will be constructed. By developing policies that respect the differential constraints of the vehicle, the sequential composition method developed in this paper can be used to address the navigation and control problem for nonholonomic systems. In this way, the control problem is abstracted away from the task and environment, and towards the specific capabilities of a given system.

One of the main goals of this research is to allow global behaviors to be expressed through the composition of local control policies. By properly deploying low-level policies, higher-level behaviors can be induced. Our research will demonstrate this abstraction for parallel parking and k-turns for a car-like robotic system. The abstraction is accomplished by finding a collection of policies that can be parameterized as a group to solve the higher-level task. The ability to generate these types of behaviors is inherent in the methods used in this paper to solve the navigation problem.

The method presented in this paper assumes a global goal and full information. Ultimately, our work will address ways to incrementally deploy control policies to allow for changing goals or exploration. This

will enable a high-level reasoning system to change the deployment on the fly, resulting in more flexible and adaptable systems, while preserving the performance guarantees.

A PDE - Separation of Variables

Another method of solving Laplace's equation, known as *separation of variables*, is applicable if the problem has natural symmetries [60]. As the name suggests, the technique involves separating the influence of the variables and solving using a Fourier series. Consider the polar coordinate form of Laplace's equation for the unit disk (see Section 4.2.3) given by

$$\begin{aligned} \frac{\partial^2 u}{\partial \rho^2} + \frac{1}{\rho}\frac{\partial u}{\partial \rho} + \frac{1}{\rho^2}\frac{\partial^2 u}{\partial \theta^2} &= 0\,, \\ u(1,\theta) &= g(theta)\,. \end{aligned}$$

Assume $u(\rho,\theta) = X(\rho)\,Y(\theta)$, which implies

$$X''(\rho)\,Y(\theta) + \frac{1}{\rho}X'(\rho)\,Y(\theta) + \frac{1}{\rho^2}X(\rho)\,Y''(\theta) = 0\,. \tag{61}$$

Dividing by u, we obtain

$$\frac{X''(\rho)}{X(\rho)} + \frac{1}{\rho}\frac{X'(\rho)}{X(\rho)} + \frac{1}{\rho^2}\frac{Y''(\theta)}{Y(\theta)} = 0\,,$$

which can be rewritten

$$\frac{\rho^2 X''(\rho)}{X(\rho)} + \frac{\rho X'(\rho)}{X(\rho)} = -\frac{Y''(\theta)}{Y(\theta)}\,.$$

Since ρ and θ are independent variables, the equation must be equal to some constant, which we denote λ^2. Separating the variables, we obtain

$$Y''(\theta) + \lambda^2 Y(\theta) = 0,,$$

which is a second order ordinary differential equation (ODE). We assume a solution of the form

$$Y(\theta) = \mathrm{e}^{(r\theta)}\,,$$

where the characteristic polynomial is $r^2 + \lambda^2 = 0$ [15]. The roots are $r = \pm\lambda\mathbf{i}$, which leaves a solution of the form

$$Y(\theta) = c_1 \exp(\lambda\mathbf{i}) + c_2 \exp(-\lambda\mathbf{i})\,.$$

The complete solution can be expressed as

$$Y(\theta) = a\cos(\lambda\theta) + b\sin\lambda\theta\,.$$

Since the boundary function is periodic, λ must be an integer, and we express the solution as

$$Y(\theta, m) = a_m \cos(m\theta) + b_m \sin m\theta = c_m \cos(m\theta + \phi)\,, \tag{62}$$

where m is an integer, and ϕ is the phase angle. Substituting into (61), and canceling the $c_m \cos(m\theta + \phi)$ term, we obtain

$$X''(\rho) + \frac{1}{\rho}X'(\rho) - \frac{m^2}{\rho^2}X(\rho) = 0\,.$$

Assume $X(\rho) = \rho^a$, and we obtain

$$a(a-1)\rho^{a-2} + a\rho^{a-2} - m^2\rho^{a-2} = 0\,.$$

For arbitrary ρ and a, we have $a^2 - m^2 = 0$, or $a = \pm m$. Combining the solutions, we obtain

$$u(\rho,\theta) = a_0 + \sum_{m=1}^{\infty} \rho^m \left(a_m \cos(m\theta) + b_m \sin m\theta\right) + \sum_{m=1}^{\infty} \rho^{-m} \left(a_m \cos(m\theta) + b_m \sin m\theta\right) .$$

