SELECTIVE DETERMINISM FOR AUTONOMOUS NAVIGATION IN MULTI-AGENT SYSTEMS

Jeffrey Kane Johnson

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements for the degree

Doctor of Philosophy

in the School of Informatics, Computing, and Engineering,

Indiana University

September, 2017

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

David Crandall, PhD, Chair

Paul Purdom, PhD

Martha White, PhD

Kevin Pilgrim, PhD

Christoph Stiller, PhD

April 28, 2017

Valerie, this is for you. And for little Katherine Kane.

ACKNOWLEDGMENTS

I would like to express my gratitude first to Dr. Kris Hauser who introduced me to the field of robotics and whose rigor and insights as a teacher, adviser, and researcher gave me the foundation necessary to pursue new ideas and become a successful researcher in my own right.

I would like to thank my research committee, Drs. David Crandall, Paul Purdom, Martha White, Kevin Pilgrim, and Christoph Stiller, for their valuable comments, suggestions, and ideas that helped shape and improve the quality of this thesis.

I also want to thank everyone that I have collaborated with. Thanks to Dr. Rob Burridge for providing expert guidance during my internship at TRACLabs. Thanks to Drs. Jan Becker, Shilpa Gulati, Sören Kammel, Philipp Robbel, Jason Hardy, Jörg Müller, and Mikael Persson, among many, many others, for countless valuable discussions during my time at Robert Bosch, LLC in Palo Alto, CA. To everyone else I have worked, talked, and debated with over the years, thank you.

Many thanks to my family, especially my wonderful mother, Veronica. My family are simply the most wonderful and supportive people I know.

Finally and foremost, I thank my wife Valerie. She is my dearest friend, closest companion, and most formidable adversary. I could have accomplished none of this without her unwavering intellectual help and emotional support.

Jeffrey Kane Johnson

SELECTIVE DETERMINISM FOR AUTONOMOUS NAVIGATION IN MULTI-AGENT SYSTEMS

Standard approaches to multi-agent navigation problems formulate them as searches for policies that are optimal mappings from belief states to actions. However, computing such policies is almost always intractable, both in theory and in practice, due in part to the combinatorial effects of reasoning about uncertain interactions into the future. This dissertation proposes a framework to address that intractability by identifying when and how interaction effects can be factored out of the problem while maintaining collision guarantees and goal-directed motion.

At a low level, stochastic optimal control theory is leveraged to formulate a constrained interference minimization principle within which multi-objective control problems can be formulated and solved to a defined level of confidence. At a high level, it is shown that, under certain conditions, complex multi-agent decision process problems can be factored into independent sub-problems, which removes coordination effects and greatly reduces overall complexity. These two results are unified into a single problem solving strategy called the *Selective Determinism* (SD) framework, which enables robust and efficient solutions to multi-agent navigation problems.

David Crandall, PhD, Chair

Paul Purdom, PhD

Martha White, PhD

Kevin Pilgrim, PhD

Christoph Stiller, PhD

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robotics, fundamentally, is the problem of how machines engage in meaningful real-world interactions. These interactions require reasoning at varying levels of complexity, and as machines have become capable of more complex reasoning, robots have become capable of more complex interactions. Examples of domains in which robot successfully engage in real-world interactions include mining [63, 146], health care [89, 99], and automated driving [43, 108, 145].

While such applications are impressive, there are still basic types of interaction problems whose general solutions have remained elusive. One such problem is the motion planning and decision making problem of how to navigate quickly and efficiently in the presence of obstacles and other intelligent agents. The problem is referred to here as the multi-agent navigation problem, and is defined broadly below:

Problem 1. For a set of agents navigating a shared space, each agent may assume that all agents will choose to avoid collision and to avoid causing collision, but that otherwise agent decision processes are not fully observable. In such a system, how can an agent compute controls to navigate toward a given goal without violating collision constraints?

Problem 1 is the kind of problem pedestrians face when navigating crowded sidewalks

or drivers face when navigating crowded roadways. Variants and aspects of this problem occur commonly in the world and have received much attention from researchers, e.g. [62, 84, 86, 131, 147, 157], but it has proven surprisingly difficult to provide efficient, robust, and practical robotics solutions for them. To address this problem in a practical and meaningful way, this dissertation builds, in part, on an idea explored in van den Berg et al. [153]: that under certain conditions multi-agent navigation can be achieved by decomposing the problem into separate collision avoidance and goal direction sub-problems. In order for this idea to be put to use, the following questions must be addressed:

1. In a stochastic multi-agent system, precisely how can an instance of Problem 1 be decomposed into independent sub-problems?

2. If an instance of Problem 1 is decomposed into separate sub-problems, how can the solutions of those sub-problems be re-combined to provide a solution to the original problem?

The primary contribution of this dissertation is an in-depth investigation of these questions and their answers, as well as experimental results produced with a novel type of motion planner. Chapter 2 develops the motion planner and notes that, while it is not suited for use in multi-agent systems on its own, it has important properties that can be exploited by a more general navigation solution. Chapter 3 specifically addresses Question 2 in the context of stochastic optimal control by defining a probabilistic framework that enables blended-priority control computation to a given level of confidence. Chapter 4 specifically addresses Question 1 by deriving a general, dynamics-based decision problem for whether a multi-agent system requires coordination among agents in order to remain collision free. As is shown, solving this decision problem is precisely what enables a correct navigation

problem decomposition. Finally, Chapter 5 unifies the results of Chapters 3 & 4 into a single Selective Determinism framework, which is the solution this dissertation provides to Problem 1. To demonstrate the framework, a path network traversal problem is defined and the motion planning algorithm of Chapter 2 is employed to formulate a Selective Determinism solution. The demonstration shows one of the more powerful aspects of the Selective Determinism framework: that it enables simple, deterministic planning algorithms to be deployed in complex multi-agent systems where their use would otherwise be inappropriate.

The following sections introduce the multi-agent navigation problem in more details and provide context for the contributions of this dissertation in the field.

1.1 Multi-agent navigation: "good enough" is often good enough

Classically, the intelligence facet of the navigation problem has been approached from two sides: perception and planning. In the context of a perception/planning decomposition, perception is the problem of creating an abstraction of the world within which reasoning can occur, while planning is the process of reasoning within that abstraction. The goal of this dissertation is to advance the state of the art in multi-agent navigation planning under the assumption of reasonable perception input.

When a person thinks about how to solve a given problem, they might naturally want to solve it in the 'best' way possible, that is, to find the 'optimal' solution. For small and well-defined problems, such a solution might be relatively easy to arrive at. But when problems become large, uncertain, or not-so-well defined, finding an optimal solution becomes difficult or impossible. In such cases, a person might instead just find a solution that is sufficient for their needs. When a machine is put to the same big, complex, problem, however, people often expect that the machine will be able to find the true optimal solution and are

disappointed when it cannot. This is an unfortunate prejudice because the ability to find sufficient solutions is exceedingly powerful, especially in domains where global optimality is not possible or practical.

Multi-agent navigation is precisely one such problem domain. Many popular approaches to these problems are made tractable through randomization or approximation, but even then they are often still modified versions of globally optimal solvers formulated fundamentally in terms of finding a globally optimal solution according to some cost or reward function. But as covered in Chapters 2, 3, & 4 such optimal navigation solutions for multi-agent systems, particularly in stochastic systems, cannot tractably be computed. They simply are not solvable. One significant cause of this intractability is the inter-dependencies of agent actions: if the goal is solve a multi-agent system optimally, one necessarily must consider interaction effects not only of one's own actions on the other agents, but of the actions of all other agents on each other.

In light of this problem with optimal solution techniques, the problem statement in Problem 1, by design, does not mention optimality. It may be advantageous to perform optimization in some manner and to some degree, but, as stated, there's no need for the solution to be optimal *with respect to the problem statement*. This leaves the door open for sufficient, or "good enough," solutions. One such sufficient solution frames the multi-agent navigation problem in terms of *guided collision avoidance*:

Definition 1. Guided collision avoidance describes the strategy of choosing goal-directed motions from the space of collision avoiding controls in order to navigate to a goal while satisfying collision constraints.

Guided collision avoidance exploits the fact that, at any given time, the set of collision

avoiding controls available to an agent typically has a non-trivial measure, which implies that some of those controls could make more progress toward a given goal than others. Choosing from within this restricted set of controls allows an agent to bias its motion toward a goal without having to reason jointly about collision avoidance *and* goal direction. This approach serves as a basis for the derivation in Chapter 5 of the Selective Determinism framework, which allows general, efficient, and robust algorithmic solutions to the general multi-agent navigation problem to be derived. Key to the efficiency of problem solutions formulated under this framework is the fact that, under certain conditions, the interaction effects mentioned earlier can be removed from the problem.

1.2 Background of approach

As alluded to earlier, solution techniques for navigation problems are typically formulated as searches for control policies, which are optimal mappings from belief states to actions. Unfortunately, algorithms that solve for these control policies generally belong to theoretically intractable complexity classes. This is partially due to the combinatorial explosion that results from reasoning about how the effects of uncertain interactions propagate into the future. An approach to achieving tractability, therefore, is to *factor out* these interaction effects.

Previous work has exhaustively demonstrated that the naïve approach of complete factorization, i.e., assuming complete independence of all agent actions, dramatically simplifies planning problems, and, for that reason, it has been a dominant paradigm in practice for solving navigation problems in multi-agent systems [14, 107, 150]. While the computational benefits of such a factorization are substantial, real-world systems often require some degree of interaction, and formulating them with the overly strong complete independence assump-

tion can produce undesirable results, including policies that are too conservative to be useful, too aggressive to be safe, or that behave non-intuitively in the presence of human agents.

A completely different approach takes the classical perception/planning decomposition and discards it in favor of a pure machine learning approach. Such *end-to-end* techniques attempt to learn a direct mapping from sensor input to control output that obviates the need for any human-designed sub-systems. End-to-end solutions have shown some success in limited domains, e.g. [103, 122, 124, 136]; however, Shalev-Shwartz et al. [127] contains a crucial observation about their use particularly in safety-critical systems, specifically automobiles. Quoted directly:

> Using machine learning, and specifically RL, raises two concerns .... The first is about ensuring functional safety of the Driving Policy–something that machine learning has difficulty with given that performance is optimized at the level of an expectation over many instances. Namely, given the very low probability of an accident the only way to guarantee safety is by scaling up the variance of the parameters to be estimated and the sample complexity of the learning problem– to a degree which becomes unwieldy to solve. Second, the Markov Decision Process model often used in robotics is problematic ... because of unpredictable behavior of other agents in this multi-agent scenario.

These concerns hold not just for automobiles, but for any robotic agents operating in environments where hard guarantees must be made about their behavior. Because these types of domains are the primary focus of this work, it is reasonable to consider end-to-end solutions on their own to be insufficient solution methods for the guided collision avoidance problem. Instead, they should be augmented in a way that desired hard constraints can be enforced.

In order to achieve the benefits of factorization and maintain hard constraint guarantees while avoiding overly strong independence assumptions, this dissertation derives theory and algorithms for the factoring of interaction effects based on collision avoidance behaviors. The factorization is enabled by formulating the navigation problem as a guided collision

avoidance problem with two key assumptions: that global optimality is not required, and that agents' behaviors are self preserving. The novelty of the contributions is demonstrated through surveys of background literature in each technical chapter, and proofs of correctness, complexity analysis, and experimental results are provided where relevant.

1.3 Outline of technical content

Chapter 2 presents the first exact, polynomial-time, speed profile planning algorithm to handle speed and acceleration bounds in the presence of dynamic obstacles. It allows arbitrary agent trajectories and polygonal models for agent geometry, and it is shown capable not only of planning time-optimal speed profiles in cluttered scenarios, but also of detecting inevitable collision states in deterministic systems. Simulation tests suggest that the planning system is fast enough for real-time use, with re-planning rates of 10Hz possible on consumer hardware.

Chapter 3 uses principles of stochastic optimal control theory to derive a generic constrained interference minimization principle that can be used to enable shared autonomy or multi-objective systems to behave in well-defined ways in the presence of uncertainty. In addition, a specialized safety-constrained variant of the principle is derived and applied to the specific use case of semi-autonomous collision avoidance systems for automobiles. Two concrete implementations of the principle are developed: First, a probabilistic collision-avoidance braking strategy is given that considers uncertainty in vehicle dynamics, sensor noise, and unpredictable obstacle behavior. Second, a technique is given for determining safe trajectories for intersection crossings in the presence of state uncertainty. A number of Monte Carlo simulations demonstrate that using the principle achieves low collision risk and driver interference in a variety of scenarios.

Chapter 4 demonstrates that formulating the general problem of multi-agent collision avoidance in terms of relaxed global optimality can allow the complexity to drop dramatically *without* losing any guarantees about being able to remain collision free. This can be accomplished by maintaining non-coordination requirements, which allows independence of behaviors to be assumed during execution and factors out the interaction effects mentioned previously. A theory for this factorization is derived from an examination of agent coordination in a variant of the reciprocal n-body collision avoidance problem. The key insight is that system dynamics can introduce a requirement for coordination where there otherwise would be none. The relationship between the coordination requirement and problem complexity is demonstrated in a constructive proof that also provides an existence test for the coordination requirement that can be used in applications.

Chapter 5 develops and analyzes the Selective Determinism (SD) framework that allows problem-specific algorithms to be defined that perform planning under factorized behaviors. The framework consists of two primary parts: a global guidance controller, and a local collision avoidance controller that maintains the non-coordination guarantees defined in Chapter 4. Output from these controllers is blended under the principle presented in Chapter 3. A sample problem is presented and problem-specific routines were derived under the SD framework to demonstrate its use. Exploiting the properties of the efficient speed profile planning algorithm derived in Chapter 2, the solution to the sample algorithm is demonstrated to be computationally efficient while still tackling a non-trivial problem.

The glossary included at the end of the document provides definitions for some common terms whose meaning may be unclear or may vary across disciplines. In the electronic version of the document, the terms are linked to allow the reader to jump between definition and usage.

Chapter 2

Path-time planning

Synopsis: *This chapter derives an optimal, exact, polynomial-time planner for bounded-acceleration trajectories along fixed paths in the presence of moving obstacles. The planner constructs reachable sets in the path-speed-time (PST-space) space by propagating reachable speed sets between obstacle tangent points in the path-time (PT-space) plane. The terminal speeds attainable by endpoint-constrained trajectories in the same homotopy class are proven to span a single interval, so the planner merges contributions from individual homotopy classes to find the exact range of reachable speeds and times at the goal. A reachability analysis proves that running time is polynomial given reasonable assumptions, and empirical tests demonstrate performance.*

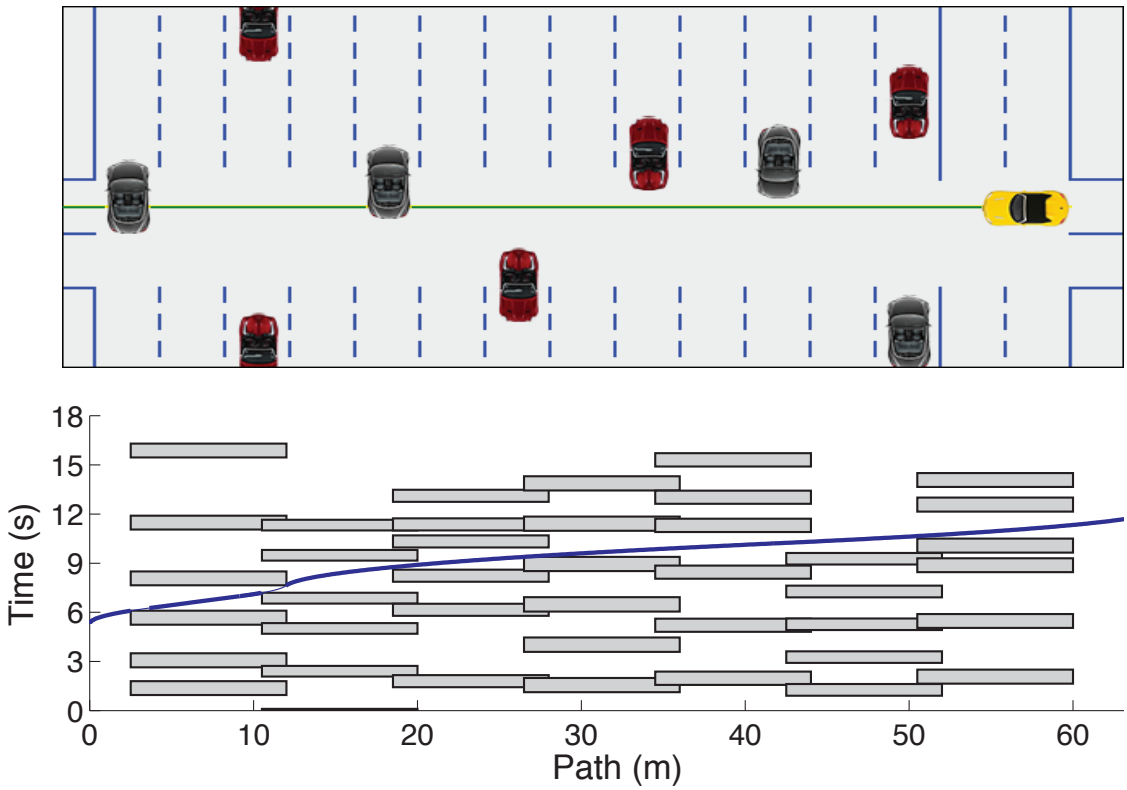


Figure 2.1: Top: A non-signalized (i.e. no lights or signs) lane crossing scenario. Bottom: A plot of the path-time obstacles (shaded) and the optimal path-position profile (curve).

2.1 Introduction

Consider the traffic scenario shown in Figure 2.1 in which a vehicle must cross multiple lanes in a non-signalized (i.e. no lights or signs) intersection. In order to cross safely, the vehicle must time its speed along a path in order to maneuver between gaps in traffic. Unsurprisingly, this particular traffic scenario and others like it (such as non-signalized lefthand turns) pose significant safety risks to drivers[1]. In general, any traffic scenario in which paths of travel cross directly increase the likelihood of an accident [2]. Points at which paths of travel cross are referred to as *conflict points*.

In an effort to reduce safety risks encountered on the road, industry and academia alike have investigated driver assistance functionality that can intervene in emergency situations

[1]They are the second-highest cause of accidents for elderly drivers in the United States [114]

to mitigate or even prevent collisions. Inspired by these efforts, this chapter develops an algorithm that specifically addresses the problem of *conflict point resolution*, which addresses how to safely navigate conflict points. The hope is that such an algorithm can aid in the development of driver assistance systems.

Building on preliminary work published in Johnson and Hauser [66, 67], §2.2 treats the conflict point resolution problem as a speed planning problem and introduces definitions and notations. §2.3 introduces and defines the representations used to solve the problem, and presents an initial algorithmic solution to the problem, and §2.4 extends that solution to the general solution. Finally, §2.5 covers empirical results and simulated applications.

2.1.1 Related work

Navigation among moving obstacles is a challenging problem with a long history. This section provides a brief overview of work related to the algorithms presented in this chapter.

Generally, in problems with moving obstacles, the agent must control its motion through both space and time, that is, it must plan a path as well as its speed profile along the path. Unfortunately, these types of problems tend to be computationally intractable: for instance, the planar motion planning problem among constant-velocity moving obstacles is PSPACE-hard [126]. To address intractability, randomized techniques have been used. Kindel et al. [82] employed a variant of Probabilistic Roadmaps [79] with fast replanning to plan arbitrary trajectories in the presence moving obstacles and sensor uncertainty. The MIT team [91] for the 2007 DARPA Urban Challenge used Rapidly-exploring Random Trees [88] to plan vehicle trajectories.

A more restricted class of problems is longitudinal control planning, or speed profile planning, which is the problem addressed in this chapter. This problem assumes well-defined

paths are available and that the agent may steer along them with reasonably high accuracy [26, 42, 83]. By decoupling spatial path planning and speed profile planning problems, the overall problem becomes more tractable. The Carnegie-Mellon team in the 2007 DARPA Urban Challenge exploited path/velocity decomposition for road navigation by computing optimal paths based on the centerline of the road lanes [150]. Kant and Zucker [74] addressed the general velocity planning problem with a visibility graph formulation for agents with bounded velocities but unbounded accelerations. This work extends that approach to also consider acceleration bounds. Canny et al. [34] presented an exact exponential-time algorithm for planning optimal motions of a point in a plane among fixed obstacles under L_∞ acceleration bounds. Optimal planning along a path with speed and acceleration bounds along time-varying upper and lower bounds was solved in polynomial time by Ó'Dúnlaing [116] whose solution is closely related to the *channel* construction used here.

Discretization-based techniques allow avoiding dynamic obstacles without a fixed path, but sacrifice exactness and completeness. A global, grid-based method for planning among moving obstacles under acceleration constraints was presented in the velocity obstacles work of Fiorini and Shiller [44]. Other work in vehicle collision avoidance has used techniques like receding horizon control [12] and randomized kinodynamic planning [91, 121] to explore the vehicle's action/state space. Another approach is to consider a discrete lattice of precomputed dynamic maneuvers which allows for deeper exploration using classical search techniques [38, 92].

2.2 The speed planning problem

Suppose an agent A is given a path P. Let s indicate a scalar-valued position (or arc length) along P, $\dot{s}$ indicate scalar-valued speed along P, $\ddot{s}$ indicate scalar-valued acceleration along P,

and t indicate time. Define a three-dimensional path-speed-time (PST-space) state space [47] consisting of dimensions s, $\dot{s}$, and t. Assume agents are always oriented tangentially to their paths such that a PST-space point $(s, \dot{s}, t)$ suffices to describe the configuration of an agent[2]. The planar path-time (PT-space) and path-speed (PS-space) projections will also be used in specifying and solving the problem. A trajectory in PST-space is a function that maps a time value to a point in PST-space. In order to be feasible, trajectories in this space must satisfy dynamic constraints, which are bounds on position, speed, and acceleration, and they must also respect obstacle constraints, which forbid the trajectory from entering regions of space that correspond to simultaneous states of occupancy of agent and obstacle.