Because solutions containing ρ^{-m} are not physically realistic at $\rho = 0$, we discard and obtain

$$u(\rho,\theta) = a_0 + \sum_{m=1}^{\infty} \rho^m \left(a_m \cos(m\theta) + b_m \sin(m\theta)\right) . \tag{63}$$

The Fourier sine and cosine series allows us to solve for the coefficients a_m and b_m as

$$\begin{array}{rcccccl} a_0 & = & \frac{1}{\pi}\int_{-\pi}^{\pi} g(\theta)\, d\theta & = & \frac{1}{\pi}\int_{-\alpha_0}^{\alpha_1} d\theta & = & \frac{\alpha_1-\alpha_0}{2\pi}, \\ a_m & = & \frac{1}{\pi}\int_{-\pi}^{\pi} g(\theta)\cos(m\theta)\, d\theta & = & \frac{1}{\pi}\int_{-\alpha_0}^{\alpha_1} \cos(m\theta)\, d\theta & = & \frac{\sin(m\,\alpha_1)-\sin(m\,\alpha_0)}{m\,\pi}, \\ b_m & = & \frac{1}{\pi}\int_{-\pi}^{\pi} g(\theta)\sin(m\theta)\, d\theta & = & \frac{1}{\pi}\int_{-\alpha_0}^{\alpha_1} \sin(m\theta)\, d\theta & = & \frac{\cos(m\,\alpha_1)-\cos(m\,\alpha_0)}{m\,\pi}, \end{array}$$

where we are using the boundary conditions from Section 4.2.3. We approximate the boundary function as a continuous function and take the limit as $\Delta\theta \to 0$. The infinite series limits the utility of this solution; fortunately, there is a further simplification remaining by converting the infinite series to a complex logarithm

$$\begin{aligned} u(\rho,\theta) = \frac{\alpha_1-\alpha_0}{2\pi} & - \frac{\mathbf{i}}{2\pi}\log\left(1 - \mathrm{e}^{\mathbf{i}(\alpha_0-\theta)}\rho\right) \\ & + \frac{\mathbf{i}}{2\pi}\log\left(1 - \mathrm{e}^{\mathbf{i}(\alpha_1-\theta)}\rho\right) \\ & + \frac{\mathbf{i}}{2\pi}\log\left(1 - \mathrm{e}^{-\mathbf{i}(\alpha_0-\theta)}\rho\right) \\ & - \frac{\mathbf{i}}{2\pi}\log\left(1 - \mathrm{e}^{-\mathbf{i}(\alpha_1-\theta)}\rho\right) , \\ u(\rho,\theta) = \frac{\alpha_1-\alpha_0}{2\pi} & - \frac{\mathbf{i}}{2\pi}\log\left(1 - \rho\cos(\alpha_0-\theta) - \mathbf{i}\rho\sin(\alpha_0-\theta)\right) \\ & + \frac{\mathbf{i}}{2\pi}\log\left(1 - \rho\cos(\alpha_1-\theta) - \mathbf{i}\rho\sin(\alpha_0-\theta)\right) \\ & + \frac{\mathbf{i}}{2\pi}\log\left(1 - \rho\cos(\alpha_0-\theta) + \mathbf{i}\rho\sin(\alpha_0-\theta)\right) \\ & - \frac{\mathbf{i}}{2\pi}\log\left(1 - \rho\cos(\alpha_1-\theta) + \mathbf{i}\rho\sin(\alpha_0-\theta)\right) . \end{aligned}$$

The complex logarithm, $\log(a + b\mathbf{i}) = \log\left(\sqrt{a^2+b^2}\right) + \mathbf{i}\left(\tan^{-1}\frac{b}{a} + 2\pi k\right)$. Letting $k = 0$, rewriting the logarithm, and collecting terms, the final simplification yields

$$\begin{aligned} u(\rho,\theta) = \frac{\alpha_1-\alpha_0}{2\pi} & + \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin(\alpha_1-\theta)}{1-\rho\cos(\alpha_1-\theta)}\right) \\ & - \frac{1}{\pi}\tan^{-1}\left(\frac{\rho\sin(\alpha_0-\theta)}{1-\rho\cos(\alpha_0-\theta)}\right) . \end{aligned} \tag{64}$$

B Boundary Discontinuities

In order to generate a vector field with continuous derivatives, as needed by our dynamic control policy, we require a C^2 diffeomorphism from an arbitrary polytope to our model space. Unfortunately, the intersections of the half space constraints form corners on the polytope, which preclude the possibility of constructing a C^2 diffeomorphism on the boundaries. Furthermore, numerical issues may arrive in the vicinity of a corner. In this case, the mapping can only approximate the polygon near the corners while maintaining C^2 continuity and numerical stability. Our construction approximates these corners in a natural way, thereby preserving the vector field properties presented in this report. We present the derivation for polygonal cells in $\mathbb{R}^2$. The ideas extend to higher dimensions with some added complications, which we discuss.