Problem 2 defines the speed planning problem addressed in this chapter:

Problem 2. For a path P, obstacle set $\mathcal{O}$, and time horizon T find a feasible trajectory $\mathbf{x}(t)$ along P subject to the below system dynamics, constraints, and conditions, whose parameters are given as input:

System dynamics:

$$\dot{s}_1 = \begin{cases} \dot{s}_{\min} & : \dot{s}_0 + \ddot{s}_0 t < \dot{s}_{\min} \\ \dot{s}_0 + \ddot{s}_0 t & : \dot{s}_0 + \ddot{s}_0 t \in [\dot{s}_{\min}, \dot{s}_{\max}] \\ \dot{s}_{\max} & : \dot{s}_0 + \ddot{s}_0 t > \dot{s}_{\max} \end{cases} \tag{2.1}$$

System constraints:

$$\dot{s}_{\min} \in [0, \infty) \tag{2.2}$$

$$\ddot{s}_{\min} < 0 < \ddot{s}_{\max} \tag{2.3}$$

[2]This is a reasonable assumption for many types of mobile robots, such as automobiles, whose dynamics can be approximated by a bicycle model.

Obstacles constraints:

For each state along a trajectory $\mathbf{x}(t)$, the geometric model of the agent at state $\mathbf{x}$, $A(\mathbf{x})$, must be disjoint from all obstacle models:

$$\forall \mathbf{x} \in \mathbf{x}(t),\ \forall O \in \mathcal{O} :: A(\mathbf{x}) \cap O = \varnothing \tag{2.4}$$

Initial conditions, where ^{s} indicates the start, or initial, value:

$$\dot{s}^s \in [\dot{s}^s_{\min}, \dot{s}^s_{\max}] \subseteq [\dot{s}_{\min}, \dot{s}_{\max}] \tag{2.5}$$

$$\ddot{s}^s \in [\ddot{s}^s_{\min}, \ddot{s}^s_{\max}] \subseteq [\ddot{s}_{\min}, \ddot{s}_{\max}] \tag{2.6}$$

Goal conditions, where ^{g} indicates the goal, or terminal, value:

$$t^g \leqslant T \tag{2.7}$$

$$s^g = |P| \tag{2.8}$$

$$\dot{s}^g \in [\dot{s}^g_{\min}, \dot{s}^g_{\max}] \subseteq [\dot{s}_{\min}, \dot{s}_{\max}] \tag{2.9}$$

$$\ddot{s}^g \in [\ddot{s}^g_{\min}, \ddot{s}^g_{\max}] \subseteq [\ddot{s}_{\min}, \ddot{s}_{\max}] \tag{2.10}$$

2.3 The visibility graph solution with axis-aligned PT-space obstacles

Kant and Zucker [74] showed how the version of this problem with speed constraints but without acceleration constraints can be addressed by a visibility graph formulation in PT-space space. This section extends the visibility graph approach to handle acceleration bounds, where the nodes of the visibility graph correspond to portions of obstacle boundaries in PT-space space, and edges indicate that two nodes can be connected by a feasible trajectory.

It will be shown that the visibility graph can be constructed efficiently, and that a solution trajectory can be efficiently extracted.

To formulate the solution, the notion of visibility is defined in terms of *channels* of dynamically feasible homotopic trajectories. Consider two PT-space points p_1 and p_2. A PST-space trajectory connects p_1 and p_2 if it is feasible and has an initial point that projects onto p_1 and has a terminal point that projects onto p_2. Let trajectories $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ be connecting trajectories between p_1 and p_2. If $\mathbf{x}_1(t)$ can be continuously deformed such that it becomes $\mathbf{x}_2(t)$ without violating connectivity or passing through obstacles, it is homotopic with respect to $\mathbf{x}_2(t)$. Define the region of space that exactly contains all homotopic trajectories between p_1 and p_2 as a channel, and say that p_2 is *visible* from p_1 if and only if there is a non-empty channel connecting them. Using the notion of channels, a visibility graph can be constructed that describes connectivity in PST-space such that feasible trajectories to be efficiently extracted.

Since only the existence of channels is of interest when determining visibility, it is not necessary to compute channel bounds explicitly. Instead, sufficient information for defining visibility is captured by the sets of reachable speeds that trajectories within a channel can attain.

Problem 3 defines the problem of computing reachable speeds:

Problem 3. Let p_1 and p_2 be PT-space points. Compute the set of reachable speeds S_R at p_2 from p_1 under the constraints of Problem 2 and from initial speed $\dot{s}_1$.

By nature, Problem 3 is an optimization problem; however, for several reasons, computing a solution is not as straightforward as applying an existing optimization technique. Standard constrained optimization approaches, such as formulating according to KKT con-

ditions [78, 85], are difficult to apply directly because there is no explicit objective function: the problem is to extremize the boundary value of a function (i.e. a feasible trajectory) that is only *implicitly* defined. Solving the problem requires solving for that extremizing function. Variational calculus[3] and optimal control theory[4] provide tools for solving such problems, but the differential-algebraic nature of the problem implies that generic solvers can probably only approximate the solution [27]. Numeric solvers, e.g. [104, 105], may approximate to arbitrary precision, or analytical techniques, e.g. the method of saturating functions [54], may provide closed-form approximations.

Rather than a numeric, or approximate analytic, solution, it will be shown that a visibility graph can be computed exactly by computing *reachable regions* in PST-space space. In addition, it can be computed efficiently and in deterministic time. The remainder of this section introduces and develops the notion of visibility under dynamic constraints, and then presents the algorithm for computing visibility in unobstructed space. Finally, the algorithm is extended to visibility computation in obstructed space, and a full path-time planning algorithm is defined.

2.3.1 Visibility under dynamic constraints

This section presents visibility computation in unobstructed PST-space.

Definition 2. A P curve is an origin-centered PT-space parabola that defines an arc-length path offset after a given time span Δt for given $\dot{s}_o$ and $\ddot{s}_o$:

$$P(\Delta t, \dot{s}_o, \ddot{s}_o) = \dot{s}\Delta t + \frac{1}{2}\ddot{s}\Delta t^2$$

[3]Problem 3 resembles a state-constrained brachistochrone problem.

[4]The solution space of trajectories derived in this section implies that Problem 3 could be modeled as a bang-singular control problem [32].

Definition 3. For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, a P^+ curve is a PT parabola that defines the path offset achievable at time t under maximum acceleration:

$$P^+(t, \dot{s}) = s_o + P(t - t_o, \dot{s}, \ddot{s}_{\max})$$

Definition 4. For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, a P^- curve is a PT parabola that defines the path offset achievable at time t under minimum acceleration:

$$P^-(t, \dot{s}) = s_o + P(t - t_o, \dot{s}, \ddot{s}_{\min})$$

Definition 5. For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, an L curve is a line that defines an arc-length path offset for an input time t assuming zero acceleration:

$$L(t, \dot{s}) = P(t - t_o, \dot{s}, 0)$$

Definition 6. Without loss of generality, let P refer to either a P^+ or P^- curve, and let t_2 be the time coordinate of p_2. Define P_2 to be a terminal curve that passes through p_2. Define P_1 to be an initial curve with derivative $\dot{s}_1$ at origin point p_1. Define an L curve that is tangent to the initial and terminal P curves at switching times t_{s1} and t_{s2} with derivative

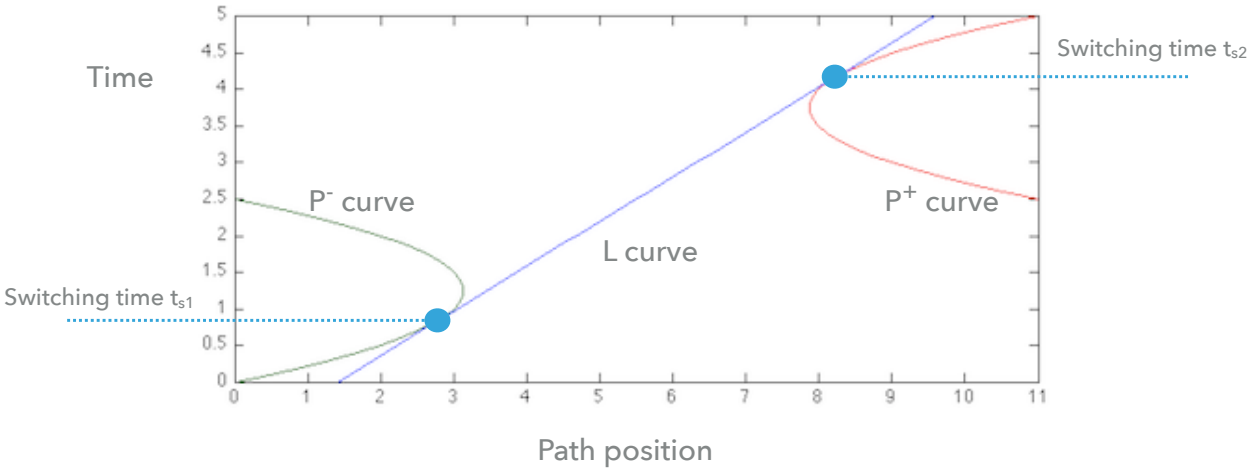


Figure 2.2: This plot shows a P^- curve (left parabola, green), an L curve (straight line, blue), and a P^+ curve (right parabola, red). A bang-singular trajectory connects the origin PT point $(0, 0)$ with the upper right PT point $(11, 5)$ by following the P^- curve until switching time t_{s1}, then following the L curve until switching time t_{s2}, and finally following the P^+ curve to the terminal point.

$\dot{s}_s$. Define a *bang-singular trajectory* as:

$$\mathbf{x}_1(t) = \begin{cases} P_1(t, \dot{s}_1) & : 0 \leqslant t \leqslant t_{s1} \\ L(t, \dot{s}_s) & : t_{s1} < t < t_{s2} \\ P_2(t, \dot{s}_s) & : t_{s2} \leqslant t \leqslant t_2 \end{cases}$$

Figure 2.2 illustrates Definitions 3–6.

Definition 7. Given a PST-space start point $p^s = (s^s, \dot{s}^s, t^s)$, define the PST-space reachable region of PS-space space from p^s at any point in time $t \geqslant t^s$ as $R(t; p^s)$. In other words, $R(t; p^s)$ is the area of the PS-space plane reachable at t from p^s via feasible trajectories.

It will be shown that $R(t; p^s)$ is a relatively simple convex region in the PS-space plane, bounded by, at most, six curves of parabolic or linear type [116], yielding three types of regions. The following assumes the PS-space plane is oriented with the path axis is horizontal, extending positively to the right, and the speed axis is vertical, extending positively upwards.

Type I regions Without loss of generality, let $t^s = 0$. Ignoring speed constraints, the reachable set $R(t; p^s)$ is bounded on the left by a parabola corresponding to bang-bang trajectories at which acceleration is at a minimum for time t_s, and then switches to a maximum for time $t - t_s$. Let these be as P^-P^+ trajectories. The right boundaries are likewise formed by P^+P^- trajectories. The parametric expressions for the left and right bounds are :

$$P^-P^+(t, t_s) = \begin{bmatrix} s(t, t_s) \\ \dot{s}(t, t_s) \end{bmatrix} = \begin{bmatrix} s^s + \dot{s}^s t + \frac{1}{2}(\ddot{s}_{\min} - \ddot{s}_{\max})t_s^2 + \frac{1}{2}\ddot{s}_{\max}t^2 \\ \dot{s}^s + (\ddot{s}_{\min} - \ddot{s}_{\max})t_s + \ddot{s}_{\max}t \end{bmatrix} \tag{2.11}$$

$$P^+P^-(t, t_s) = \begin{bmatrix} s(t, t_s) \\ \dot{s}(t, t_s) \end{bmatrix} = \begin{bmatrix} s^s + \dot{s}^s t + \frac{1}{2}(\ddot{s}_{\max} - \ddot{s}_{\min})t_s^2 + \frac{1}{2}\ddot{s}_{\min}t^2 \\ \dot{s}^s + (\dot{s}_{\max} - \dot{s}_{\min})t_s + \ddot{s}_{\min}t \end{bmatrix} \tag{2.12}$$

where t_s ranges from 0 to t. Note that traveling at constant maximum acceleration achieves the upper right vertex of $R(t; p^s)$, while traveling at minimum acceleration achieves the lower left vertex. Note that both curves are parabolic functions and are monotonically increasing along the domain. See Figure 2.3a.

Type II regions Now consider the case in which the reachable set exceeds a speed maximum before time t. The reachable set boundary in this case adds two segments: a linear portion at $\dot{s} = \dot{s}_{\max}$ and a parabolic segment corresponding to P^+LP^- trajectories that reach $\dot{s}_{\max}$, travel along it for some time, and then decelerate with $\ddot{s}_{\min}$. This latter segment replaces the section of $P^+P^-(t, t_s)$ trajectories that would otherwise exceed $\dot{s}_{\max}$ at their switching time t_s.

The state arrives at the boundary by accelerating until $\dot{s}_{\max}$ is reached at time t_v, progressing with constant speed until time t_s, and then decelerating for time $t - t_s$. It is straightforward to show that this construction maximizes the p value for a given v value given the constraints. The expression for a P^+LP^- boundary curve is:

$$P^+LP^- = \begin{bmatrix} s(t) \\ \dot{s}(t) \end{bmatrix} = \begin{bmatrix} s^s + \dot{s}_{\max} t - \frac{(\dot{s}_{\max} - \dot{s}^s)^2}{2\ddot{s}_{\max}} + \frac{1}{2}(t - t_s)^2 \ddot{s}_{\min} \\ \dot{s}_{\max} + \ddot{s}_{\max}(t - t_s) \end{bmatrix} \tag{2.13}$$

which can be seen in Figure 2.3b. A similar P^-LP^+ curve is obtained for $\dot{s}_{\min}$.

Type III regions These regions occur when both the speed maximum and minimum are exceeded, and consist of six boundary curves: two PP segments, two linear, and two PLP-type segments. See Figure 2.3c.

A simple interpolation lemma proves that $R(t; p^s)$ is convex for all t and p^s.

Lemma 1. *A convex combination $x(t) = \alpha x_a(t) + (1 - \alpha)x_b(t)$, $\alpha \in [0, 1]$ of dynamically feasible trajectories $x_a(t)$ and $x_b(t)$ is also dynamically feasible.*

Proof. This follows from linearity of the differentiation operator and convexity of the speed and acceleration bounds. This results also in the the convexity of $R(t; p^s)$. □

The following lemma is used to simplify reachability analysis because it implies that only bang-singular trajectory classes need be examined:

Lemma 2. *Any state that is dynamically reachable from p^s can be reached by a feasible bang-singular trajectory.*

Proof. Note that all boundary curves of the parabolic type can be reached by PLP or PP trajectories. Let $p = (s, \dot{s}, t)$ be dynamically reachable, that is $p \in R(t; p^s)$. The argument considers examining how $R(t; p^s)$ changes as the acceleration limits are scaled. Write this dependence explicitly as $R(t; p^s, \ddot{s}_{\min}, \ddot{s}_{\max})$, and note that the dependence is continuous. Consider scaling the limits by a factor of γ from $\gamma = 1$ to 0 until p lies on a parabolic

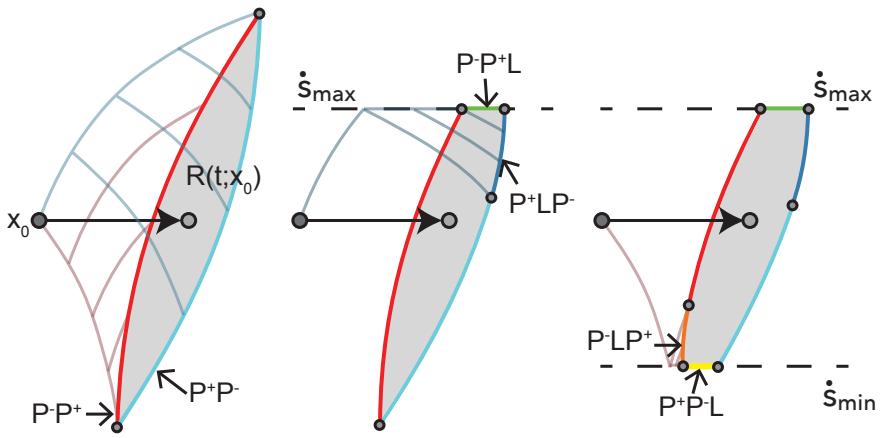


Figure 2.3: Reachable sets (shaded) in the PS plane at a new path position from an initial state x_0 fall into Types I, II, and III. PP curves that remain within speed limits form quadratic left and right boundary segments (left). The boundaries of Type II and III sets also contain horizontal linear segments formed by PPL curves and quadratic segments formed by PLP curves (middle and right).

boundary segment of $R(t; p^s, \gamma\ddot{s}_{\min}, \gamma\ddot{s}_{\max})$. At this value of γ, p can be reached from p^s via a PP or PLP trajectory where the P segments accelerate at rate $\gamma\ddot{s}_{\min}$ or $\gamma\ddot{s}_{\max}$. By definition, any P or PLP trajectory is a bang-singular trajectory. □

Determining reachable speeds

Given the theory derived above, the planner uses the operation $SReach(p^s, s^g, t^g)$ to compute the set of speeds reachable at the goal point (s^g, t^g) from an initial PST-space state $\mathbf{x}^s$. In other words, $SReach$ computes the intersection $[\dot{s}^g_{\min}, \dot{s}^g_{\max}] \equiv R(t^g; p^s) \cap \{(s, v) \quad | \quad s = s^g\}$. As a side effect, it also constructs example trajectories for each extremum of the interval, which will later be used by the planner. The remainder of the section describes in detail how to find the maximum reachable speed $\dot{s}^g_{\max}$. Finding the minimum $\dot{s}^g_{\min}$ is essentially symmetric, so it is not covered in detail.

The maximum reachable speed $\dot{s}^g_{\max}$ is defined at the intersection of a boundary curve of $R(t^g; p^g)$. There are only two possible intersections due to the monotonicity of $R(t^g; p^g)$,

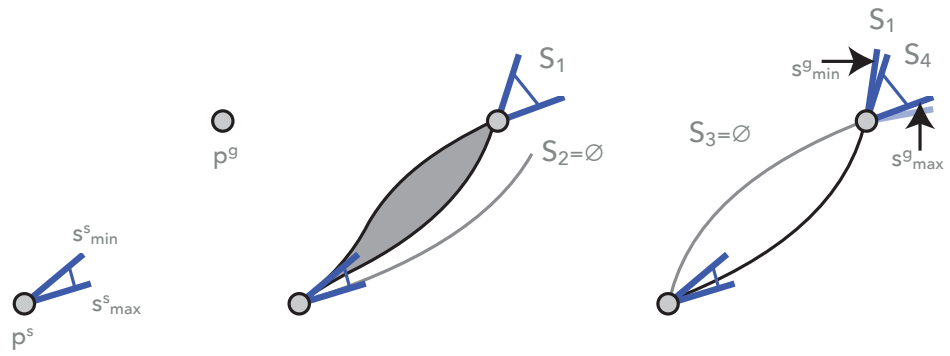


Figure 2.4: Speed interval propagation in the PT-space plane. Left: SIP finds the interval of reachable speeds at p^g from p^s given bounds $[\dot{s}^g_{\min}, \dot{s}^g_{\max}]$ on initial speed. Middle: connecting the extrema of the initial speeds produces intervals S_1, S_2. Right: extremizing goal speeds without imposing initial speed constraint produces intervals S_3, S_4. The final interval $[\dot{s}^g_{\min}, \dot{s}^g_{\max}]$ bounds all four intervals.

and in fact only these types of boundary curves need be considered: P^-P^+, P^-LP^+, and the linear segment boundary with $\dot{s} = \dot{s}_{\max}$. To handle the PP case, the switching time variable t_s is determined by substituting $p = p^g$ into Equation 2.11 and solving the resulting quadratic equation. Similarly, the PLP case is handled by solving for t_s after substituing $p = p^g$ into Equation 2.13. The linear segment case is solved using a PLP scaling method with unknown scale factor γ (see Lemma 2). After enumerating the cases, the $\dot{s}$ bounds and validity of switching times are checked. If no trajectory can be found, then this indicates that $R(t^g; p^s) \cap \{(s, v) \quad | \quad s = s^g\} = \varnothing$, so p^g is unreachable.

Speed interval propagation

This section presents the *speed interval propagation* (SIP) subroutine that forms the backbone of the planner. It takes as input a PT-space start point p^s and an interval of starting speeds $[\dot{s}^s_{\min}, \dot{s}^s_{\max}]$, and a goal point p^g. It outputs the range of speeds $[\dot{s}^g_{\min}, \dot{s}^g_{\max}]$ reachable from p^s for all $\dot{s} \in [\dot{s}^s_{\min}, \dot{s}^s_{\max}]$ (Figure 2.4, left).

SIP proceeds by computing the reachable goal speed intervals S_1 and S_2 from minimum and maximum starting speed: $S_1 = SReach((s^s, \dot{s}^s_{\min}, t^s), p^g)$, and $S_2 = SReach((s^s, \dot{s}^s_{\max}, t^s), p^g)$.

Note that either S_1, S_2, or both may be empty (Figure 2.4, middle). Next, SIP maximizes the goal speed without input speed constraint by constructing a parabolic trajectory with acceleration $\ddot{s}_{\max}$ that interpolates p^s and p^g. If the initial speed $\dot{s}^s$ is in $[\dot{s}^s_{\min}, \dot{s}^s_{\max}]$, it sets $S_3 = \{\dot{s}^g\}$, or $S_3 = \varnothing$ otherwise. The minimum goal speed interval S_4 is similarly computed using a parabola of acceleration $\ddot{s}_{\min}$ (Figure 2.4, right). If S_1 is empty and p^g is indeed reachable, then S_3 must be non-empty. Likewise if S_2 is empty then S_4 must be non-empty. Finally SIP outputs $[\dot{s}^g_{\min}, \dot{s}^g_{\max}]$ which is the smallest interval containing $S_1, \ldots, S_4$.

The planner also uses two variants of SIP. First, it uses the goal speed interval propagation ($GSIP$) subroutine to connect a PT-space point p to the goal position p^g with time and speed constrained within $[0, t_{\max}]$ and $[\dot{s}^g_{\min}, \dot{s}^g_{\max}]$. The second variant is the reverse speed interval propagation ($RSIP$) subroutine used in the backward pass of the planner. $RSIP$ determines the range of incoming speeds that reach an interval of goal speeds. Both are similar to SIP, so they are not discussed in detail.

2.3.2 Reachable sets with one obstacle

The previous section developed a method for computing reachability in unobstructed space, but if the goal is to plan speed profiles in occupied space, the effect of obstacles on reachability must be quantified. This section will illustrate how to determine the set of reachable speeds S^g attainable at a target point p^g with a collision-free trajectory from an initial point p^s with initial $\dot{s}^s \in S^s$ when a single PT-space obstacle occupies the reachable space between p^s and p^g. It will be shown that S^g is the union of two speed intervals, each reachable via trajectories through one of the two homotopy classes (above or below the obstacle), and, furthermore, that these intervals are fully determined by considering connecting extremizing trajectories, and by propagating reachable speeds through the vertices of the obstacle.

Let O be an axis-aligned bounding box in the PT-space plane that represents a *forbidden region* that feasible trajectories may not enter. Let $S^g_{s\to g} \subseteq [\dot{s}_{\min}, \dot{s}_{\max}]$ denote the interval of dynamically reachable speeds at p^g from p^s starting with some set of initial speeds S^s. Note that $S^g_{s\to g} \subseteq S^g$. The SIP algorithm computes $S^g_{s\to g}$ as well as two example trajectories $\mathbf{x}_{max}(t)$ terminating in $\dot{s}_{\max}$ and $\mathbf{x}_{min}(t)$ terminating in $\dot{s}_{\min}$. Topologically, the obstacle O has the effect of splitting the reachable region into at most two distinct channels: C_U, which lies above O, and C_B, which lies below O. Now consider computing S^g_U and S^g_B, which are the sets of speeds reachable through C_U and C_B.

In computing S^g_U there are three cases to consider:

1. O lies below $\mathbf{x}_{min}(t)$ or above $\mathbf{x}_{max}(t)$

2. O is strictly between $\mathbf{x}_{min}(t)$ and $\mathbf{x}_{max}(t)$

3. O is above (or intersects) $\mathbf{x}_{min}(t)$ and intersects $\mathbf{x}_{max}(t)$

In Case 1 it is trivially true that $S^g_U = S^g$ because O does not obstruct the channel formed by $\mathbf{x}_{max}(t)$ and $\mathbf{x}_{min}(t)$. In Case 2 $\dot{s}_{\max}$ is reachable at p^g because $\mathbf{x}_{max}(t)$ connects p^s to p^g unobstructed. But it cannot immediately be guaranteed that $\dot{s}_{\min}$ is reachable through the top channel because O is obstructing that part of the reachable space. To compute $\dot{s}^g_{\min}$, note that p_{ul} is a boundary point along C_U, so consider trajectories that pass through p_{ul}. The set of reachable speeds at p_{ul} is $S_{s\to ul} = SIP(p^s, S^s, p_{ul})$ and then the set of reachable speeds at p^g from p_{ul} is $S_{ul\to g} = SIP(p_{ul}, S_{s\to ul}, p^g)$. It will be shown that $S_{ul\to g}$ is non-empty and contains the minimum reachable speed, hence $S^g_U = [\min(S_{ul\to g}), \dot{s}_{\max}]$. In case 3, O intersects $\mathbf{x}_{max}(t)$, so it cannot be guaranteed that $\dot{s}_{\max}$ is reachable. However, as will be shown, $S^g_U = S_{ul\to g}$ because any speed attainable at p^g can also be attained by a trajectory

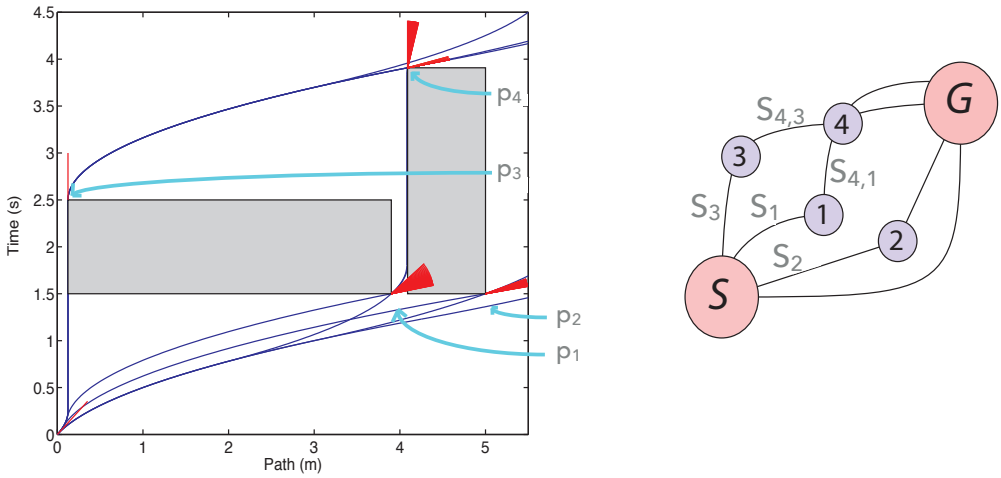


Figure 2.5: Left: Extremizing trajectories (curves) computed by SIP represent visibility between obstacle vertices with the set of reachable speeds at each vertex visualized as a wedge. Point 4 can be reached by a disjoint reachable set of speeds due to obstruction. Right: Reachability represented as a graph where nodes correspond to vertices, and edges indicate visibility.

that passes through p_{ul}. The proofs are special cases of Theorem 1 presented below. The lower speed interval S_L^g is computed similarly using the lower-right vertex.

The final reachable set is $S_f^g = S_U^g \cup S_L^g$, which may consist of up to two disjoint intervals, as seen in Figure 2.5.

2.3.3 Reachable sets with many obstacles

In problems with $\dot{s}_{\min} \geqslant 0$ and without acceleration constraints Liu and Arimoto [94] showed that optimal collision-free trajectories either connect directly to the goal state or pass tangentially along the convex edges of one or more obstacles, a property which Ó'Dúnlaing [116] showed also holds with acceleration constraints. So the algorithm searches among collision-free trajectories that pass along obstacle tangencies, which, in the axis-aligned case, reduces to the upper-left and lower-right obstacle vertices. The algorithm is first formulated for a single channel and proven correct, then generalized to arbitrary channels for the full algorithm.

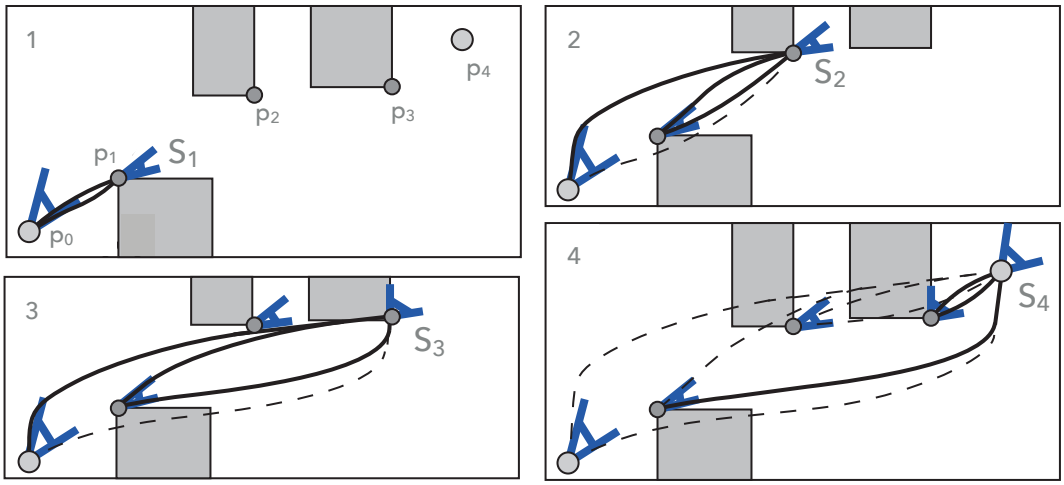


Figure 2.6: Incremental construction of reachable sets. By Theorem 1 an incremental reachable speed computed by SIP contributes to the final reachable set if its example trajectory is collision free (solid curves) but can be ignored if not (dotted curves)

Speed interval propagation through channels

For the single channel case, let $p_1, \ldots, p_{n-1}$ be the sequence of upper left and lower right obstacle vertices that bound the channel such that $t_0 \leqslant s_i \leqslant t_f$ for all i. Sort the vertices by increasing t and let $p_n = p^g$. Define the following recursive quantities for all $0 < k \leqslant n$, where $\mathbf{x}_{max}(t)$ and $\mathbf{x}_{min}(t)$ are example trajectories computed by the immediately preceding SIP call, and $Coll(\mathbf{x}(t))$ is a collision indicator function. Figure 2.6 illustrates this construction.

$$\dot{s}_{\max}^{i \to k} = \begin{cases} \max(SIP(p_i, S_i, p_k) & \text{if } S_i \neq \varnothing \text{ and} \\ & Coll(\mathbf{x}_{max}(t)) = 0 \\ -\infty & \text{otherwise,} \end{cases} \tag{2.14}$$

$$\dot{s}_{\min}^{i \to k} = \begin{cases} \min(SIP(p_i, S_i, p_k)) & \text{if } S_i \neq \varnothing \text{ and} \\ & Coll(\mathbf{x}_{min}(t)) = 0 \\ \infty & \text{otherwise,} \end{cases} \tag{2.15}$$

$$\dot{s}_{\max}^{k} = \max_{0 \leqslant i < k} \dot{s}_{\max}^{i \to k}, \tag{2.16}$$

$$\dot{s}^k_{\min} = \min_{0 \leqslant i < k} \dot{s}^{i \to k}_{\min}, \tag{2.17}$$

and

$$S_k = \begin{cases} [\dot{s}^k_{\min}, \dot{s}^k_{\max}] & \text{if } \dot{s}^k_{\min} \leqslant \dot{s}^k_{\max} \\ \varnothing & \text{otherwise.} \end{cases} \tag{2.18}$$

To connect to p^g define $\dot{s}^{i \to goal}_{\max}$ and $\dot{s}^{i \to goal}_{\min}$ similarly to Equations 2.14 & 2.15 except that the call to SIP is replaced by $GSIP$. These quantities are used to define the set of reachable speeds S^g through the channel by application of Equations 2.16, 2.17, & 2.18. The following two theorems can now be stated.

Theorem 1. *$S^g = S_n$ is the set of speeds attainable at p^g over all feasible trajectories in the channel starting at p^s.*

Proof. The proof proceeds by induction. The base case is $n = 1$, in which case there are no obstacles and by inspection $S_n = SIP(p^s, S^s, p^g) = S^g$. Now consider the inductive case with $p_n = p^g$, and compare the true set of reachable speeds S^g against S_n given by Equation 2.18. Obviously $S_n \subseteq S^g$ because each of its endpoints is defined by a feasible trajectory; the converse will now be proven.

The inductive assumption states that the theorem holds for all possible $n-2$ obstacle vertices p_i with $i = 1, \ldots, n-2$ and boundary conditions. So S_i is the set of speeds reachable from p^s for each obstacle vertex p_i for $i = 1, \ldots, n-1$. Furthermore, for $i = 1, \ldots, n-1$ denote $S_{i \to n}$ as the set of speeds reachable at p_n starting at each obstacle vertex p_i with initial speed interval S_i. Note that $\dot{s}^n_{\min}$ defined by Equation 2.17 is equivalent to the minimum of $\dot{s}^{0 \to n}_{\min}$ and $\min_{\{i=1,\ldots,n-1\}} \min(S_{i \to n})$, while $\dot{s}^n_{\max}$ defined by Equation 2.17 is the maximum of $\dot{s}^{0 \to n}_{\max}$ and $\max_{\{i=1,\ldots,n-1\}} \max(S_{i \to n})$.

Consider any feasible trajectory $\mathbf{x}(t)$ connecting p^s to p^g. The proof will show that the

terminal speed $\dot{s}^g \in S_n$ as defined by Equation 2.18. Let $\mathbf{x}_{max}(t)$ be the upper example trajectory for the connection between p^s and p_n. If $\mathbf{x}_{max}(t)$ is collision free, then $\dot{s}^{0\to n}_{\max}$ would be the maximum achievable dynamically feasible speed, and $\dot{s}^g$ would satisfy $\dot{s}^g \leqslant \dot{s}^{0\to n}_{\max}$. If on the other hand, $\mathbf{x}_{max}(t)$ is in collision with some obstacle, it is possible to construct a feasible trajectory $x_\gamma(t) = \gamma\mathbf{x}(t) + (1-\gamma)\mathbf{x}_{max}(t)$ that is a convex combination of $\mathbf{x}(t)$ and $\mathbf{x}_{max}(t)$ such that $x_\gamma(t)$ touches an obstacle vertex, say p_i. Since $x_\gamma(\cdot)$ is a feasible path starting at p_i, it is the case that $\dot{s}^g_\gamma \in S_{i\to k} \subseteq S_n$, and therefore $\dot{s}^g_\gamma \leqslant \dot{s}^n_{\max}$. Since the goal speed due to the upper bounding trajectory is at least $\dot{s}^n_{\max}$, the fact that $\mathbf{x}_\gamma(t)$ is a convex combination of $\mathbf{x}(t)$ and $\mathbf{x}_{max}(t)$ implies that goal speed due to $\mathbf{x}(t)$ is at most $\dot{s}^m_{\max}$. Using symmetry, one can similarly prove the goal speed due to $\mathbf{x}(t)$ is at least $\dot{s}^n_{\min}$. Hence $S^g \subseteq S_n$, and therefore $S^g = S_n$. By induction Theorem 1 holds for any $n \geqslant 1$. □

Theorem 2. *S^g is the set of terminal speeds reached by feasible trajectories through the channel, and the time attained at the maximum speed in S^g is precisely the duration of the minimum-time trajectory.*

Proof. The proof is similar to Theorem 1. □

2.3.4 P-time visibility graph construction

Theorems 1 & 2 imply a simple method for computing the time-optimal solution: enumerate all channels and compute the minimal time for each. The minimum over all of the channels produces the time of the optimal trajectory and its final speed; the trajectory itself is recovered by a backwards search using $RSIP$. But enumeration is intractable in general because there are potentially 2^n distinct channels for n obstacles, and obstacle configurations that yield $O(2^n)$ channels are not uncommon.

Instead the planner uses a polynomial-time algorithm that iteratively computes reachable sets over vertices, and, assuming the number of vertices for each obstacle is bounded by some constant, there are $O(n)$ vertices for n obstacles. Each iteration accumulates terminal speed intervals over all equivalent homotopy classes, and then *merges* the intervals before proceeding to the next vertex. The backwards search remains the same. Merging is key, because for any point there are $O(n)$ merged speed intervals (see Theorem 3) but there are $O(2^n)$ homotopy classes. Because merging can be done in polynomial-time, the overall algorithm can achieve $O(n^4)$ running time. The result is a visibility graph over the PT-space points from which a time-optimal trajectory can be efficiently[5] extracted.

The algorithm

For a start point p^s with set of initial speeds $S^s = \{\dot{s}^s\}$, let ε be a *resolution threshold* that allows the algorithm to discard small speed intervals that cannot be reached with confidence; for example ε may indicate the measurement uncertainty of the speedometer. Theorem 3 will use ε to aid in complexity analysis.

First, sort the upper left and lower right obstacle vertices by increasing t coordinate into the list $p_1, \ldots, p_{2n}$. Create an indexed map $R(i)$ that for each obstacle vertex point $i \in [1, 2n]$ stores a set S_i of reachable speeds at p_i annotated by a bit sequence denoting the channel of the incoming trajectory. Initialize each S_i to an empty set. *Forward* (Algorithm 1) builds the forward visibility graph by iteratively propagating sets of reachable speeds toward the goal.

[5]It is worth noting that, theoretically speaking, the $O(2^n)$ enumeration technique is more efficient than the $O(n^4)$ technique for small n; however, it quickly becomes unusable as n grows. For instance, 2^n and n^4 are equal for $n = 16$, but for $n = 20$, 2^n is already almost an order of magnitude larger than n^4. This difference is even more magnified in practice because, as shown in Table 2.1, the expected behavior of the planner is actually closer to $O(n^2)$ or $O(n^3)$.

Algorithm 1 This procedure propagates reachable speed sets forward across PT-space points in the presence of obstacles. After completion, R contains exactly the speeds reachable at each point p_i.

procedure FORWARD(R, ε)
  for $j = 1, \ldots, 2n$ **do**
    $V_j \leftarrow \varnothing$
    for $i = 0, \ldots, j-1$ **do**
      $S_i \leftarrow R(i)$
      for $[a, b] \in S_i$ **do**
        $Propagate([a, b], p_i, p_j, V_j)$
      end for
    end for
    $R(j) \leftarrow Merge(V_j, \varepsilon)$
    for $[a, b] \in R(j)$ **do**
      $PropagateGoal([a, b], p_j)$
    end for
  end for
end procedure

Algorithm 2 Computes speeds directly reachable at point j starting from point i with speed $\dot{s} \in [a, b]$, and adds them to the set V with annotation.

procedure PROPAGATE($[a, b], p_i, p_j, V$)
  $[a', b'] \leftarrow SIP(p_i, [a, b], p_j)$
  Let $\mathbf{x}_{min}(t)$ and $\mathbf{x}_{max}(t)$ be the exemplar trajectories computed by SIP
  $B_L \leftarrow Channel(\mathbf{x}_{min}(t))$
  $B_U \leftarrow Channel(\mathbf{x}_{max}(t))$
  if $B_L = B_U \neq \varnothing$ **then**
    $Insert(([a', b'], B_L), V)$
  else
    if $B_L \neq \varnothing$ **then**
      $Insert((\{a'\}, B_L), V)$
    end if
    if $B_U \neq \varnothing$ **then**
      $Insert((\{b'\}, B_U), V)$
    end if
  end if
end procedure

Algorithm 3 Extends speed intervals in S from same homotopy class and outputs the union of the reachable speeds.

```
1: procedure MERGE(S, ε)
2:     S' ← ∅
3:     while S ≠ ∅ do
4:         if ∃([a, b], B), ([a', b'], B') ∈ S such that Suffix?(B, B') then
5:             Remove ([a, b], B) and ([a', b'], B') from S
6:             S_tmp ← [a, b] ∪ [a', b']
7:             if |S_tmp| ⩾ ε then
8:                 Add (S_tmp, B') to S'
9:             end if
10:        end if
11:    end while
12:    return S'
13: end procedure
```

Of the calls made by $Forward$, $PropagateGoal$ is very similar to $Propagate$ (Algorithm 2) except $GSIP$ is used instead of SIP. $Propagate$ makes use of the subroutine $Channel(\mathbf{x}(t))$ that returns $\varnothing$ if $\mathbf{x}(t)$ collides, and otherwise outputs a bit sequence $b = (b_i, \ldots, b_j)$ where b_k indicates 0 if $x(t_k) < p_k$ in time, and is 1 otherwise; b acts as a signature for the channel. $Merge$ uses the subroutine $Suffix?(B, B')$ that returns true if and only if B' is a suffix of B.

2.3.5 Complexity analysis

The complexity of $Forward$ depends on how the number of disjoint speed intervals grows during propagation. To this end, consider the number of disjoint intervals in each S_k. A naïve bound is 2^k because there are at most 2^k distinct channels leading to k. Further analysis produces a linear bound after discarding intervals of width less than ε. Theorem 3 will use Lemma 3 to establish this bound.

Lemma 3. *Consider two initial states* $\mathbf{x} = (s, \dot{s}, t)$ *and* $\mathbf{x}' = (s, \dot{s}, t)$ *that differ only in speed with* $\dot{s} < \dot{s}'$. *Let* $p = (s, t)$ *be a PST-space point reachable from* $\mathbf{x}$ *and* $\mathbf{x}'$. *Let*

$S = SReach(x, p')$ and $S' = SReach(x', p')$ for $t' > t$ and $s' \leqslant p + (t' - t)v$. If S and S' are non-empty and disjoint, then

$$|S|/2c \geqslant \sqrt{\sqrt{|S'|/2c} - |S'|/2c} - \sqrt{|S'|/2c} \tag{2.19}$$

with $c = (t' - t)(\overline{a} - \underline{a})$ the range of reachable speeds from x at time t'. Furthermore the bounds $|S|/2c \leqslant 1/4$ and $|S'|/2c \leqslant (|S|\sqrt{2 + \sqrt{2}}/2c)^4 \approx 11.65(|S|/2c)^4$ hold.