Consider the polygon vertex shown in Figure 34. The j^{th} vertex is formed by the intersection of the i^{th} and j^{th} faces. Associated with each face is a normal vector, n_i and n_j. We wish to approximate the polygon near the vertex by "rounding the corner" with a *fillet* curve, borrowing the term from computer aided design (CAD). Figure 34 shows such a curve. The fillet curve should become tangent to the face in a way that provides C^2 continuity along the boundary. This implies that the second derivative should be zero at the point of tangency to each face. To construct such a curve, we begin by defining a frame of reference.

The bisector of the two faces is given by the vector

$$\bar{n} = \frac{n_i + n_j}{||n_i + n_j||}.$$

Let $\bar{n}^\perp$ be given by the 90 degree clockwise rotation of $\bar{n}$. Our right-hand frame of reference is given by $\bar{n}^\perp$ and $\bar{n}$. We need to define an equation for each line corresponding to the polygon faces. Let $n_i^\perp$ and $n_j^\perp$ be given by the rotation of the i^{th} and j^{th} normal vectors clockwise and counter clockwise respectively, as shown in Figure 34. Note, we are assuming the faces and vertices are numbered clockwise around the polygon.

We wish to approximate the polygon within fixed distance, β_R, of the vertex, v. Let β_R be the radius of a circle centered at the vertex, as shown in Figure 34. Let the point of tangency on the i^{th} (resp. j^{th}) face be given by $p_i = v + \beta_R n_i^\perp$ ($p_j = v + \beta_R n_j^\perp$). The coordinates of the point of tangency in our local coordinate

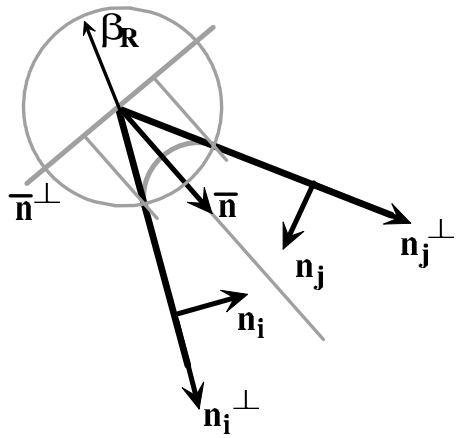


Figure 34: The polygon vertex is approximated with a C^2 fillet curve.

frame are given by

$$\begin{aligned} p_i &= (x_i\,,\,y_i) = \beta_R\,\left(n_i^{\perp}\cdot\bar{n}^{\perp}\,,\,n_i^{\perp}\cdot\bar{n}\right)\,, \\ p_j &= (x_j\,,\,y_j) = \beta_R\,\left(n_j^{\perp}\cdot\bar{n}^{\perp}\,,\,n_j^{\perp}\cdot\bar{n}\right)\,. \end{aligned}$$

Because the local frame bisects the vertex, the tangency points are symmetric with respect to the local frame.

The equation for the fillet curve, in local coordinates, is given as a fourth order polynomial. Fourth order is required because we must match point, slope (first derivative), and second derivative at one point of tangency , the point at the second endpoint (to enforce symmetry), and the slope at the mid point. This requires five parameters in total. Our fillet curve is then expressed as

$$y\,(x) = a\,x^4 + b\,x^3 + c\,x^2 + d\,x + e\,, \tag{65}$$

where a, b, c, d, and e are the unknown parameters. We want the curve to be at its minimum (in local coordinates) at $x = 0$, which implies that $d = 0$. The remaining parameters can be solved using

$$\begin{bmatrix} x_i^4 & x_i^3 & x_i^2 & 1 \\ 4\,x_i^3 & 3\,x_i^2 & 2\,x_i & 0 \\ 12\,x_i^2 & 6\,x_i & 2 & 0 \\ x_j^4 & x_j^3 & x_j^2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ e \end{bmatrix} = \begin{bmatrix} y_i \\ \frac{y_i}{x_i} \\ 0 \\ y_j \end{bmatrix}\,. \tag{66}$$

Given the solution to (66), we may test any point q lying within the circle of radius β_R. Let $(x_q\,,\,y_q) = \left(\bar{n}^{\perp}\cdot q\,,\,\bar{n}\cdot q\right)$ be the point in local coordinates. If $y\,(x_q) > y_q$, then we use $(x_q\,,\,y\,(x_q))$ as our approximate point for evaluating the vector field properties. In cell coordinates, we have $q^* = v + x_q\,\bar{n}^{\perp} + y\,(x_q)\,\bar{n}$, where q^* is our approximated point if $y\,(x_q) > y_q$. We use q^* to evaluate the vector field, and in this way points between the fillet curve and the polygon boundary inherit the properties of points on the fillet curve. Figure 35 shows the results of such a mapping.