Proof. The proof is algebraically lengthy, so only the conceptually important points are provided here. Let $R(t; p^s)$ denote the reachable set of speeds from a PT-space point p^s. Up to speed limits, $R(t'; x)$ and $R(t'; x')$ are identical except for an offset in the P and V axes (see Figure 2.7). Consider the conditions under which $|S'|$ space lies between the right boundary of $R(t'; s)$ and the line of slope $1/(t' - t)$ underneath (the lower dotted line in Figure 2.7). Some algebra shows that to obtain this difference, s' must exceed the leftmost boundary of $R(t', x)$ by at least $2c^2(\sqrt{|S'|/2c} - |S'|/2c)$, and the bound in the theorem follows from algebraic manipulations. □

Theorem 3. *Each S_i contributes $O(d)$ disjoint intervals of width at least ε to S_k, where $d = 2\log_4(-\log_2 \frac{\varepsilon}{2\dot{s}_{max}}) + 2$ and $\dot{s}_{max} = t_{max}(\ddot{s}_{max} - \ddot{s}_{min})$ bounds the range of achievable speeds.*

Proof. Let $\dot{s}_1, \ldots, \dot{s}_r$ be a sorted list of speeds each from a single disjoint interval of S_i, such that $p_i + (t_k - t_i)\dot{s}_q \geqslant p_k$ for all $q = 1, \ldots, r$. Let $S_1, \ldots, S_r$ be the sets of reachable speeds at (p_k, t_k) respectively starting at $(p_i, \dot{s}_1, t_i), \ldots, (p_i, \dot{s}_r, t_i)$. Also let $\dot{s}_{ik} = (t_k - t_i)(\ddot{s}_{\max} - \ddot{s}_{\min})$ be the range of speeds attainable from a single PT-space point after time $t_k - t_i$ has elapsed. By Lemma 3, $|S_{i+1}|/2\dot{s}_{ik} \leqslant (|S_i|\sqrt{2 + \sqrt{2}}/2\dot{s}_{ik})^4$. Because $|S_1/2\dot{s}_{ik}| \leqslant 1/4$ and $\sqrt{2 + \sqrt{2}} < 2$,

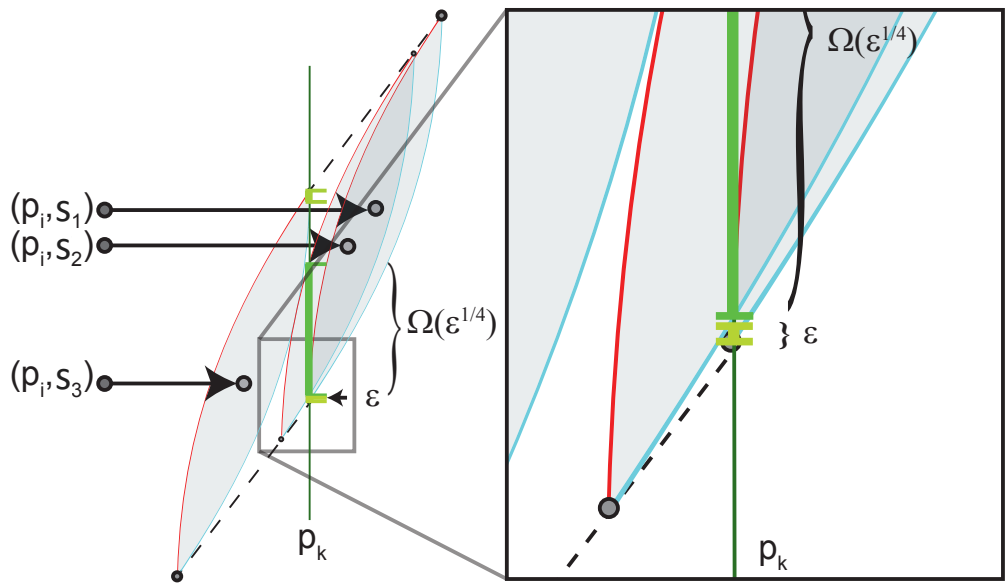


Figure 2.7: It is difficult to construct a scenario in which $d > 2$ disjoint reachable speeds continue to be disjoint after propagation. Here three initial speeds at p_i produce three disjoint speed intervals at p_k. The region denoted with ε demonstrates that the space available for non-overlapping regions is tiny even with $d = 3$.

$|S_r|/2\dot{s}_{ik} \leqslant (|S_1|/2\dot{s}_{ik})^{4^{r-1}} \leqslant (1/2)^{4^{r-1}}$. So if $|S_r| > \varepsilon$, then

$$\varepsilon/2\dot{s}_{ik} \leqslant 2^{-4^{r-1}} \tag{2.20}$$

$$\Rightarrow r \leqslant \log_4(-\log_2(\varepsilon/2\dot{s}_{max})) + 1 \tag{2.21}$$

A similar bound applies for $p_i + (t_k - t_i)\dot{s}_q \leqslant p_k$ as well. Since d is at most $2r$, this completes the proof. □

Due to the double logarithm, the size of the smallest interval shrinks rapidly as d increases (Figure 2.7). The 7th and 8th intervals are no larger than $10^{-26}\dot{s}_{max}$, which implies that, in practice, no more than 6 significant disjoint intervals may remain after each propagation. For further analysis, assume $\varepsilon > 0$, which makes d constant.

With the ε-bound assumption, the planner has $O(n^4)$ time and $O(n^2)$ space complexity in

the number of obstacle vertices. Because each prior point can contribute up to n elements to V_j at Line 7 of $Forward$, V_j has $O(n^2)$ elements. $Propagate$ is $O(j-i)$ because determining channel is $O(j-i)$ and all other steps are $O(1)$. Since $Propagate$ is called $O(n^2)$ times, steps 1-9 of $Forward$ are $O(n^4)$. Using a trie data structure indexed by the reverse of the bit vector B, $Merge$ takes time $O(j^3)$, so the n merge steps of $Forward$ take time $O(n^4)$ overall. For space complexity, each of the n vertices can have up to n disjoint intervals, giving $O(n^2)$ space complexity.

Of note is that $O(n^4)$ is a conservative bound; in practice it is extremely difficult to construct or encounter a scenario in which the number of disjoint intervals at each node is actually $O(n)$. One can typically expect the number to be bounded by some constant, which is indicated by the empirical results in §2.5. It may also be possible to extend Theorem 3 to prove this theoretically.

2.4 Extension to polygonal PT-space obstacles

Until now the assumption has been made that PT-space obstacles are axis-aligned rectangles in the PT-space plane. This simplifies the derivation of the algorithm, but is only a reasonable approximation of the obstacles if they *all* cross the agent's path perpendicularly and have roughly rectangular geometries that are axis-aligned with the path. To allow for a more general class of crossing paths, the algorithm can be extended in a straightforward way to deal with general polygonal PT-space obstacles as shown in Figure 2.8. This section presents algorithmic extensions that allow propagation in the presence of general polygonal obstacles, as well as an obstacle construction routine that builds general polygonal PT-space obstacles given general convex polygonal workspace agent models.

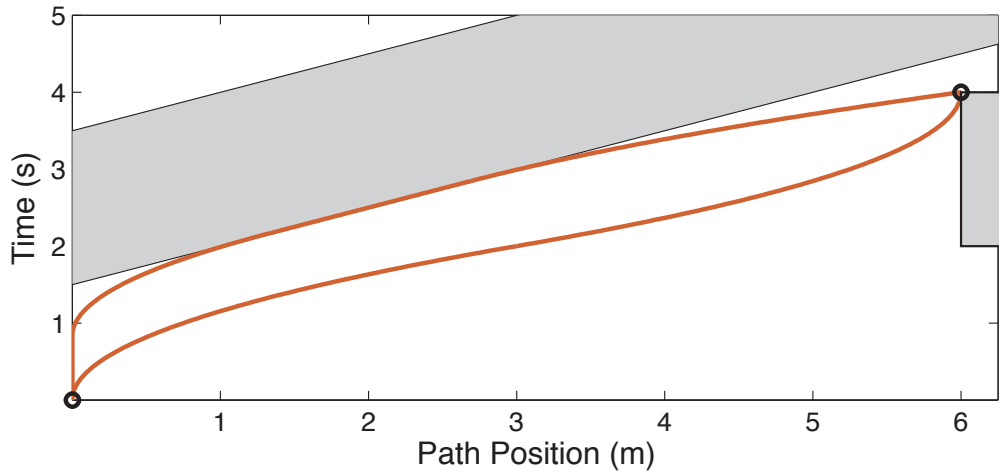


Figure 2.8: Upper and lower feasible trajectories (curves) at the upper right point, given an initial speed range of $[0, 0]m/s$ and acceleration bounds of $\pm 1.5m/s^2$.

2.4.1 Algorithm extension

To begin, again consider the cases from §2.3.2 for computing reachable speeds in the presence of a single obstacle, and let $\mathbf{x}(t)$ be one of the example trajectories computed by SIP. To deal with general polygonal obstacles, modify the cases to account for intersection with *diagonal* obstacle edges in addition to horizontal and vertical:

1. If $\mathbf{x}(t)$ is collision-free, then it is optimal and SIP operates as before.

2. If $\mathbf{x}(t)$ collides with a vertical, horizontal, or negatively sloping PT-space obstacle edge, SIP outputs nothing.

3. If $\mathbf{x}(t)$ collides with a diagonal edge with positive slope, SIP uses Algorithm 4, defined below, in order to propagate along one or more edges.

As before, Case 1 is obviously correct. For Case 2 it was shown that either there is no feasible trajectory or there is an extremizing trajectory that passes through one of the endpoints of that PT-space obstacle edge. In this latter case, the planner will find the correct

trajectory by propagating from p_1 to that vertex and then from that vertex to p_2, and hence this case reduces to previous treatment. Case 3 requires further examination because it may be possible for the trajectory to follow the diagonal edge tangentially (Figure 2.8), which means the planner must consider point/edge propagations. It may also be possible (although unlikely) for the trajectory to touch *several* diagonal edges, so it must also consider edge/edge propagations.

To compute these types of propagations, the planner invokes a recursive procedure that makes use of the following subroutines, each of which compute example trajectories between various primitives in the absence of obstacles:

1. $PointToEdge(p, S, e)$, which accepts an initial point p with speed interval S and edge e, and builds a time-minimizing trajectory segment $p \rightarrow e$ that terminates at a point of tangency on e without crossing.

2. $EdgeToPoint(e, p_e, p_2)$, which accepts an initial point p_e along edge e and builds a speed extremizing trajectory segment $e \rightarrow p_2$ that does not cross e.

3. $EdgeToEdge(p_{e_1}, e_1, e_2)$, which accepts an initial point p_{e_1} along edge e_1 and builds a trajectory segment $p_1 \rightarrow e_2$ that terminates at a point of tangency to e_2 without crossing e_1 or e_2. The trajectory is constructed such that the time before e_1 is departed is minimized.

A time-minimizing trajectory is used to propagate to edges because the feasible speed of a trajectory traveling along an edge is exactly the slope of the edge, hence speed is fixed, leaving time as the free variable. It should be clear that time-minimizing trajectories are correct for propagating along edges because that maximizes the visibility of the edge. Algorithm 4,

which computes the example trajectory, is given generically below as extremizing because, as before, the algorithms for maximizing/minimizing terminal speeds are essentially symmetric.

Algorithm 4 Compute the speed-extremizing trajectory at an edge e from a PT-space point p_1 with initial speed interval S_1 and attempt to propagate recursively forward to p_2

```
1: procedure ExtremalSpeed_Diagonal(p_1, S_1, e, p_2)
2:     T_1 ← PointToEdgeRecursive(p, S, e)
3:     if T_1 = ∅ then
4:         return ∅
5:     end if
6:     Let x_f be the terminal state of T_1
7:     T_2 ← EdgeToPointRecursive(e, x_f, p)
8:     if T_2 = ∅ then
9:         return ∅
10:    end if
11:    return Concatenate(T_1, T_2)
12: end procedure
```

Algorithm 4 calls *PointToEdgeRecursive* that computes a time-minimizing trajectory from p_1 to the diagonal edge e, and *EdgeToPointRecursive* that computes a speed-extremizing trajectory from e to p_2. Both may call *EdgeToEdge* that computes a time-minimizing trajectory between two edges e_1 and e_2. A definition for *PointToEdge* is given in Algorithm 5. *EdgeToEdge* and *EdgeToPoint* are similar.

Complexity analysis

The complexity class of *Forward* does not change with the addition of edge propagation; however, it does add a coefficient to the bounding polynomial. If a diagonal edge is hit, two or more additional trajectory generation and collision detection routines are called. In all, *Propagate* becomes $O(kn)$ where k is the number of diagonal edges hit during the construction. In practice k is a small constant, but in the worst case it may be $O(n)$. The resulting complexity is now $O(n^5)$.

Algorithm 5 Recursively compute a time-minimizing trajectory from p_1 to edge e given initial speed range S

```
1: procedure PointToEdge(p, S, e)
2:     T_1 ← PointToEdge(p_1, t_1, V_1, e)
3:     if T_1 = ∅ then
4:         return ∅
5:     end if
6:     e' ← InitialCollidingEdge(T_1)
7:     if e' = ∅ then
8:         return T_1
9:     end if
10:    T_1 ← PointToEdge(p_1, t_1 e', V_1)
11:    if T_1 = ∅ then
12:        return ∅
13:    end if
14:    T_2 ← EdgeToEdge(FinalPoint(T_1), e, e')
15:    if T_2 = ∅ then
16:        return ∅
17:    end if
18:    return Concatenate(T_1, T_2)
19: end procedure
```

Edge propagation could also be handled by explicitly treating edges as nodes in the visibility graph, which would lead to a complexity of $O((2n)^4)$. Informal testing showed the recursive method to do fewer edge-related propagations in practice.

2.4.2 General PT-space obstacle construction

Computing the exact boundaries of PT-space obstacles is challenging because they may be arbitrarily curved. Geometrically, the problem is that of computing swept volumes [6, 15]. For arbitrary motions and shapes, computation of swept volumes to arbitrary precision cannot be done in real-time, but many techniques have been developed for efficient approximations [81, 141, 156]. For building PT-space obstacles, a simple approximation would discretize PT-space space with resolution τ and test each cell for collision, but this requires 2D discretization resulting in $O(T|P|\tau^{-2})$ cells. Instead, the approach presented here dis-

cretizes only the time dimension and *analytically* computes forbidden intervals in the path dimension. This leads to $O(T\tau^{-1})$ cells, so is more efficient while still being resolution complete (Figure 2.9).

The algorithm is defined for a single workspace obstacle $O_{\mathbb{W}}$. Consider a uniform grid on the time dimension $0, \tau, 2\tau, \ldots, t_{max}$. The algorithm computes the leftmost and rightmost extent of PT-space obstacle O_i within each horizontal strip $t \in [k\tau, (k+1)\tau]$ in the PT-space plane, resulting in a conservative *forbidden rectangle* $[a_k, b_k] \times [k\tau, (k+1)\tau]$. Figure 2.9c shows PT-space obstacles constructed at progressively finer resolutions. The rectangles are then wrapped with a polygon to smooth their jagged edges as shown in Figure 2.9d. This procedure is defined in Algorithm 6 and relies on two key functions: $SweptVolume$ and $ComputeForbiddenInterval$.

Algorithm 6 Computes the PT-space obstacle corresponding to the workspace obstacle $O_{\mathbb{W}}$ following a path P_O with a trajectory bounded by $\mathbf{x}_{min}(t)$ and $\mathbf{x}_{max}(t)$ for time discretization τ with respect to an agent model A following a path P_A

procedure BUILDPTOBSTACLE($O, P_O, \tau, t_{max}, x_{\min}(\cdot), x_{\max}(\cdot)$)
  $O \leftarrow \varnothing$
  for $t = 0, \tau, 2\tau, \ldots, t_{max} - \tau$ **do**
    $\mathcal{V}_O \leftarrow SweptVolume(O, P_O, x_{\min}(t), x_{\max}(t+\tau))$
    $[a, b] \leftarrow ComputeForbiddenInterval(\mathcal{V}_O, A, P_A)$
    $O \leftarrow O \cup [a, b] \times [t, t+\tau]$
  end for
  return $BoundingPolygon(O)$
end procedure

SweptVolume This function computes a conservative approximation to the region of workspace $\mathcal{V}_O \subseteq \mathbb{W}$ swept out by obstacle $O_{\mathbb{W}}$ between times $k\tau$ and $(k+1)\tau$ as it follows a piecewise linear path P_O. Let $\underline{W}$ and $\overline{W}$ be the workspace regions occupied by $O_{\mathbb{W}}$ at the minimum and maximum possible path extents at times $k\tau$ and $(k+1)\tau$. When $\underline{W}$ and $\overline{W}$ are on the same piece of the path P_O, $\mathcal{V}_O = Conv(\underline{W}, \overline{W})$ where $Conv$ denotes the

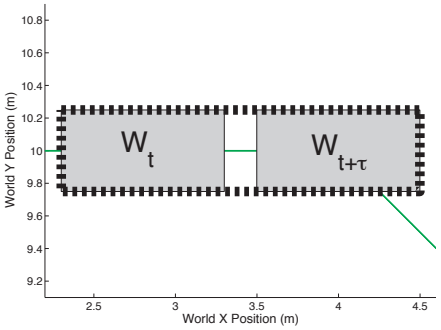
convex hull (Figure 2.9a).

When they lie on adjacent segments, additional models W_{θ_1} and W_{θ_2} are computed at the path segment junction, oriented at angles θ_1 of the previous segment and θ_2 of the next. This is necessary to account for the intermediate postures of the rotation, because each vertex of O sweeps out an arc centered at the junction. Tangents of the arc endpoints are computed for each vertex, and the intersection point of the tangents is added to a set of points $\mathcal{O}_\theta$. When $\mathcal{O}_\theta$ is computed, let $W_\theta = Conv(W_{\theta_1}, \mathcal{S}, W_{\theta_2})$, as in Figure 2.9b. $\mathcal{V}_O$ is then the union of of $Conv(\underline{W}, W_{\theta_1})$, W_θ, and $Conv(W_{\theta_2}, \overline{W})$. This construction generalizes easily to multiple traversed path segments.

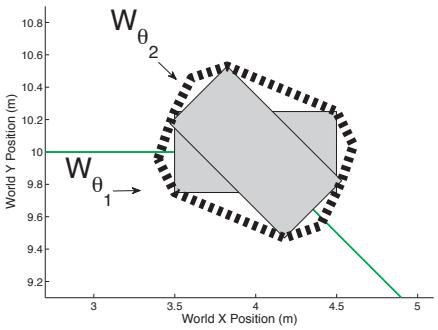
ComputeForbiddenInterval This function computes the points a_k and b_k at which the agent A following a piecewise linear path P_A first comes into contact with a swept world space obstacle $\mathcal{V}_O$, and last leaves contact with $\mathcal{V}_O$. At each extremum, it is either the case that a vertex of A lies on an edge of $\mathcal{V}_O$, or that a vertex of $\mathcal{V}_O$ lies on an edge of A. A search of all vertex-edge combinations will then find all such extrema along each line segment of P_A, and a_k and b_k are output as the minimum and maximum of these extrema[6].

The *BoundingPolygon* call at the end of the function simply wraps the forbidden rectangles with a polygon and discards interior vertices. This can significantly reduce the number PT-space obstacle vertices, yielding faster visibility graph construction.

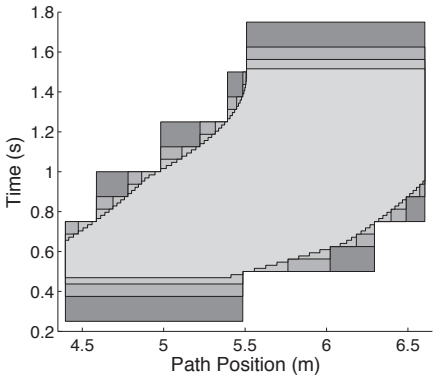
[6]This treatment assumes the system is not initialized in a collision state



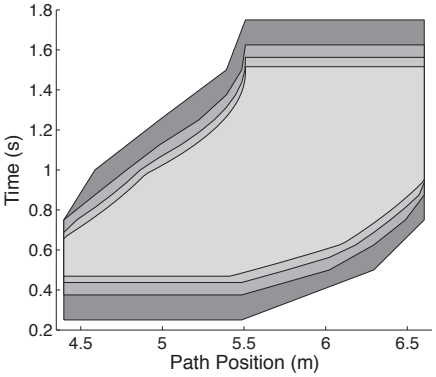
(a) Convex hull (dashed) of $\underline{W}$ and $\overline{W}$ on same path segment.



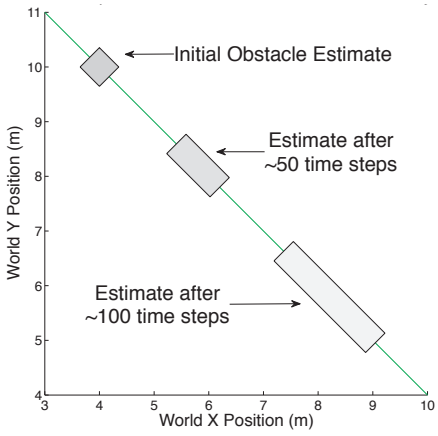
(b) Convex hull (dashed) of W_{θ_1} and W_{θ_2} for path segment junction.



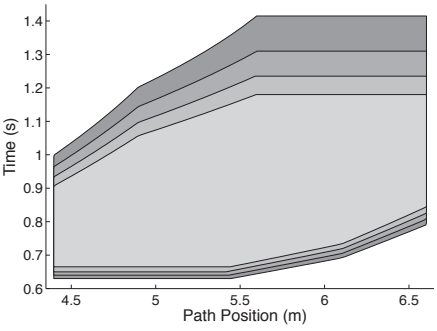
(c) Forbidden region approximations at successively finer resolutions.



(d) The generated PT-space obstacle wraps the jagged forbidden regions with a polygon.



(e) Example of growing uncertainty in an obstacle's swept volume as time progresses.



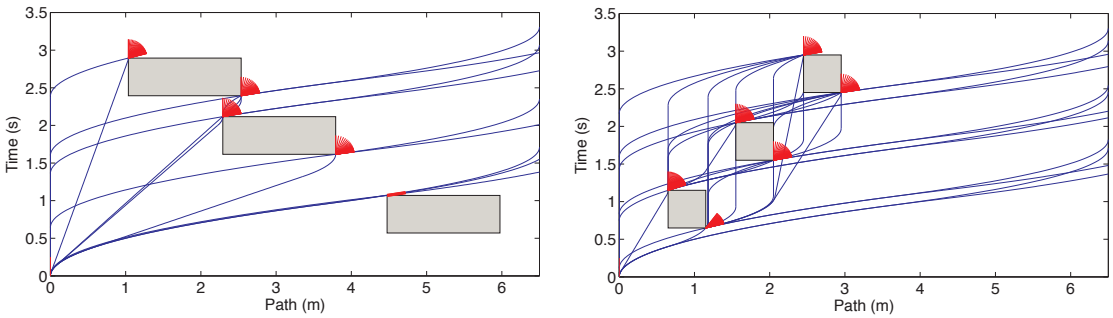
(f) PT-space obstacles (computed at a fixed resolution) for increasing obstacle behavior uncertainty.

Figure 2.9: Illustrating PT-space obstacle construction.

2.4.3 Complexity analysis

Let $k = T\tau^{-1}$ be the number of grid points. Assuming $|A|$ and $|O|$ are bounded by m, the average case running time for $SweptVolume$ is $O(m \log m |P_O| k^{-1})$ and the average case running time for $ComputeForbiddenInterval$ is $O(|P_A| m^2)$. $BoundingPolygon$ is $O(k)$. Overall, complexity is $O(|P_A| m^2 k + |P_O| m \log m)$.