This basic idea for approximating the discontinuities of the boundary tangents extends to higher dimensions, although, the calculations become much more complicated. The intersection of two half space constraints in $\mathbb{R}^2$ is a single point, the vertex. In three dimensions, two half space constraints intersect in a line. The methodology for calculating the fillet curve is the same, but now the curve is extruded along the line of intersection forming a fillet surface. Three half space constraints intersect for form a vertex for the polyhedra. In this case, the approximation requires a 2-surface that continuously blends the approximation of the three pairs of intersections. For higher dimensions, the number of intersection types increases, as does the order of the required approximating surface. The formulation of these approximations is beyond the scope of this report.

a) Normalized vector field near outlet vertex

b) Normalized vector field near inlet vertex

Figure 35: The resulting vector field near a polygon vertex using the fillet curve approximation.

References

[1] Chris Atkeson, Andrew Moore, and Stefan Schaal. Locally weighted learning for control. *AI Review*, 11:75–113, April 1997.

[2] John S. Bay. *Fundamentals of Linear State Space Systems*. WCB/McGraw-Hill, 1999.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[4] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, 4(3):3–17, 1985.

[5] William M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry* . Academic Press, 1986.

[6] Johann Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[7] Michael S. Branicky. Stability of switched hybrid systems. In *Proceedings of the 33rd Conference on Decision and Control*, pages 3498–3503, 1994.

[8] Michael S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, MIT, Dept. of Elec. Eng. And Computer Sci., June 1995.

[9] Michael S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, April 1998.

[10] Michael S. Branicky. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1), January 1998.

[11] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, 1999.

[12] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *29th Annual Symposium on Foundations of Computer Science*, pages 306–316, October 1988.

[13] John Canny, Ashutosh Rege, and John Reif. An exact algorithm for kinodynamic planning in the plane. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 271–280. ACM Press, 1990.

[14] Alongkrit Chutinan and Bruce H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410, September 2001.

[15] Earl A. Coddington. *An Introduction to Ordinary Differential Equations*. Dover Publications, Inc., 1961.

[16] C. I. Connolly. Applications of harmonic functions to robotics. In *IEEE International Symposium on Intelligent Control*, pages 498–502, August 1992.

[17] C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using laplace's equation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2102–2106, May 1990.

[18] C. I. Connolly and R. A. Grupen. The applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10:931–946, October 1993.

[19] C. I. Connolly, K. X. Souccar, and R. A. Grupen. A hamiltonian framework for kinodynamic planning and control. In *IEEE International Conference on Robotics and Automation*, pages 2746–2751, May 1995.

[20] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE, Special Issue on Hybrid Systems*, 88(7):1069–1082, July 2000.

[21] Bruce Randall Donald, Patrick G. Xavier, John F. Canny, and John H. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, 1993.

[22] Tobin A. Driscoll and Lloyd N. Trefethen. *Schwarz-Christoffel Mapping*. Cambridge University Press, 2002.

[23] Howie Choset *et al.* . *Foundations of Robot Motion*. www.motionplanning.com, 2003.

[24] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, RI, 1998.

[25] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, March 1997.

[26] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pages 278–292. IEEE Computer Society Press, 1996.

[27] Robert Holmberg and Oussama Khatib. Development and control of a holonomic mobile robot for mobile manipulation tasks. *International Journal of Robotics Research*, 19(11):1066–1074, 2000.

[28] George Kantor and Alfred A. Rizzi. Feedback control of underactuated systems via sequential composition: Visually guided control of a unicycle. In *Proceedings of 11th International Symposium of Robotics Research*, October 2003.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[30] J. M. Keil. Decomposing a polygon into simpler components. *SIAM J. Comput.*, 14:799–817, 1985.

[31] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[32] Jin-Oh Kim and Pradeep Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, June 1992.

[33] Robert Kindel, David Hsu, Jean-Claude Latombe, and Stephen Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE International Conference on Robotics and Automation*, pages 537–543, March 2000.

[34] Nak Yong Ko and Reid Simmons. The lane curvature method for local obstacle avoidance. In *IEEE International Conference on Robotics and Automation*, October 1998.

[35] Daniel E. Kodischek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11:412–442, 1990.

[36] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE International Conference on Robotics and Automation*, pages 1–6, 1987.

[37] Daniel E. Koditschek. The control of natural motion in mechanical systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, 113(4):548–551, December 1991.

[38] Daniel E. Koditschek. Some applications of natural motion control. *ASME Journal of Dynamic Systems, Measurement, and Control*, 113(4):552–557, December 1991.

[39] A. N. Kolmogorov and S. V. Fomin. *Introduction to Real Analysis*. Dover Publications, Inc., 1975.

[40] B. H. Krogh. A generalized potential field approach to obstacle avoidance control. In *SME Conf. Proc. Robotics Research: The Next Five Years and Beyond*, Bethlehem, Pennsylvania, August 1984.

[41] Bruce Krogh and Charles Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation (ICRA '86)*, pages 1664 – 1669, April 1986.

[42] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[43] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 473–479, 1999.

[44] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[45] Steven M. LaValle. From dynamic programming to RRTs: Algorithmic design of feasible trajectories. In A. Bicchi, H. I. Christensen, and D. Prattichizzo, editors, *Control Problems in Robotics*, pages 19–37. Springer-Verlag, Berlin, 2002.

[46] Steven M. LaValle and Michael S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Workshop on the Algorithmic Foundation of Robotics (WAFR)*, Nice, France, December 2002.

[47] Daniel Liberzon and A. Stephen Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems*, pages 59–70, October 1999.

[48] John Lygeros, Datta N. Godbole, and Shankar Sastry. Verified hybrid controllers for automated vehicles, April 1998.

[49] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[50] Matthew T. Mason. *Mechanics of Robotic Manipulation*. The MIT Press, 2001.

[51] Mathworld.com. Poincaré Conjecture, 2003.

[52] E. Mazer, J. Ahuactzin, G. Talbi, and P. Bessiere. The ariadne's clew algorithm. In *Second International Conference on Simulation of Adaptive Behavior SAB92*, 1992.

[53] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessiere. The ariadne's clew algorithm. *Journal of Artificial Intelligence Research*, 9:295–316, November 1998.

[54] Arthur Quaid and Alfred A. Rizzi. Robust and efficient motion planning for a planar robot using hybrid control. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4021 – 4026, April 2000.

[55] E. Rimon and D. E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 21–26, May 1989.

[56] Elon Rimon. A navigation function for a simple rigid body. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 546–551, April 1991.

[57] Elon Rimon and Daniel E. Kodischek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[58] Elon Rimon and Daniel E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Transactions of the American Mathematical Society*, 327(1):71–115, Sept 1991.

[59] Alfred A. Rizzi. Hybrid control as a method for robot motion programming. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 832 – 837, May 1998.

[60] Robert C. McOwen. *Partial Differential Equations: Methods and Applications*. Pearson Education, Prentice Hall, 2nd edition, 2003.

[61] Nicholas Roy and Sebastian Thrun. Motion planning through policy search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 2419–2424, Lausanne, Switzerland, September 2002.

[62] Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, 1999.

[63] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *IEEE International Conference on Robotics and Automation*, April 1996.

[64] Herbert A. Simon. *Models of Man: Social and Rational*. John Wiley and Sons, Inc., 1957.

[65] J.-J.E. Slotine and H.S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, 1989.

[66] Peng Song and Vijay Kumar. A potential field based approach to multi-robot manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1217–1222, 2002.

[67] Cyrill Stachniss and Wolfram Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments, September 2002.

[68] John A. Thorpe. *Elementary Topics in Differential Geometry*. Springer, 1978.

[69] Iwan Ulrich and Johann Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1572–1577, Leuven, Belgium, May 1998.

[70] Iwan Ulrich and Johann Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation*, pages 2505–2511, San Francisco, CA, April 2000.

[71] W. D. Wallis. *A Beginner's Guide to Graph Theory*. Birkhäuser, 2000.

Acknowledgements

This work was supported by the Army Research Office under grant DAAD19-02-01-0383.

David Conner would like to thank Adrian Tudorascu for his help with the general form of the solution to Laplace's equation, and Ji Yeong Lee for help with the proof of Lemma 5.1. Thanks also to Mr. Conner's colleagues Aaron Greenfield, David Maiwand, Prasad Atkar, and Sarjoun Skaff for other helpful conversations.