2.5 Simulation tests



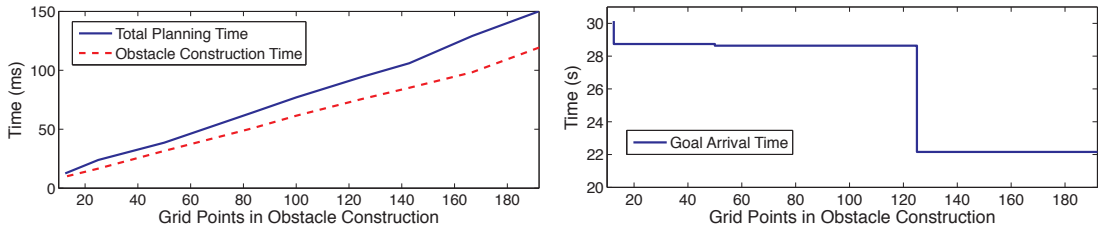
(a) (b)

Figure 2.10: Left: Three obstacles are arranged randomly, with the reachable sets of speeds (wedges) at each upper-left and lower-right vertex. Right: Three obstacles are arranged in a staircase configuration, with the reachable sets of speeds (wedges) at each upper-left and lower-right vertex.

An implementation of the graph construction and trajectory retrieval algorithms for axis-aligned obstacles was implemented and evaluated against random (Figure 2.10a) and pathological staircase (Figure 2.10b) obstacle configurations. In both cases $O(n^2)$ speed

Table 2.1: Run Times for Pathological Configurations of Axis-Aligned Obstacles

obstacles	1	5	10	15	~Growth
Random 1	0.233ms	1.66ms	4.21ms	7.98ms	$O(0.0002n^3)$
Random 2	0.206ms	1.36ms	3.59ms	5.47ms	$O(0.012n^2)$
Random 3	0.055ms	1.08ms	3.09ms	9.50ms	$O(0.007n^3)$
Staircase	0.244ms	2.77ms	16.6ms	55.9ms	$O(0.021n^3)$



(a) (b)

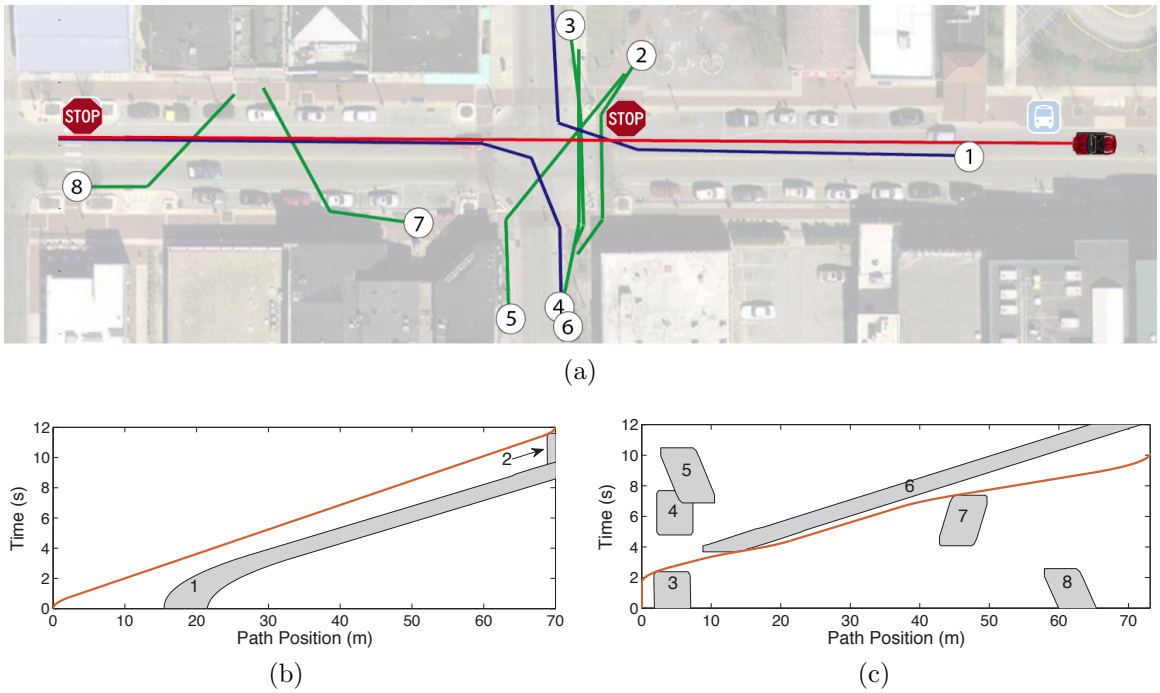
Figure 2.11: (a) Run times according to increasingly fine τ discretization. (b) Trade-off between τ discretization and time optimality.

intervals must be propagated, but in the staircase configuration the visibility graph is almost complete while in the random configurations only 1/3 or 1/5 of possible edges are present. Running times on a 2.4GHz PC for $n = 1, \ldots, 15$ axis-aligned obstacles are presented in Table 2.1. In all cases the data were best interpolated by 2^{nd}- or 3^{rd}-degree polynomials with small leading coefficients, coming in below the theoretically expected $O(n^4)$.

Figure 2.11 shows empirical performance of general polygonal PT-space obstacle construction and visibility graph construction for a scenario similar to that in Figure 2.12. Results are averages of 10 runs on a single core of a 2.3GHz PC. In Figure 2.11a the planner is run with varying levels of discretization in PT-space obstacle construction, showing a roughly linear relationship. Figure 2.11b shows how discretization affects the optimal goal arrival time. At coarse discretizations, narrow homotopy channels are occluded and the planner goes around them. At finer discretizations, these channels open up and the planner finds faster routes.

Real world scenarios

In this section results are given for simulations of two real world scenarios.



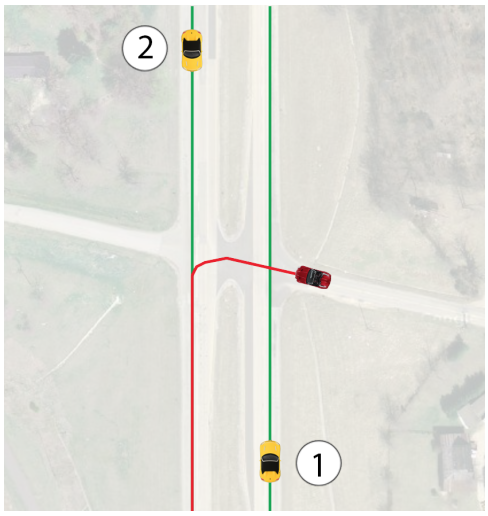
(a) (b) (c)

Figure 2.12: (a) An urban driving scenario involving pedestrians and bicyclists. The car position, pedestrian and bicycle positions, and their paths are overlaid on the map. (b) PT obstacles of the stage 1 problem leading to the first stop sign, with the time-optimal trajectory shown in red. (c) PT obstacles of the stage 2 problem leading to the second stop sign. Imagery ©2012 DigitalGlobe, GeoEye, IndianaMap Framework Data, USDA Farm Service Agency

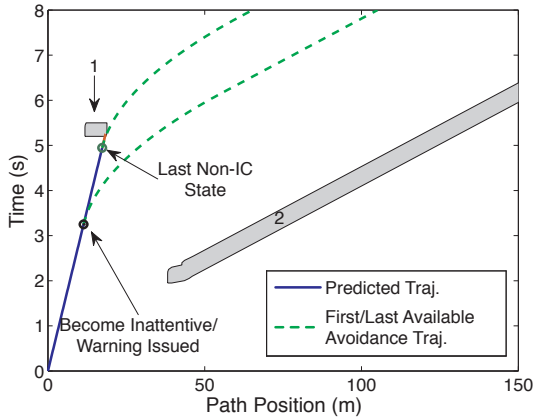
Urban driving The planner is given a simulated urban scenario modeled in Figure 2.12. The car travels along a downtown section of Kirkwood Avenue in Bloomington, Indiana. Bicyclists often share the road and pedestrian traffic is heavy, both at and away from crosswalks. The problem has two stages: 1) reaching the first stop sign, then 2) reaching a second stop sign. Acceleration bounds are $[-10, 8] m/s^2$ and speed bounds are $[0, 13.4] m/s$.

Figure 2.12b shows the stage 1 trajectory. The car must avoid a bicycle (obstacle 1) moving in front of it. The bicycle accelerates from an initial stop, causing its PT obstacle to be curved initially, and then turns off the road after the stop, so its PT obstacle ends. Near the stop sign the car must avoid a pedestrian (obstacle 2) that cuts in front of the crosswalk.

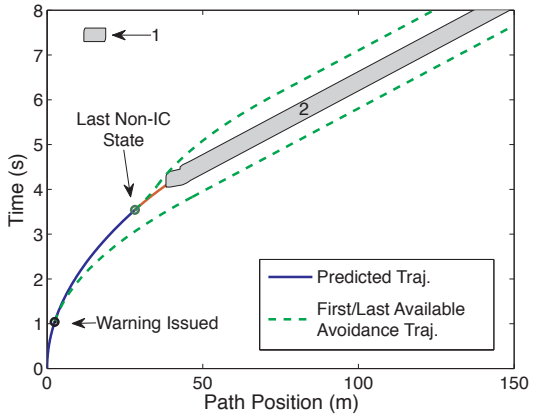
Figure 2.12c shows the stage 2 trajectory. The car must now avoid pedestrians in the



(a)



(b)



(c)

Figure 2.13: Illustration of a collision warning application. (a) An inattentive driver (rectangle) is merging onto the southbound lane of a rural highway, but fails to notice oncoming vehicles (numbered). (b) Given a lead time $t_r = 2.5s$, an inevitable collision state is detected within t_r after the driver become inattentive and a collision warning is issued. (c) An inattentive driver accelerates too slowly while merging and a warning is issued at t_r from the first IC state.

crosswalk, as well as another bicycle (obstacle 6) that turns into the car's lane. The optimal plan has the car accelerate out in front of the bicycle, then decelerate slightly to avoid a pedestrian (obstacle 7) before going on to the final position.

Imminent collision detection The planner can also be applied to collision warning systems for inattentive drivers. Suppose such a system can detect driver inattention and has a reaction time parameter t_r sufficient for a driver to perceive and respond to a warning, but not so long as to generate unnecessary false positives. The planner can be called repeatedly to verify that a feasible trajectory exists[7], assuming the driver continues his/her current behavior up to time t_r. If not, then the driver is about to enter an inevitable collision state (ICS) [49], and a warning is issued. In order to do so, the system first collision checks the driver's predicted trajectory T_p up to time t_r. If a collision is found, a warning is issued. Otherwise, the planner attempts to find a feasible trajectory starting from the final state of T_p. If none is found, a warning is issued.

Consider a rural highway intersection scenario. The driver attempts to merge south onto State Road 37 in Indiana (Figure 2.13a) after crossing two northbound lanes of traffic. The driver incorrectly judges the speed of a northbound vehicle and begins the merge too slowly to cross safely. The planner detects an ICS within t_r and a warning is issued to the driver at the point marked in Figure 2.13b. In a second example with different initial conditions (Figure 2.13c), the driver incorrectly judges the speed of the southbound vehicle and attempts to merge too slowly. The warning indicates that the driver should either slow down or stop at the median, or accelerate ahead of the oncoming vehicle.

2.6 Conclusion

This chapter presented a complete, optimal speed planner that operates in the presence of moving obstacles. It allows arbitrary polygonal models and agent trajectories and was shown capable of planning time-optimal speed profiles in cluttered scenarios and detecting

[7] Chapter 3 provides a rigorous framework for such use cases.

inevitable collision states. Simulation tests suggest that the planning system is fast enough for real-time navigation among many dynamic obstacles.

On its own, this planner is of limited use for real-world situations: the requirement that the world evolve deterministically and the assumption that all agent actions are independent from ego actions is only realistic in so-called *asteroid avoidance* problems [118] and not in the presence of other intelligent agents. As will be shown in the following chapters, however, the properties provided by this algorithm, especially its completeness and efficiency, can be exploited as part of a broader framework to robustly solve navigation in more general multi-agent systems.

Chapter 3

The constrained interference minimization principle

Synopsis: *This chapter introduces a general constrained optimization framework and demonstrates it with the use case of semi-autonomous automobile safety systems. Such systems must only interfere with driver control when they are extremely confident that it is the right thing to to. In this use case, the constrained interference minimization principle allows the decision to interfere to be made probabilistically under a stochastic optimal control framework, thus ensuring a level of confidence in maintaining safety, while also allowing the system to minimize the magnitude of interference required to maintain that safety confidence.*

3.1 Introduction

This chapter introduces a framework for computing constrained interference minimization problems. These types of problems occur frequently in shared autonomy and redundant control robotic systems. For example, a common architecture for such systems is to provide a backup, or emergency, controller that takes over from, or interferes with, a nominal controller when the robot enters into states that the nominal controller is not designed to handle. Such systems need to be able to decide confidently that interference is the proper action, and when they do interfere they need to minimize the magnitude of that interference such that the interruption from nominal control is minimized.

The principle presented in this chapter builds on preliminary work presented in Johnson et al. [68]. It formulates the interference decision problem probabilistically using one or more given constraints to define the decision boundary. If the decision boundary is crossed, i.e., the constraints are violated, then the problem of moving the system state back across the boundary is formulated as a stochastic optimal control problem that minimizes the distance in control space necessary to perform that move. §3.2 introduces the necessary background from stochastic optimal control, and then formulates the general constrained interference minimization principle. §3.4 presents a concrete example of a framework built with the principle called the *safety-constrained interference minimization principle* (SCIMP). §3.5 defines real-world scenarios for which SCIMP formulations are provided and then presents evaluations of SCIMP on those scenarios. Finally, §3.6 provides summary and conclusions.

3.2 Stochastic optimal control

This section briefly introduces relevant ideas and notations[1] from stochastic optimal control that will be used in §3.3 to present the constrained interference minimization principle. First the general solution to the control problem is presented, followed by efficient approximating solutions that are exploited by the constrained interference minimization principle.

In stochastic optimal control, the goal is to find a control trajectory that drives the evolution of a stochastic process in a way that minimizes the expected value of some cost function (equivalently, the problem can be formulated to maximize the expected value of some reward function). Consider the following stochastic dynamical system:

$$d\hat{\mathbf{x}} = f(t, \mathbf{x}_t, \mathbf{u}_t)dt + d\xi \tag{3.1}$$

where $\hat{\mathbf{x}}$ is a collection of estimated state variables, $\mathbf{x}$ is the true and unobservable state, $\mathbf{u}$ is a control, and $d\xi$ denotes some stochastic process. Note that the evolution of the system depends on an unobservable state $\mathbf{x}$, but the value of $\mathbf{u}$ is observable and the value of $\hat{\mathbf{x}}$ can be estimated, and these values can be used to define a belief distribution for the current state that can be updated recursively using Bayes rule:

$$p(\mathbf{x}_t \mid \hat{\mathbf{x}}_{0:t}, \mathbf{u}_{0:t}) \propto p(\hat{\mathbf{x}}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \hat{\mathbf{x}}_{0:t-dt}, \mathbf{u}_{0:t}) \tag{3.2}$$

where $p(\hat{\mathbf{x}}_t \mid \mathbf{x}_t)$, typically referred to as the *measurement probability*, is a distribution of

[1] In this work, notation conventions are mostly taken from Kappen [76].

known form. Thus, the evolution of the belief depends on the estimated and observed variables $\hat{\mathbf{x}}$ and $\mathbf{u}$, which means an optimal control trajectory must manage the dual problem of maintaining progress toward the goal while also maintaining an acceptable belief distribution. To facilitate this, the state space can be augmented with a set of parameters θ_t that are sufficient statistics for $p(\mathbf{x}_t)$. For convenience, let $\mathbf{z}_t = (\hat{\mathbf{x}}_t, \theta_t)$. The goal of the control problem is then to determine a control policy $\pi = (\mathbf{u}_0, \ldots, \mathbf{u}_{T-1})$ that defines the optimal action for each belief state, where for each $t \in [0, T]$ the optimal control is given as $\mathbf{u}_t^\star = \pi(\mathbf{z}_{t-1})$.[2]

For the given system, the expected cost $\hat{C}(\cdot)$ of a control sequence $\mathbf{u}_{t:T}$ is defined as an expectation (denoted here with angle brackets) over all stochastic trajectories beginning with an initial belief $\mathbf{z}_t = (\hat{\mathbf{x}}_t, \theta_t)$ and terminating in $\mathbf{z}_T = (\hat{\mathbf{x}}_T, \theta_T)$:

$$\hat{C}(\mathbf{z}_t, \mathbf{u}_{t:T}) = \left\langle \phi_T + \int_t^T R(\tau, \hat{\mathbf{x}}_\tau, \mathbf{u}_\tau)\, d\tau \right\rangle_{\mathbf{z}_t} \tag{3.3}$$

where ϕ_T is a terminal cost, R an immediate cost at time τ, and the subscript $\mathbf{z}_t$ indicates that the expectation is taken over all stochastic trajectories originating from $\mathbf{z}_t$. The task is to find $\mathbf{u}_{t:T}$ such that $\hat{C}(\cdot)$ is minimized. By defining a minimum cost-to-go function $J(\cdot)$ the problem can be formulated as a Bellman recursion:

$$J(t, \mathbf{z}_t) = \min_{\mathbf{u}_t} \hat{C}(\mathbf{z}_t, \mathbf{u}_{t:T}) = \min_{\mathbf{u}_t} \left(R(t, \mathbf{z}_t, \mathbf{u}_t) + \left\langle J(t + dt, \mathbf{z}_{t+dt}) \right\rangle_{\mathbf{z}_t} \right) \tag{3.4}$$

[2]Nominally, $\mathbf{u}_t$ is dependent on the entire history, as in Equation 3.2; however, Åström [13] showed that a belief distribution provides a sufficient statistic for the history of Markov processes.

Applying a Taylor expansion to the J term on the right-hand side and exploiting linearity of expectation yields:

$$\begin{aligned}\left\langle J(t+dt, \mathbf{z}_{t+dt})\right\rangle_{\mathbf{u}_t} &= J(t, \mathbf{z}_t) + dt\partial_t J(t, \mathbf{z}_t) + \left\langle d\mathbf{z}\right\rangle_{\mathbf{z}_t}\partial_{\mathbf{z}} J(t, \mathbf{z}_t) \\ &\quad + \frac{1}{2}\left\langle d\mathbf{z}^2\right\rangle_{\mathbf{z}_t}\partial^2_{\mathbf{z}} J(t, \mathbf{z}_t) \qquad (3.5)\end{aligned}$$

$$\left\langle d\mathbf{z}\right\rangle_{\mathbf{z}_t} = f(t, \mathbf{z}_t, \mathbf{u}_t)dt \qquad (3.6)$$

$$\left\langle d\mathbf{z}^2\right\rangle_{\mathbf{z}_t} = \nu(t, \mathbf{z}_t, \mathbf{u}_t)dt \qquad (3.7)$$

Substituting the above into Equation 3.4 and simplifying yields the *stochastic Hamilton-Jacobi-Bellman equation*[3] with boundary condition $J(T, \mathbf{z}_T) = \phi_T$:

$$-\partial_t J(t, \mathbf{z}_t) = \min_{\mathbf{u}_t}\left(R(t, \mathbf{z}_t, \mathbf{u}_t) + f(t, \mathbf{z}_t, \mathbf{u}_t)\partial_{\mathbf{z}} J(t, \mathbf{z}) + \frac{1}{2}\nu(t, \mathbf{z}_t, \mathbf{u}_t)\partial^2_{\mathbf{z}} J(t, \mathbf{z})\right) \qquad (3.8)$$

The solution of Equation 3.8 provides a necessary and sufficient condition for the optimal control trajectory [22]. As attractive as this general form is, it suffers in practice from being extremely difficult to compute and generally intractable [95]. Worse, [120] showed that the problem likely has no efficient online implementation even when arbitrary offline computation is allowed. Despite this, there are important classes of problems for which efficient exact solutions are possible, which are detailed in the next sections.

[3]The Hamilton-Jacobi-Bellman equation is the continuous-time counterpart to the discrete-time Bellman equation, which is more familiar to economics and decision process theory. The continuous variant is used here simply because it is common in optimal control formulations. Equivalent results hold for the discrete-time variant.

3.2.1 The separation principle and certainty equivalence

Given the general intractability of Equation 3.8, it is often necessary to formulate problems such that they can be decomposed into tractable sub-problems. A natural way to accomplish this decomposition is to separate the problems of observation and control. To introduce this, define a deterministic cost-to-go function $C(\cdot)$ as:

$$C(\mathbf{x}_t, \mathbf{u}_{t:T}) = \phi_T + \int_t^T R(\tau, \mathbf{x}_\tau, \mathbf{u}_\tau)\, d\tau \tag{3.9}$$

Equation 3.3 can then be written:

$$\hat{C}(\mathbf{z}_t, \mathbf{u}_{t:T}) = \langle C(\hat{\mathbf{x}}_t, \mathbf{u}_t) \rangle_{\mathbf{z}_t} \tag{3.10}$$

As noted in the previous section, the optimal general solution to minimizing Equation 3.10, Equation 3.8, is often difficult, or impossible, to solve, due in part to the need to compute an expectation over stochastic trajectories. But in some cases the expected cost over stochastic trajectories is equal to the deterministic cost over the expected trajectory. In other words, the stochastic expectation in Equation 3.10 can sometimes simply be dropped and the state estimate $\hat{\mathbf{x}}$ used directly. Systems for which this applies can be solved optimally by designing an optimal state estimator and feeding the estimate into an optimal deterministic controller. Such systems are said to adhere to the *separation principle* (SP), because they allow the problem to be separated into estimation and control problems. As detailed by O'Reilly [117], Bay [16], and Georgiou and Lindquist [51], SP has been proven

to hold for linear, time-invariant (LTI) systems (provided both estimator and controller are optimal and stable), which makes them popular in practice as system models.

For a certain class of LTI systems, a stronger, and more constructive, property can be shown to hold. For LTI systems with quadratic costs and only additive noise, the optimal stochastic controller is exactly the optimal deterministic controller given the mean state of an optimal observer. Such problems are said to exhibit *certainty equivalence* (CE), which is a stricter form of the separation principle.[4] One important class of these problems is the class of *linear-quadratic-gaussian* (LQG) problems. Consider a system with the following dynamics:

$$d\mathbf{x} = (\mathbf{x} + \mathbf{u})dt + d\xi \tag{3.11}$$

$$\hat{\mathbf{x}} = \mathbf{x} + \eta \tag{3.12}$$

where η is additive Gaussian noise. For simplicity, assume terminal cost $\phi_T = 0$ with the cost function for the fully observable problem quadratic and of the form:

$$C(\mathbf{x}_t, \mathbf{u}_{t:T}) = \frac{1}{2} \left\langle \sum_{\tau=t}^{T} (\mathbf{u}_\tau^2 + \mathbf{x}_\tau^2) \right\rangle_{\mathbf{x}_t} \tag{3.13}$$

For the partially observable problem, because state dynamics are linear (Equation 3.11) and observation noise is additive (Equation 3.12), a Gaussian belief distribution $\mathbf{z}_t = p(\mathbf{x}_t \mid \hat{\mathbf{x}}_{0:t}, \mathbf{u}_{0:t}) = \mathcal{N}(\mathbf{x}_t \mid \mu_t, \sigma_t^2)$ for $\mathbf{x}_t$ can be computed analytically using a Kalman filter. An explicit belief

[4] Unlike CE, the optimal controller satisfying the separation principle is not necessarily the same controller that is optimal for the same problem without uncertainty [117].

distribution then allows the expectations to be computed for the partially observable problem, and, as shown in [76], the associated cost function can be rewritten as an integration over state hypotheses:

$$\begin{aligned}\hat{C}(\hat{\mathbf{x}}_{t:T}, \mathbf{u}_{t:T}) &= \int_{\mathbf{z}_t} C(\mathbf{x}_t, \mathbf{u}_{t:T})\, p(\mathbf{x}_t) \\ &= \frac{1}{2}\sum_{\tau=t}^{T} \mathbf{u}_\tau^2 + \frac{1}{2}\sum_{\tau=t}^{T} \left\langle \mathbf{x}_\tau^2 \right\rangle_{\mu_t, \sigma_t} \\ &= \frac{1}{2}\sum_{\tau=t}^{T} \mathbf{u}_\tau^2 + \frac{1}{2}\sum_{\tau=t}^{T} (\mu_\tau^2 + \sigma_t^2) \\ &= C(\mu_t, \mathbf{u}_{t:T}) + \frac{1}{2}(T-t)\sigma_t^2\end{aligned} \tag{3.14}$$

Notice that the second term in the final line of Equation 3.14 is independent of $\mathbf{u}$. This means that minimizing the deterministic function $C(\cdot)$ with $\mathbf{x} \leftarrow \mu$ is equivalent to minimizing the expectation $\hat{C}(\cdot)$.

Of course, a guarantee of optimality is not a guarantee of good behavior. As shown by Simpkins et al. [132], CE controllers (and any controller designed under the separation principle) can demonstrate severe instability in practice, particularly when the approximations used to guarantee their formulation are not well justified. Furthermore, because the problem is modeled such that the policy is independent of the variance in the noise (Equation 3.12), the controller has no capability to choose controls that reduce the variance, which is particularly problematic for *Interception* problems, where small variance in the terminal state is crucial [70]. To attempt to guide the controller to low variance states, Simpkins et al. [132] propose modifying the cost function to penalize large uncertainties, which produces a lazy adaptive controller that performs exploration as needed. Unfortunately, no comparison to

other adaptive techniques is provided, so the practical advantages are unclear.

A partial separation of inference and control for problems that do not exhibit CE was proposed by Witsenhausen [165], who stated that the independence of the inference and control computations can be asserted provided that the optimal control function is derived based on the conditional PDF of the state estimate. However, no formal guidance for control function derivation was given. In practice, the use of CE controllers is often justified empirically, or for the simple reason that no other usable control formulation can be found. However, as demonstrated by Witsenhausen [163] and Mitter and Sahai [106], one must always be aware in such circumstances that the CE assumption may lead to distinctly sub-optimal performance.

3.2.2 Rollout approximations

Both the SP and the CE properties are very powerful, but in many applications, particularly robotics control, the assumptions required to employ them simply are not justifiable. In such cases, approximation techniques are often employed that gain flexibility and tractability by relaxing the optimality requirement. Suppose the optimal cost-to-go function (Equation 3.4) is replaced with a tractable heuristic function $H(\cdot)$ that approximates $J(\cdot)$:

$$J(t, \mathbf{z}_t) = \min_{\mathbf{u}_t} \hat{C}(\mathbf{z}_t, \mathbf{u}_{t:T}) \approx H(t, \mathbf{z}_t) = \min_{\mathbf{u}_t} H(t + dt, \mathbf{z}_{t+dt}) \tag{3.15}$$

With this substitution it may be possible to compute a solution that is not optimal, but still useful. As noted by Bertsekas [24], making this substitution recursively in deterministic problems often results in good performance. In fact, in many problems the performance

gain is *sequentially improving*, that is, strictly non-decreasing over time. For stochastic problems, similar results can be obtained by using a *path integral* formulation to compute $H(\cdot)$ [75, 138, 142], but this generally works well only for parameterized control policy formulations, which limits the representation space. However, Wu et al. [166] and Mansley et al. [100] demonstrated separately that rollouts applied directly to stochastic problems generally yield good results.

3.3 The constrained interference minimization principle

This section draws from the control principles described in the previous sections to define the constrained interference minimization principle. The principle is designed to model control in shared autonomy systems where controllers can be prioritized. This general method forms the basis of the SCIMP method, defined later in the chapter, and the Selective Determinism framework, described in Chapter 5. First the general problem is defined, and then a solution framework is presented.

For a given stochastic system at time t, the optimal constrained control $\mathbf{u}_t^\star$ is defined with respect to a desired control $\mathbf{u}_t^d$ for a confidence $\alpha \in [0, 1]$ as the result of the following optimization:

$$\begin{aligned} \mathbf{u}_t^\star &= \arg\min_{\mathbf{u}} \mu(\mathbf{u}, \mathbf{u}_t^d) \\ \text{s.t.} \quad & P(\text{good} \mid \mathbf{u}_t = \mathbf{u}) \geqslant \alpha \end{aligned} \tag{3.16}$$

In Equation 3.16 $\mu(\cdot)$ is a metric function over the control space and $\mathbf{u}_t$ is a control executed at time t. $P(\text{good} \mid \mathbf{u}_t = \mathbf{u})$ is the probability that the agent can remain on the good side of the decision boundary given $\mathbf{u}$ as the control executed at time t under belief

distribution $\mathbf{z}_t$ and followed by the control sequence most likely to remain on the good side of the decision boundary thereafter. If no $\mathbf{u}$ meets the α threshold, then $\mathbf{u}_t^\star$ is computed such that:

$$\mathbf{u}_t^\star = \arg\max_{\mathbf{u}} P(\text{good} \mid \mathbf{u}_t = \mathbf{u}) \tag{3.17}$$

This principle formulates the optimal cost-to-go function $J(\cdot)$ as a probabilistic decision problem that chooses between open-loop control for values above the confidence threshold, and closed-loop control for values below. One advantage is that confidence and control interference can be tuned in a problem-specific way using a single parameter α, with the resulting control $\mathbf{u}_t^\star$ satisfying the following properties:

1. The input control will be replicated *exactly* ($\mathbf{u}_t^\star = \mathbf{u}_t^d$) if there exists a future sequence of controls that is good with at least confidence α.

2. If the input control is such that no satisfactory sequence exists, then $\mathbf{u}_t^\star$ will be the closest control to $\mathbf{u}_t^d$ such that the α threshold is achieved.

3. If no control can meet the α threshold, the control that maximizes α is chosen.

The major challenge in implementing a framework utilizing this principle is evaluating $P(\text{good} \mid \mathbf{u}_t = \mathbf{u})$ because it requires solving a stochastic optimal control problem. Because of the practicality of implementation and generality of representation, the solution presented here will use a rollout approximation technique.

The heuristic chosen for the rollout approximation is based on the assumption that $P(\text{good} \mid \mathbf{u}_t = \mathbf{u})$ can be approximated by integrating over the optimal hypotheses evaluated under $\mathbf{z}_t$ by *assuming that the underlying state evolution is deterministic.* In other words, this framework assumes that the underlying transition function governing state transitions

starting from $\mathbf{x}_t$ is deterministic. This means the stochastic expectation in Equation 3.10 can be dropped, so that the cost-to-go becomes:

$$\hat{C}(\mathbf{z}_t, \mathbf{u}_{t:T}) = P(\text{good} \mid \mathbf{u}_t = \mathbf{u}) \approx \phi_T + \int_{\mathbf{z}_t} S(\mathbf{x}, \mathbf{u})\, p(\hat{\mathbf{x}}_t) \tag{3.18}$$

where $S(\mathbf{x}, \mathbf{u}) = C(\hat{\mathbf{x}}_t, \mathbf{u}_t)$ specifically denotes a deterministic cost function.

The problem is then to maximize Equation 3.18. Since this is formulated partially as a decision problem, the deterministic cost function $S(\mathbf{x}, \mathbf{u})$ becomes an indicator function that evaluates whether the system can remain in good states.

As formulated, $S(\mathbf{x}, \mathbf{u})$ is a deterministic problem that can be solved using any variety of optimal control or analytical techniques. But the problem remains of how to compute the integral in Equation 3.18; in practice it will typically not have analytical solutions. Thankfully, Monte Carlo methods provide simple, general, and powerful ways to estimate such integrals [75]. This framework employs such a method in its solution.

The integral in Equation 3.18 is estimated using Monte Carlo integration by sampling n state hypotheses $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ independently and at random according to $\mathbf{z}(\mathbf{x}_t = \mathbf{x})$ and evaluating $S(\mathbf{x}^{(i)}, \mathbf{u})$. The value of n required to estimate the probability that $P(\text{good} \mid \mathbf{u}_t = \mathbf{u}) \geqslant \alpha$ can be determined using a Bayesian interpretation: Suppose $k \leqslant n$ samples are found to satisfy $S(\mathbf{x}^{(i)}, \mathbf{u}) = 1$. The outcomes of each test are viewed as coin flips S_i for $i = 1, \ldots, n$ from a Bernoulli distribution with underlying probability of success $\theta \equiv P(\text{good} \mid \mathbf{u}_t = \mathbf{u})$. Let the results of these flips be the data D. Assume the prior over θ is given as a Beta distribution $Beta(\theta; a, b)$ where a and b are hyperparameters that indicate the prior belief that a control is good. Given the information D that k of n samples are good, the posterior

on θ is $Beta(\theta; a+k, b+n-k)$. The probability that the state $\mathbf{x}_t$ is good is then the expectation of θ, namely $(a+k)/(a+b+n)$. For convenience, choose $k = n$ so that all samples must be good. The necessary value of n becomes:

$$n = \left\lceil \frac{\alpha(a+b) - a}{1-\alpha} \right\rceil \tag{3.19}$$

Note that the hyperparameters a and b can be tuned, or learned, to reflect varying degrees of optimism with $a > b$ or pessimism with $a < b$. When no prior information is available, the uninformed prior $a = b = 1$ can be used. Also note that if the problem exhibits CE (§3.2.1), then the integral over the belief distribution $\mathbf{z}$ disappears (as shown in Equation 3.14) and $P(\text{good} \mid \mathbf{u}_t = \mathbf{u}) = S(\mathbf{x}, \mathbf{u})$. Technically this is equivalent to saying that the sample size n is always just 1, so the Monte Carlo formulation still applies, but clearly other simplifications can be made if CE holds.

The next issue is how to optimize Equation 3.16 subject to the probabilistic constraint. A naïve method would generate n samples of $\mathbf{u}$, sort them in increasing order of $\mu(\mathbf{u}, \mathbf{u}_t^d)$, then return the first that estimates $P(\text{good} \mid \mathbf{u}) \geqslant \alpha$. This is simple, but potentially computationally expensive. Instead, for certain problems it may be possible to extend $S(\mathbf{x}, \mathbf{u})$ so that it also provides the *set* of good controls $\mathcal{U}_{\text{good}}(\mathbf{x}) = \{\mathbf{u} \mid S(\mathbf{x}, \mathbf{u}) = 1\}$. Given such an extension, the constrained optimal control can be found by sampling n hypothetical states and solving the problem:

$$\begin{aligned} \mathbf{u}_t^\star &= \arg\min_{\mathbf{u} \in \mathcal{U}} \mu(\mathbf{u}, \mathbf{u}_t^d) \\ \text{s.t. } \mathbf{u} &\in \cap_{i=1}^n \mathcal{U}_{\text{good}}(\mathbf{x}^{(i)}) \end{aligned} \tag{3.20}$$

In particular, if the set of feasible controls is convex, then the feasible region is convex as well and efficient convex optimization approaches can be employed.

3.4 SCIMP: The safety-constrained interference minimization principle

This section introduces SCIMP, which is a specific use case for the constrained interference minimization principle. A motivation for this formulation is given, followed by its problem definition. An application to collision avoidance is demonstrated in two scenarios: rear-end collisions during single-lane driving, and transverse collisions during unprotected intersection crossings. In both cases only longitudinal control is considered. The intersection case poses a unique challenge for emergency systems because acceleration may need to be employed in addition to braking. In both cases SCIMP demonstrates that control can be calculated quickly using a tractable approximation. The α parameter allows the system to be tuned to trade off between two performance metrics: collision severity and driver interference. Experiments suggest that SCIMP is more robust than systems that do not consider uncertainty in their decision-making, and that the α parameter provides an intuitive interface to tune to desired levels of safety/interference tradeoff.

3.4.1 Motivation

Semi-autonomous collision avoidance and mitigation systems for automobiles are shared autonomy systems that treat a human driver as the nominal controller, and an automated collision avoidance or mitigation system as the backup controller. As industry races to develop and deploy these systems, they are also rapidly becoming commonplace. To understand why, consider that over 6.3 million automobile accidents occurred in the U.S. in 2007, including 1.8 million injury crashes and 37,435 fatalities at a cost of hundreds of billions of

dollars [113]. Although the numbers of injuries and fatalities per traveled mile have decreased significantly over the last four decades due to advances in active safety systems for vehicles, the rate of decline has leveled off over the last two decades. Contemporary active safety systems attempt to regulate the state of the vehicle without consideration for the state of the environment around the vehicle, specifically, the other vehicles on the road. Further gains in safety could be had by reasoning about the state of the vehicle in conjunction with the states of others on the road. Semi-autonomous safety systems do precisely this by incorporating information about other vehicles on the road in order to detect emergency situations.

Because they are a shared autonomy system, semi-autonomous safety systems must be able to recognize emergency scenarios and assess the risk of interfering with the driver's control versus the benefit of doing so. Additionally, designers of these systems must also consider the long-term effect they have on the driver's habits. In the literature for in-vehicle collision avoidance warning systems, Alm et al. [8] and Ben-Yaacov et al. [20] observed that a system considered to be a nuisance might simply be disabled by the driver, whereas Lehto [90] observed that too much automation can lead to inattentive or risk-seeking behavior behind the wheel. Worse, Fujita et al. [50] found that semi-autonomous collision avoidance systems that brake harshly can startle the driver and may cause them to lose control. Hence the important of designing semi-autonomous systems that *minimize interference* during operation.

Another major challenge for semi-autonomous systems is that of dealing with uncertainty in the driving environment. Uncertainty arises, for example, due to noisy sensor readings (from lidar, radar, camera, etc.), unobservable behavior intent of other agents (other traffic, pedestrians, etc.), errors in actuation due to tire wear and environmental factors, and so on. SCIMP is presented as a framework for reasoning under the safety constraints and uncertainty in these systems in a rigorous way, allowing the system and driver to share

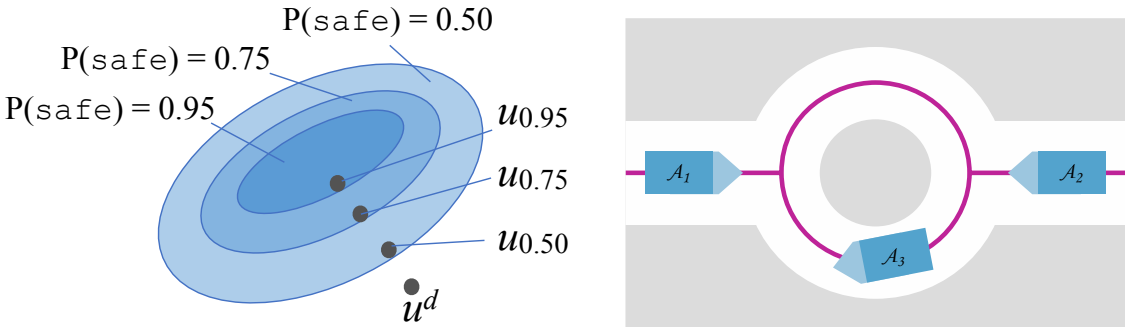


Figure 3.1: Left: Hypothetical control set subsets for agent A_1 corresponding to varying safety levels, with safe control $\mathbf{u}$ nearest to desired control $\mathbf{u}_d$ in each subset. Right: Agents A_1, A_2, and A_3 attempt to navigate a shared space and must choose safe controls to do so.

autonomy in a safe and predictable way.

3.4.2 Related work

Collision-imminent braking systems in some existing Volvo and Mercedes-Benz models use a variety of sensors to detect collision-imminent scenarios and apply brakes to reduce the severity of a crash. The interest here is in extrapolating collision-*imminent* braking to its inevitable conclusion: collision-*prevention* braking.

Autonomous driving has recently become a tremendously active field of research, inspired by major successes such as the DARPA Grand Challenge [144]. Waymo (previously Google Auto, LLC) alone logged over 635,000 autonomous miles in 2016, suffering only 124 disengagements [109]. Despite these major advances, there is still a major gap between these systems and human drivers. Although human drivers occasionally err, they are extremely reliable in general: a fatality crash occurs approximately once in 100 million miles driven [113]. Even if an autonomous vehicle relies on the human driver input once every 10,000 miles, the driver must be attentive 99.99% of the time for the system to perform as well as a human alone.

Active safety systems take control of the vehicle only during an emergency or when a

potential accident is foreseen in order to mitigate or avoid the consequences of an accident. A longitudinal collision-mitigation braking strategy was described by Hillenbrand et al. [59] that gradually applies stronger braking as the collision boundary is approached, which smoothes the control output and copes with some uncertainty by virtue of being slightly conservative. Anderson et al. [12] present a 2D hazard avoidance scheme based on model predictive control, which allows varying levels of autonomy based on risk assessed by control magnitudes, and Sivaraman [133] adaptively maintains distance from leading and following vehicles to safely perform cruise control in congested traffic. The approach presented in this chapter introduces the additional considerations of uncertainty, which provides a more natural definition of risk. Karlsson et al. [77] introduced a statistical decision rule that applies the brake if the probability of impact is greater than some threshold α. This approach is advantageous in the presence of uncertainty. Althoff et al. [11] applied similar thresholding techniques to autonomous driving in environments mapped using 2D range finders. The SCIMP framework presented in this work generalizes the probabilistic threshold approach to treat driver inputs and environmental uncertainty in a unified manner.

3.4.3 Problem definition

Assume an agent A travels along a known path in the presence of one or more moving obstacles $\mathcal{O}$ and that the configuration of A is fully determined by its position along the path. Suppose that at every time step t, A is given a desired control $\mathbf{u}_t^d$ and sensor input $\hat{\mathbf{x}}_t$, which it uses to infer a belief distribution $\mathbf{z}_t$ over hypothetical agent and obstacle system states. The control problem A must solve is described below in Problem 4:

Problem 4. For a path P, obstacle set $\mathcal{O}$, time $t \in T$, confidence level α, and desired control $\mathbf{u}_t^d$, find a control $\mathbf{u}_t^\star$ that attempts to satisfy an ordered set of constraints:

1. A is at least α confident that $\mathbf{u}_t^\star$ will not lead to future collision. If no $\mathbf{u}_t^\star$ exists that is collision free to α confidence, A computes $\mathbf{u}_t^\star$ to maximize confidence.

2. The computed control $\mathbf{u}_t^\star$ should deviate minimally from $\mathbf{u}_t^d$.

By the ordering, Constraint 2 is applied to the satisfying set of Constraint 1.

Problem 4 is the general SCIMP problem. To formulate it as a constrained interference minimization problem, let the term *safe* replace the term *good*, and let it be a descriptor that indicates that the ego vehicle remains collision free. Define $S(\mathbf{x}, \mathbf{u})$ to be a deterministic and complete control algorithm that serves as an indicator function for whether the ego vehicle can *remain* safe starting in state $\mathbf{x}$, executing $\mathbf{u}$, and following the optimally safe policy thereafter. Finally, because this formulation deals only with one-dimensional longitudinal control, the metric function is defined simply as the absolute difference between controls: $\mu(\mathbf{u}_1, \mathbf{u}_2) = |\mathbf{u}_1 - \mathbf{u}_2|$.

3.5 SCIMP application scenarios

This section applies SCIMP to two scenarios: 1) rear-end collisions along a single-lane, and 2) transverse collisions during lane crossing. These are two of the most significant sources of automobile accidents involving elderly drivers [115]. In both cases, an implementation of the SCIMP subroutine $S(\mathbf{x}, \mathbf{u})$ for evaluating the safety of a control $\mathbf{u}$ at a state $\mathbf{x}$ and remaining safe for all deterministic rollouts of $\mathbf{x}$ is given. Subroutines to calculate the set of safe controls $\mathcal{U}_{\text{safe}}(\mathbf{x})$ are also given. The rear-end collision scenario in particular lends itself to convenient optimization because $\mathcal{U}_{\text{safe}}(\mathbf{x})$ has a simple form. The lane crossing scenario is more challenging because acceleration may be needed in addition to braking, and multiple dynamic obstacles may need to be tracked. In this case, the developments of Chapter 2

are used to furnish an optimal, exact, polynomial-time planner that is used to compute $S(\mathbf{x}, \mathbf{u})$ and $\mathcal{U}_{\text{safe}}(\mathbf{x})$ in lane crossing scenarios. As noted earlier, the control spaces dealt with here are strictly one-dimensional (they only concern longitudinal controls), but the principle generalizes to arbitrary controls.

3.5.1 Collision-imminent braking

[5]Assume a single obstacle, and that the vehicle is moving in the same direction as the obstacle and that the obstacle does not move in reverse. Assume further that the vehicle is equipped with odometric and ranging sensors. The behavior of the obstacle is represented as its acceleration, which the ego vehicle must infer from sensor measurements. In order to synthesize this information, suppose the ego vehicle runs an Extended Kalman Filter (EKF), which acts as its perception algorithm, to maintain a belief distribution over the system states[6] [55]. The belief distribution produced by the EKF is then incorporated into the probabilistic decision rule.

Stochastic dynamics model

The state of the ego vehicle can be described using its position p, velocity v and the maximum deceleration $a_{\min}$ that it can currently apply. Additionally, when an obstacle is present, the obstacle position p_o, velocity v_o, and acceleration a_o are modeled as part of the system state. The vehicle receives a noisy odometry signal v_s and range reading d. The dynamics and sensing are stochastic, with conservative noise parameters in Table 3.1.

[5]The work presented in this subsection is due to Yajia Zhang, who developed it for [68].

[6]For more complex scenes, with multiple objects and/or multiple sensor modalities, more sophisticated tracking techniques should be used, e.g. [36, 98, 155]

Extended Kalman Filter

The ego vehicle is assumed to employ an EKF in order to estimate the state from the stochastic dynamics and observations. An EKF is a version of the Kalman filter that addresses nonlinear systems by linearizing about the estimated mean and covariance [159]. While the EKF suffers from problems in highly nonlinear systems, in this case the system is close to linear and the EKF provides sufficiently accurate performance. For highly non-linear systems, other and more general estimators can be used, such as particle filters.

For this problem, the dynamics at time step t can be written as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + w_t \tag{3.21}$$

$$z_t = h(\mathbf{x}_t) + v_t \tag{3.22}$$

$$w_t \sim \mathcal{N}(0, Q_t) \tag{3.23}$$

$$v_t \sim \mathcal{N}(0, R_t) \tag{3.24}$$

Here, $\mathbf{x}_t$ denotes the joint state $(p, v, a_{\min}, p_o, v_o, a_o)$ that describes for the ego vehicle the position, velocity, and minimum acceleration available due to road surface conditions, and for the obstacle vehicle its position, velocity, and acceleration. $\mathbf{u}_t$ denotes the braking control input, z_t is the observation (d, v_s) at time step k. w_t is the process error term with Q_t as its covariance matrix. v_t is the measurement noise term with R_t as its covariance matrix. At each step, the EKF maintains a state estimate $\hat{\mathbf{x}}_t$ and covariance matrix P_t. Upon reading the observation z_t from the vehicle's sensors, the EKF performs a Kalman update using the

Table 3.1: Dynamics and Observation Models

Road Surface $\dot{a}$	Max. applicable acceleration is a random walk $\dot{a} \sim \mathcal{N}(0, \Delta t)$
Actuation Errors $\dot{v}$	Proportional to control and max. deceleration, $\dot{v} = ua(1 + e_u)$, with $\epsilon_u \sim \mathcal{N}(0, 0.01^2)$
Object Behavior $\dot{a}_o$	Random noise with 99.99% within $[-5.0\, m/s^2, 5.0\, m/s^2]$, $\dot{a}_o \sim \mathcal{N}(0, 1.25^2)$
Speedometer v_s	Multiplicative noise on actual velocity, $v_s = v(1 + \epsilon_s), \epsilon_s \sim \mathcal{N}(0, 0.025^2)$ (99.99% within 10% of current velocity)
Range Reading d	Combined linear and multiplicative noise $d = n_{dL} + (p_o - p)(1 + n_{dM})$, with $n_{dL}, n_{dM} \sim \mathcal{N}(0, 0.0125^2)$ (99.99% within $5cm$ + 5% of true distance)

system linearized about $\hat{\mathbf{x}}_t$ to obtain a new state estimate $\hat{\mathbf{x}}_{t+1}$ and covariance P_{t+1}.

Because obstacles may move in and out of sensor range, the obstacle state and distance measurements are included in the EKF update only when an obstacle is detected. When an obstacle appears for the first time, its position estimate is initialized to the raw range sensor estimate $\mathcal{N}(d, (0.0125d)^2)$. Its velocity is initialized to a broad distribution $\mathcal{N}(\hat{v}/2, (\hat{v}/2)^2)$, and its acceleration is initialized to $\mathcal{N}(0, (2.5\, m/s^2)^2)$.

Known-state braking policy

Given known state information, a naïve braking control policy $\pi(\mathbf{x}_t)$ is defined in Algorithm 7. Note that $\pi(\mathbf{x}_t)$ assumes constant obstacle behavior forward in time and it only considers a constant headway C. More sophisticated control policies, like the intelligent driver model [148], or learned behavior models [127], could be used to consider interactive obstacle behavior and variable headway.

In $\pi(\mathbf{x}_t)$, C is a constant that is used for indicating a nominal headway margin, which in experiments is set to $1m$. p' is the estimated stopping position of the vehicle if it initiates

Algorithm 7 This braking policy

```
1: procedure BangBangPolicy(x_t)
2:     p' ← p + v^2/(2a_min) + C
3:     t' ← v/a_min
4:     if v_o + a_o t' ⩾ 0 then
5:         p_o' ← p_o + v_o t' + 1/2 a_o t'^2
6:     else
7:         p_o' ← p_o + v_o^2/(−2a_o)
8:     end if
9:     if p' > p'_o then
10:        return -1
11:    else
12:        return 0
13:    end if
14: end procedure
```

maximum braking and t' the estimated time it will take to reach a stop. p'_o defines the estimated position of the obstacle when the vehicle stops. Because the obstacle is assumed to not move backwards, line 4 tests whether the obstacle achieves zero velocity after t' or before, and then computes p'_o accordingly. Finally, if $p' > p'_o$, there will be a collision between vehicle and obstacle, otherwise, no collision.

Using this policy, $S(\mathbf{x}, \mathbf{u})$ and Equation 3.20 can be implemented in a straightforward manner. Note that stronger braking is always guaranteed to be safer, so for each state sample $\mathbf{x}^{(k)}$ all controls lower than $\pi(\mathbf{x})$ are safe. So, the SCIMP optimization is reduced to 1) testing if the user's control is safe, and if not, 2) finding the control policy with the weakest braking control $a_{\min}$ that keeps the system safe with probability α.

3.5.2 Intersection crossing

Intersection crossing requires consideration of both braking and acceleration in order to navigate. It also requires considering the behavior of multiple obstacles, which makes optimal decision boundaries more complex even in the known-state case. Even so, SCIMP applies

directly once $S(\mathbf{x}, \mathbf{u})$ and $\mathcal{U}_{\text{safe}}(\mathbf{x})$ have been defined. Because the PST-space-planning algorithm derived in Chapter 2 is deterministic and exhibits completeness, it is valid for use in defining $S(\mathbf{x}, \mathbf{u})$. In this case, the value of $S(\mathbf{x}, \mathbf{u})$ is computed by running the planner and indicating whether any feasible control trajectory exists.

Efficient SCIMP optimization

The visibility graph data structure of the PST-space-planner is useful because it can be used to propagate visibility forwards and backwards between the goal and regions in PT-space space. In this case, let $R^s(\mathbf{x}_t; t + \Delta t, \mathcal{C})$ be the PST-space region reachable from a start state $\mathbf{x}_t$ at a time $t + \Delta t$ and under a set of constraints $\mathcal{C}$, and let $R^g(t + \Delta t, \mathcal{C})$ be the PST-space region at time $t + \Delta t$ and under constraints $\mathcal{C}$ that admits feasible trajectories that connect to the goal region. The intersection $R = R^s \cap R^g$ in PST-space space is exactly the region that is reachable from $\mathbf{x}_t$ at time $t + \Delta t$ and that also admits feasible trajectories to the goal. R also allows straightforward computation of the safe control set necessary for the SCIMP optimization (Equation 3.20). The computation of R^s and R^g follow straightforwardly from the algorithms derived in Chapter 2, so they will not be described here. Instead, this section will focus on how the PST-space planning algorithms are employed to compute $\mathbf{u}^\star$.

Recall that the goal is to compute the control that is closest to $\mathbf{u}^d$ and is safe for all state samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}$. If no such control exists, the goal is to compute the control that is maximally safe. The optimization proceeds by iteratively drawing k state samples and incrementally building an obstacle set $\mathcal{O}$ and constraint set $\mathcal{C}$ that are guaranteed to contain and satisfy, respectively, the obstacles and constraints from all satisfying samples so far. To build $\mathcal{O}$, the algorithm first initializes an empty $\mathcal{O}$ and with state sample $\mathbf{x}^{(i)}$ grows the obstacles in $O_j \in \mathcal{O}$ using a convex hull operation such that each $O_j^i \supseteq O_j^{(i-1)}$. Likewise,

the minimum acceleration available in $\mathcal{C}$ is initialized as $\mathcal{C}_{a_{\min}} = -\infty$ and maintained as $\mathcal{C}_{a_{\min}} = \max\left(\mathcal{C}_{a_{\min}}, a^{(i)}_{\min}\right)$.

At each iteration the reachable regions R^s and R^g then can be computed using $\mathcal{O}$ and $\mathcal{C}$ with the algorithms from Chapter 2, from which R can then also be directly computed. Once R is computed, the set of feasible speeds $\mathcal{F}$ at R can be retrieved, and the control signals that achieve those speeds can be backed out, yielding $\mathcal{U}_{\text{safe}}$. Note that the PST-space algorithms may generate an $\mathcal{F}$ that consists of multiple disjoint intervals (see Chapter 2, §2.3), leading to a $\mathcal{U}_{\text{safe}}$ that also consists of multiple disjoint intervals. This means that computing $\mathbf{u}^\star$ is a matter of checking for containment in some sub-interval of $\mathcal{U}_{\text{safe}}$, in which case $\mathbf{u}^\star = \mathbf{u}^d$, or choosing as $\mathbf{u}^\star$ the sub-interval boundary of $\mathcal{U}_{\text{safe}}$ nearest $\mathbf{u}^d$.

If, during optimization, $\mathcal{U}_{\text{safe}}$ becomes empty, this indicates that the α threshold cannot be satisfied and optimization changes to satisfy Equation 3.17. To do this, a *Minimum Constraint Removal* [40, 58] problem is formulated to find the fewest samples that must be discarded in order for $\mathcal{U}_{\text{safe}}$ to be non-empty. While this problem is NP-Hard, and so impractical for large sample sets, greedy approximations work well in practice. Once it is constructed, $\mathbf{u}^\star$ is chosen as the control that is maximally distant from any boundary in $\mathcal{U}_{\text{safe}}$. The assumption in doing so is that this is the control that has the most room for error that may occur during actuation.

3.5.3 Evaluation

A good emergency safety system should be able to achieve low collision risk and low driver interference. To evaluate performance, the characteristics of collision velocity (CV) and an *interference index* (II) that combines several aspects of driver interference are considered. II attempts to measure deviation from the driver's desired behavior and is a function of the

following components:

Discontinuity Time (DT) The amount of excess acceleration experienced. This is computed as the integral of the time over which the acceleration at two subsequent time steps is greater than a threshold, which is here set to $4m/s^2$.

Excess Time (ET) The amount of time consumed by excessive interference in a scenario. A scenario is considered completed if the car reaches a goal, collides with an obstacle, or, in the braking case, comes to a stop. ET is computed by measuring the policy completion time and subtracting the completion time for an optimal collision-free policy with perfect state information. Because some policies can complete a scenario faster than the optimal policy (e.g., by colliding with an obstacle), ET may be negative.

Stopping Distance (SD) This metric measures the distance to the obstacle after the vehicle stops, or 0 if the scenario is completed in any other manner. SD is only measurable in braking scenarios 1–3 and 5; it is 0 for all other scenarios.

Given the above components, II is computed as follows, where c_1, c_2, and c_3 are proportionality constants:

$$\text{II} = c_1 DT + c_2 ET + c_3 SD$$

Based on empirical tuning, the proportionality constraints are to $10\,s^{-1}$, $1\,s^{-1}$, and $1/2\,m^{-1}$ respectively. In practice, human subjects experiments could be used to determine weights that yield an interference index that is better tuned for human drivers.

Five test scenarios for the braking condition and three for the intersection crossing condition are defined. The scenarios simulate actual environments that an emergency safety system might face in practice, and the behavior of obstacles and simulation constants are *not* known in advance to the ego vehicle. Rather, it must infer them through sensor readings.

The braking scenarios are illustrated in Figure 3.2. They include fixed obstacles, hard-braking obstacles, transient lane-crossing obstacles, and false positives and negatives. In all cases, the vehicle starts at $20\,m/s$ and the driver's control maintains a steady velocity. The intersection crossing scenarios are illustrated in Figure 3.3. In all cases the driver's control is a constant acceleration of $4\,m/s^2$. In the *between obstacles* scenario, the car is $22\,m$ from crossing a two-lane road intersection and has an initial velocity of $4\,m/s$. Two obstacles are approaching from either direction at $8\,m/s$ with constant velocity. The two *side-impact* scenarios have the following initial conditions: 1) the vehicle collides with the obstacle unless it brakes, and 2) the obstacle collides head-on with the vehicle unless the vehicle accelerates. In condition 1, the car's initial velocity is $4\,m/s$ and the obstacle approaches with velocity $7.5m/s$, while in condition 2, the car has initial velocity $10m/s$ and the obstacle approaches with velocity $7m/s$.

The plots above each scenario in Figures 3.2 & 3.3 compare the SCIMP policies with $\alpha =$ 60, 70, 80, 90, 95, and 99, to the ideal omniscient controller (Ideal), which has perfect information about the future behavior of the environment; the controller that runs deterministic optimal control on the most-likely environment state (Basic); and the raw driver's control (No control). Monte Carlo evaluations were performed for each policy with 100 samples under the stochastic sensing and dynamics models.

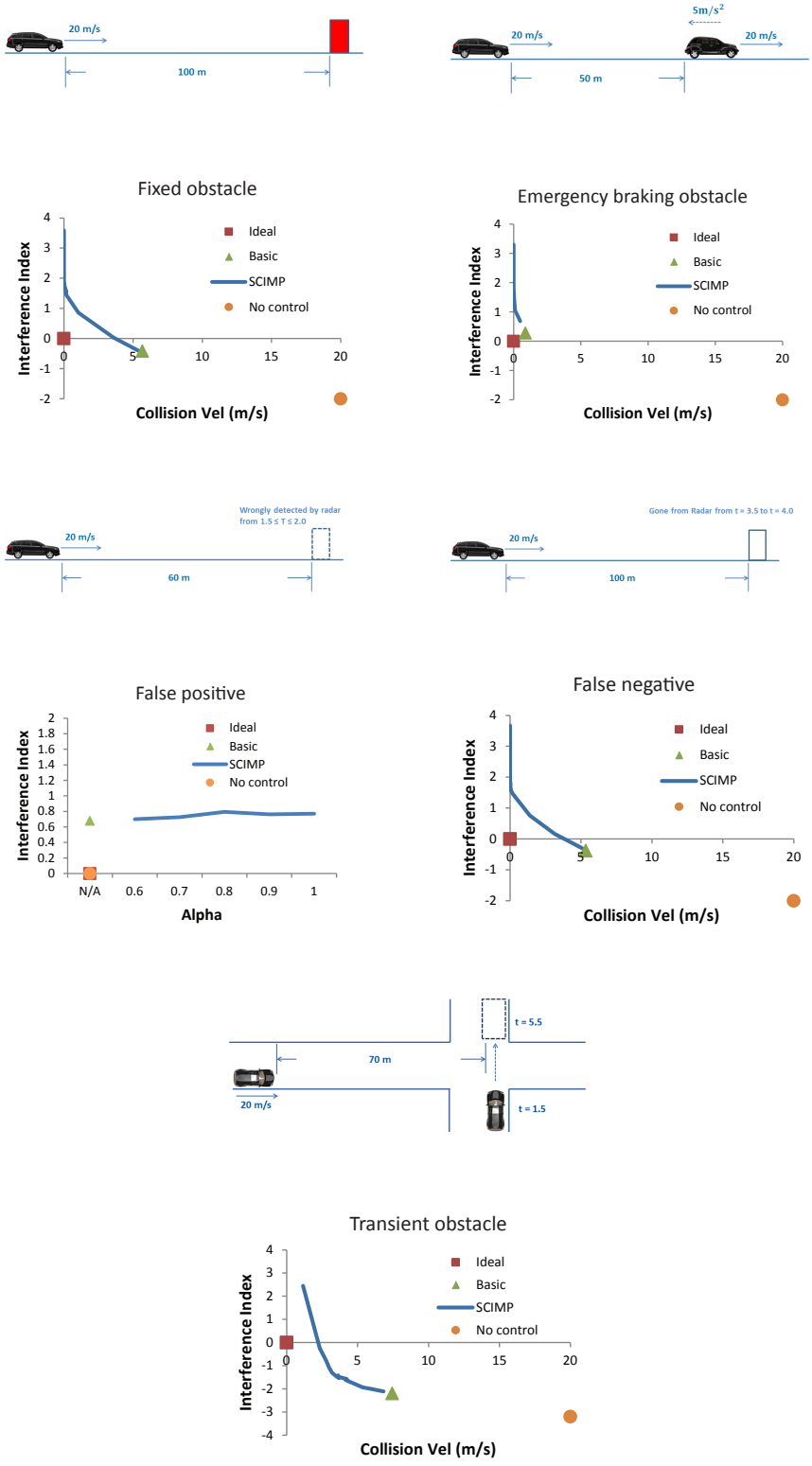


Figure 3.2: Five braking scenarios. Results are averaged over 100 trials

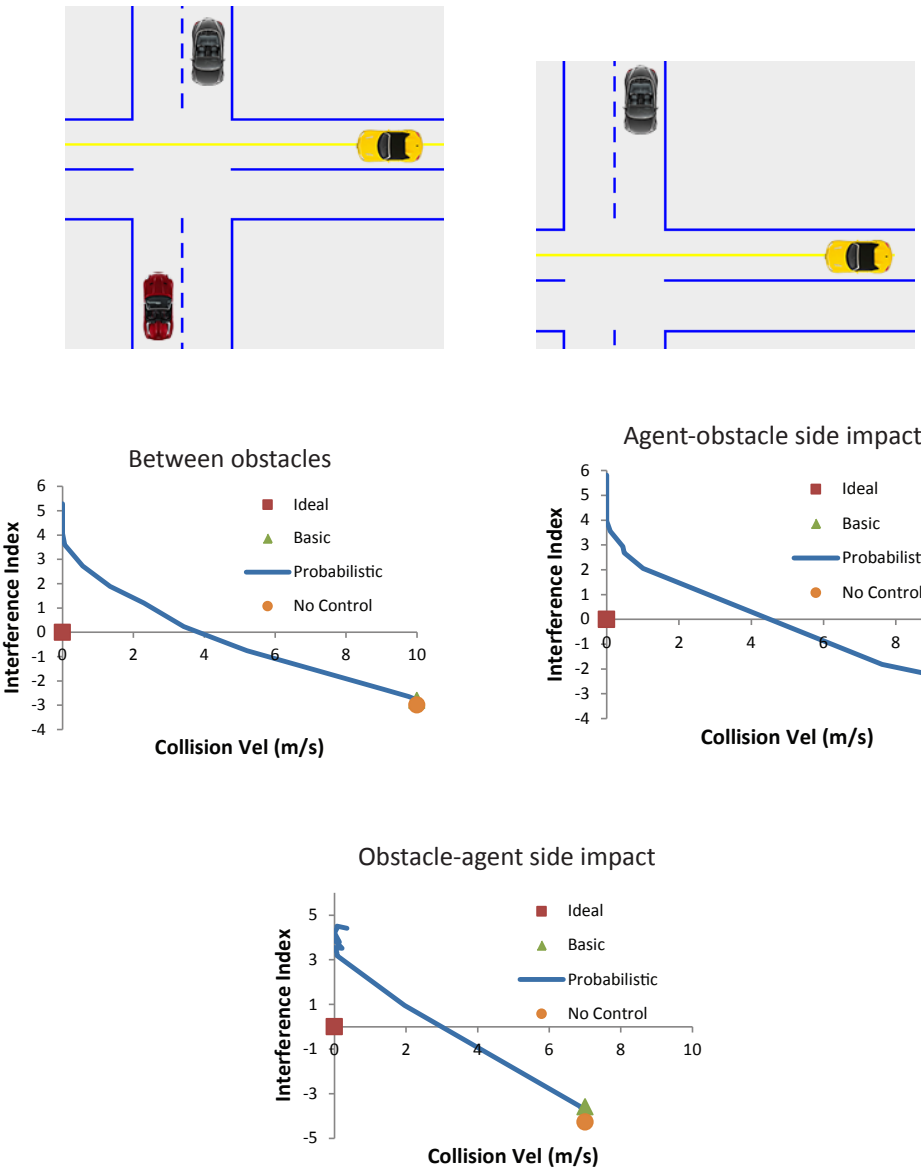


Figure 3.3: Three intersection scenarios. Results are averaged over 100 trials.

3.6 Conclusion

A generic constrained interference minimization principle has been derived from stochastic optimal control theory that can be used to enable shared autonomy systems to interact in well-defined ways in the presence of behavioral and environmental uncertainty. Background is given on the assumptions underlying the rule so that informed design decisions can be made during implementation.

In addition, a specialized safety-constrained variant of the principle was derived, which was applied to the specific use case of semi-autonomous collision avoidance systems for automobiles. Two concrete implementations of SCIMP are developed: First, a probabilistic collision-avoidance braking strategy is given that considers uncertainty in vehicle dynamics, sensor noise, and unpredictable obstacle behavior. Second, a technique for determining safe trajectories for intersection crossings in the presence of state uncertainty is presented. A number of Monte Carlo simulations demonstrate that SCIMP achieves low collision risk and driver interference in a variety of scenarios.

3.7 Acknowledgements

This work was partially supported by the Indiana University Collaborative Research Grant fund of the Office of the Vice President for Research.

Chapter 4

Factoring interaction effects in collision avoidance

Synopsis: *This chapter examines how dynamics and complexity are related in multi-agent collision avoidance. Specifically, that interaction effects between agents can, under certain conditions, be factored out of the problem. Motivated particularly by work in the field of automated driving, this chapter considers a variant of the reciprocal n-body collision avoidance problem. In this problem, agents must avoid collision while moving according to individual reward functions in a crowded environment. The main contribution of this chapter is the result that there is a quantifiable relationship between system dynamics and the requirement for agent coordination, and that this requirement can allow interaction effects to be factored out, thereby changing the complexity class of the problem dramatically: from NEXP, or even $NEXP^{NP}$, to P. A constructive proof demonstrates the relationship, and potential applications are discussed.*

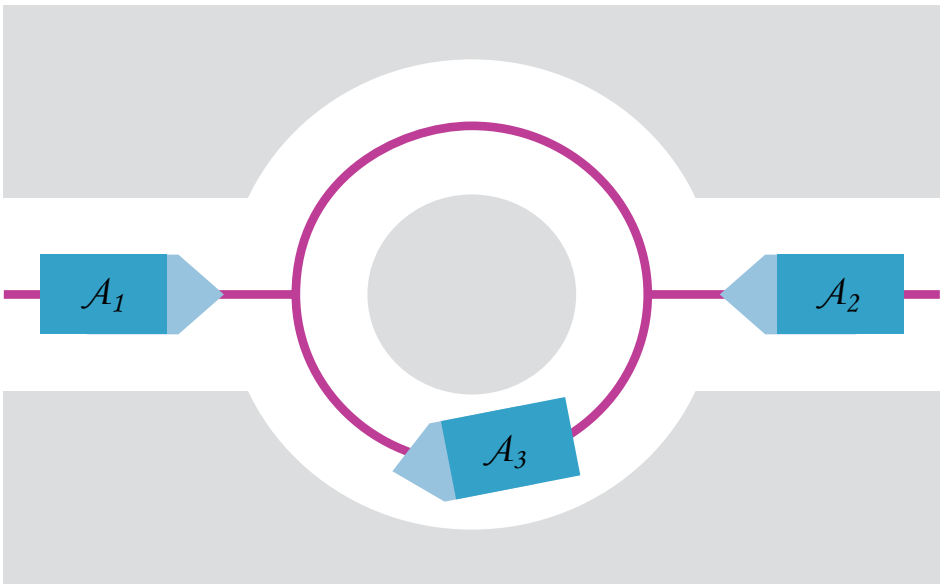


Figure 4.1: Agents A_1, A_2, and A_3 attempt to navigate past each other along fixed paths. This chapter examines how system dynamics affect the need for them to coordinate their actions.

4.1 Introduction

In industries as varied as mining, agriculture, health care, and automated driving, many practical applications in robotics involve navigating through dynamic environments in the presence of intelligent agents. A large and relatively mature body of literature has been developed that examines various types of these multi-agent systems and the theoretical complexity of planning within them. The focus of this work is specifically how system dynamics interact with problem complexity. For single agent systems, an early result due to Reif and Sharir [126] showed that adding velocity bounds to one type of motion planning problem can change its complexity from NP-hard to PSPACE-hard. This result indicates that system dynamics can play a role in determining complexity class, however, relatively little attention has been paid to the role that system dynamics play in the complexity of problems involving systems of multiple agents.

As will be shown, one of the primary factors affecting complexity of multi-agent problems

is agent coordination. Despite this, agent coordination is often not treated as a primary factory when mutli-agent problems are approached in the world. Typically, path planning is addressed first with agent coordination being added in. In many cases, this prioritization may cause the problem to be modeled in a way that introduces prohibitive amounts of complexity. Take, for example, the case of an automated vehicle moving along a pre-defined road network. While it is tempting to model the problem primarily as a path planning problem, doing so may put in place requirements that make addressing agent coordination more difficult, specifically, the long planning horizons and precise knowledge of future world states that path planning tends to require.

As robotics research has moved further from laboratories and into the real world, multi-agent problems requiring non-trivial coordination have become more important. The team behind the planner used in the Bertha Benz drive (Ziegler et al. [168]), as well as the winning teams of the 2007 DARPA Urban Challenge (Urmson et al. [150], Montemerlo et al. [107], and Bacha et al. [14]) all cite coordination ahead of path planning as an area of future work. The coordination problem also has deep ties to long-standing problems in optimal control theory. Mitter and Sahai [106] identified the coordination problem as the primary difficulty in designing an optimal controller for Witsenhausen's counterexample [163], which counters the idea that the optimality properties of linear-quadratic-Gaussian control extend to de-centralized systems. Given the importance of the coordination problem and that practically any real-world system must reason under dynamic constraints, it is important to understand how coordination, system dynamics, and complexity interplay.

Efforts to address coordination in the vehicle domain have explored many avenues. Explicit coordination among vehicles (V2V) would seem to provide a solution, and much work has been done to develop the technology and standards. However, it is not likely that the

availability of the required communication channels can be guaranteed to levels needed for safety-critical applications [57]. But it is also unclear to what degree that kind of communication is actually necessary: human drivers navigate successfully with only limited[1] forms of communication, which implies that the coordination they do is also limited. This raises the question of to what extent coordination is actually required for navigating multi-agent systems, and it implies that a better understanding of that requirement will lead to the development of more practical and robust navigation algorithms.

This chapter examines agent coordination in a variant of the reciprocal n-body collision avoidance problem described by van den Berg et al. [152]. The key insight is that system dynamics can introduce a requirement for coordination where there otherwise would be none, and a constructive proof is given that allows the existence of this requirement to be determined. The importance of the coordination requirement is that once it exists within a system, the space of appropriate models for the problem changes, which changes the complexity class of any solutions to the problem. This result demonstrates the existence of fundamental ties between system dynamics and problem complexity for multi-agent collision avoidance problems. Preliminary work for this chapter was published in Johnson [69].

4.2 Related work

This section will detail a selection of the large body of relevant work on collision avoidance, planning, and complexity. Several general theoretical complexity results are described, followed by a survey of notable solution techniques for multi-agent collision avoidance and of solution techniques for hybrid dynamical systems.

[1]Indicator lights are a common channel of communication, but they are notoriously unreliable. Horns also provide a form of communication, but are limited by context. Relative positions and speeds can convey intent, but, as channels of communication, these are very low bandwidth.

In the absence of dynamic constraints and other moving agents, the problem of planning a collision-free path through an environment is typically referred to as the *mover's problem*, which is the problem of moving an articulated polyhedral body through a Euclidean space populated with static polyhedral obstacles. Reif [125] showed the general problem to be complete for PSPACE and the classical problem, referred to as the *piano mover's problem*, where the moving body is a rigid polyhedron moving in $\mathbb{R}^2$ or $\mathbb{R}^3$, to be in P under the condition that geometric constraints can be expressed algebraically. Work by Halperin and Sharir [56] further showed near quadratic bounds for the $\mathbb{R}^2$ case. The multi-body variant of the piano mover's problem, known as the *warehouseman's problem*, was shown by Hopcroft et al. [60] to be PSPACE-hard. Reif and Sharir [126] additionally showed that introducing agents that follow fixed-trajectories into the piano mover's problem for $\mathbb{R}^3$ changes the complexity class of the problem to NP-hard, and that adding velocity bounds makes the problem PSPACE-hard.

For multiple agents following non-fixed trajectories, the problem is generally formulated in terms of sequential decision making in a discretized space rather than geometric motion in a continuous space. When planning among the agents can be done independently while still achieving a jointly optimal solution, the problem can be formulated as a type of Markov decision process (MDP), which Papadimitriou and Tsitsiklis [120] showed belongs to complexity class P. As noted by Boutilier [29], independent planning cannot in general guarantee a globally optimal plan; only joint planning can guarantee global optimality. While joint planning problems with a central planner that computes motions for all agents can also be formulated as types of MDP's, and therefore belong to P, others, such as the unlabeled variant, where multiple agents must reach multiple goal positions without restrictions for which agent reaches which goal, were shown by Solovey and Halperin [134] to be PSPACE-hard. For

decentralized problems Bernstein et al. [21] showed that for cooperative agents (i.e., agents that share a reward function) this class of problems is at least complete for NEXP in both the jointly fully-observable (DEC-MDP) and jointly partially observable (DEC-POMDP) cases. Goldsmith and Mundhenk [53] showed that the partially observable stochastic game (POSG), which is the non-cooperative version of this problem (i.e., the problem in which agents do not share a reward function), is complete for NEXP^{NP}.

In the context of collision avoidance in multi-agent systems, Fiorini and Shiller [45] introduced the notion of *velocity obstacles* to address the pairwise collision avoidance problem. In their approach the set of velocities resulting in collision between a robot and another moving agent are computed explicitly, and this set is called the velocity obstacle (VO). Collision avoidance is then guaranteed by assigning velocities outside the VO to the agent. Fraichard and Asama [49] described a more general *inevitable collision state* (ICS) as a "state for which, no matter what the future trajectory followed by the system is, a collision with an obstacle eventually occurs." Similar state descriptors had been proposed by LaValle and Kuffner [88]. Owing to the inherent computational complexity of the ICS representation, Bekris [18] examined sampling-based approximation methods. The Optimal Reciprocal Collision Avoidance (ORCA) framework introduced by van den Berg et al. [152] expanded the ideas of VO and ICS to driftless, non-inertially constrained multi-agent systems. Later, van den Berg et al. [154] extended their results to consider systems with inertial constraints. Pairwise collision avoidance for holonomically constrained systems was demonstrated by Wilkie et al. [162] and extended to general multi-agent systems by Alonso-Mora et al. [9]. When coordination among agents is allowed, Bekris et al. [19] demonstrated that non-collision can be guaranteed for a broad class of de-centralized motion planning problems. Shoham and Tennenholtz [130] describe an alternate approach to these types of problems that imposes

artificial rules, or *social laws*, on agent coordination in order to remove the need for online coordination altogether.

The distinction between sequential decision making and continuous geometric motion planning problems is typically formulated mathematically as the problem of choosing among a finite number of homotopy channels in some state space (decision making), and generating actuation commands to navigate those channels (motion planning). In practice, most problems have characteristics of both problem types, and therefore are hybrid problems with hybrid solutions. An early solution approach from Kambhampati et al. [73] interleaved graph and motion planning in pre-defined discrete spaces. Later, Kaelbling and Lozano-Pérez [71] dealt with uncertainty and introduced sophisticated task description languages. When the connectivity of the state space is not known beforehand, Alterovitz et al. [10] introduced a sampling-based approach that can be used to construct roadmaps in the state space.

4.3 Problem description

In this work, the reciprocal n-body collision avoidance problem is extended to allow general system dynamics, and to make each agent's task to choose an appropriate control rather than velocity:

Problem 5. Let $\mathcal{A}$ be a set of agents navigating a shared space with a shared reference frame and assume that collision is never inevitable in the initial system state. Assume each agent can observe the instantaneous dynamic state of the environment and has limited (i.e. non-infinite) computational resources. Assume each agent has knowledge of the physical dynamic properties of the environment and of other agents, but that each agent actuates according to a unique decision process. Each agent may assume with certainty that other agents

will prefer both to avoid collision and to avoid causing collision, but otherwise the decision processes of other agents are not fully observable except through coordination. Agents may coordinate their decision processes via communication under the following restrictions:

1. Communications are strictly pairwise
2. Agents may only communicate with regard to their own actions (i.e., they may not relay information)
3. There is always some non-zero cost associated with communication

When $|\mathcal{A}| > 2$, how can a given agent choose a control with the guarantee that it will be possible for *all* agents to remain collision free for some time horizon?

The focus of this chapter is on how system dynamics affect the model space and complexity of Problem 5.

4.3.1 Notations and definitions

Assume all agents operate in a shared workspace $\mathbb{W}$, and let $\mathbb{S}$ denote the state space for all agents. Let Φ denote the set of control trajectories available to an agent A, where for each $\phi \in \Phi$ the state of A at time t from initial state $\mathbf{x} \in \mathbb{S}$ under control trajectory ϕ is $\mathbf{x}_t = \phi(\mathbf{x}, t)$. Let $A(\mathbf{x})$ denote the state and volume of state space occupied by the geometric model of agent A at state $\mathbf{x}$. Assume agents may be *interacting* or *non-interacting* as defined below.

Definition 8. An *interacting agent* is one whose dynamic state is a function of the dynamic state of the system and an internal policy (for example, pedestrians or animals could be interacting agents).

Definition 9. A *non-interacting agent* is one whose dynamic state is a function only of the dynamic state of the system (for example, trees or rolling rocks could be non-interacting agents).

Definition 10. The actions of two agents are said to require *coordination* when the feasibility the control sequence either agent uses is not independent of the other's.

Definition 11. For an agent A navigating an environment occupied by a set of interacting agents and non-interacting agents $\mathcal{A}$, an *obstacle* O is a member of the set of obstacles $\mathcal{O}$, which is defined as:

$$\mathcal{O} = \mathcal{A} \setminus A$$

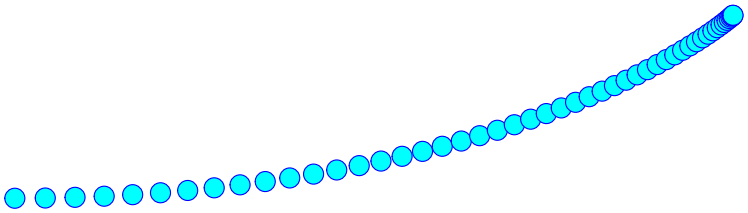
Definition 12. A *state space obstacle* (B) is the volume swept out by an obstacle O over a time T as it evolves from an initial state $\mathbf{x}_i$ under a control trajectory ϕ_i:

$$B = \bigcup_{t \in T} O(\phi_i(\mathbf{x}_i, t))$$

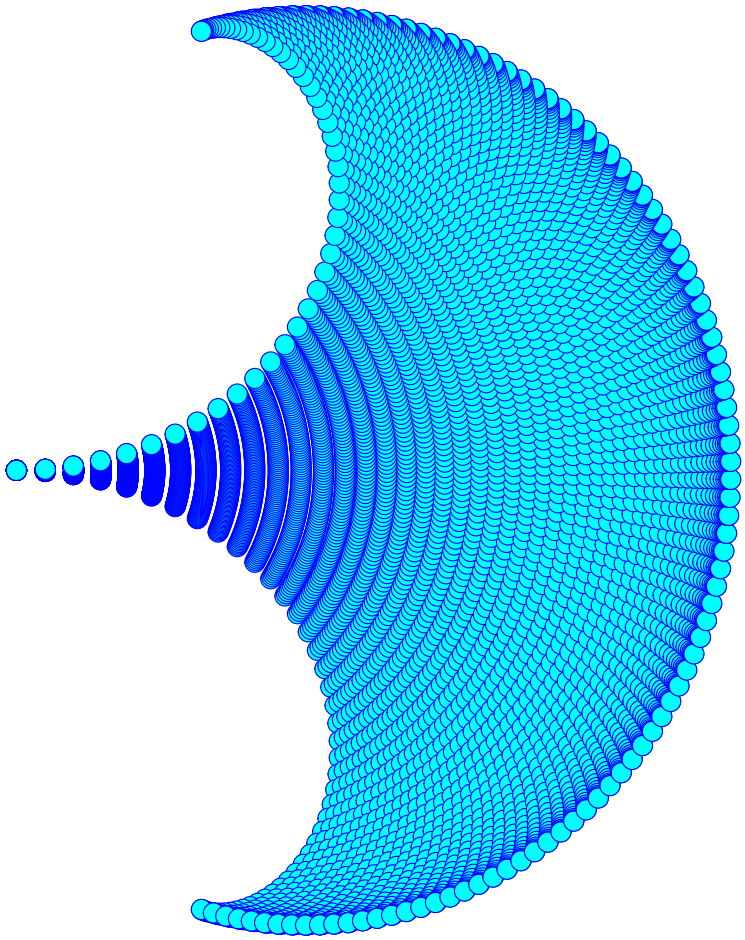
Definition 13. An *inevitable collision state* (ICS) for an agent A is a state from which all feasible future trajectories of A result in collision:

$$\mathbf{x} \text{ is ICS} \leftrightarrow \forall \phi, \exists B_i, \exists t :: A(\phi(\mathbf{x}, t)) \cap B_i \neq \varnothing$$

ICS notations and definitions adapted from Fraichard and Asama [49].



(a) For a given agent state $A(\mathbf{x})$ and path P, the stopping path $SP(A(\mathbf{x}), p)$ is the minimal set of states A must occupy while coming to zero velocity from $\mathbf{x}$ along P. Here disc agent A starts on the left and comes to a stop in the upper right. In the illustration the motion is discretized at fixed time intervals, so the spacing between steps indicates relative speed.



(b) For a given agent state $A(\mathbf{x})$ and complete set of followable paths $\mathcal{P}$, the stopping region $SR(A(\mathbf{x}), \mathcal{P})$ is the union of all SPs over $\mathcal{P}$. This illustration shows the SR for disc agent A from (a). The region is plotted by sampling agent trajectories generated by sweeping steering from hard right to hard left.

Figure 4.2: Illustrations of the shapes of an SP (Definition 15) and an SR (Definition 16) for a disc agent following constant control trajectories with unicycle dynamics on a 2D plane. The system was initialized with non-zero velocity, bounded deceleration, and bounded yaw rate.

It is important to note that, due to Definition 12, the computation of ICS space requires knowledge of future control trajectories of all obstacles.

Definition 14. A *contingency plan* is a control sequence that an agent can execute that is guaranteed to avoid ICS space.

Definitions 15 & 16 below introduce concepts that will aid in the analysis of Problem 5. While sufficient to derive the results of this chapter, these definitions will be refined in §4.4.5 and §4.4.6 to generalize how dynamic systems can be described.

Definition 15. For a state $\mathbf{x}$ and path P, the *stopping path* $SP(A(\mathbf{x}), P)$ is the minimal set of agent states A must occupy while coming to zero velocity from $\mathbf{x}$ along P (see Figure 4.2a).

Definition 16. For a given agent state $A(\mathbf{x})$ let $\mathcal{P}$ be the set of all followable paths and let I be its index set. Define the *stopping region* $SR(A(\mathbf{x}), \mathcal{P})$ as the disjoint union of all SPs over $\mathcal{P}$ (see Figure 4.2b):

$$SR(A(\mathbf{x}), \mathcal{P}) = \bigsqcup_{i \in I} SP(A(\mathbf{x}), P)$$

The disjoint union in Definition 16 preserves information about the originating set, which is useful for algorithmic operations involving SRs (as in Chapter 5).

4.4 Theory

This section will derive the main results of the chapter. First the specific conditions are derived for which solutions to Problem 5 can make non-collision guarantees with and without agent coordination. An explicit problem formulation for Problem 5 is given and it is

shown that the complexity of the problem is directly influenced by system dynamics via the coordination requirement. The section is closed with a discussion of the results.

To aid with the derivation, Assumption 1 makes explicit the assumption in Problem 5 that agents will maintain and invoke contingency plans during navigation

Assumption 1. *Agents in a multi-agent system will compute and maintain motion plans independently of interaction effects with other agents to use as collision avoidance maneuvers.*

Conjecture 1 posits that the stopping path is the unique coordination-free contingency plan available to agents under Problem 5.

Conjecture 1. *Stopping paths are the unique category of motion plan that can enable coordination-free collision avoidance under Problem 5.*

Conjecture 1 is often a reasonable assumption, and in this work it is assumed to be true. It is an interesting, and open, question whether and how Conjecture 1 could be verified in a multi-agent system. Chapter 6 discusses this as a point of potential future work.

4.4.1 Relating dynamics to coordination

This section derives the relationship between system dynamics and the requirement for agent coordination under Problem 5.

Lemma 4. *In order to guarantee that any system can remain collision free, at least some reachable subset of ICS space must be computable from any state.*

Proof. This follows from Definition 13. In order to guarantee that a system can remain collision free, it must only move into states that are *not* in ICS space. To do that, some subset of the *complement* of ICS space must be computable. This equivalently means that some subset of ICS space must be computable. □

Lemma 5. *In Problem 5, ICS space is not computable without agent coordination.*

Proof. This follows from Definitions 12 & 13: in order to compute ICS space the future control trajectories of *all* agents must be fully observable. However, under Problem 5, those are not fully observable without coordination among the agents. □

Theorem 4. *In Problem 5, coordination is required in general for each agent to maintain the ability to remain collision free.*

Proof. It follows from Definition 14 that all agents must have a contingency plan at all times in order to guarantee the system can remain collision free. By Lemma 4 and Definition 14, computing a contingency plan requires computing some subset of ICS space. By Lemma 5, computing ICS space requires coordination among agents. □

The SR concept from Definition 16 will now be used to frame the coordination requirement in terms of system dynamics.

Lemma 6. *Consider a system with agent set* $\mathcal{A} = \{A_1(\mathbf{x}_1), \ldots, A_n(\mathbf{x}_n)\}$ *and a set of followable path sets* $\mathbb{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$*, where each* $\mathcal{P}_j$ *contains the followable paths* $\{P_1, \ldots, P_m\}$ *for agent* A_j*. Let* i *and* j *be elements of the index set* $\{1, \ldots, n\}$*. For each* j *and all* i*, define the union of obstacle SRs as:*

$$\mathcal{SR}_j(\mathcal{A}, \mathbb{P}) = \bigcup_{i \neq j} SR(A_i(\mathbf{x}_i), \mathcal{P}_i)$$

For each j*, also define an intersection set* $\mathcal{Q}_j$*:*

$$SR(A_j(\mathbf{x}_j), \mathcal{P}_j) \cap \mathcal{SR}_j(\mathcal{A}, \mathbb{P}) = \mathcal{Q}_j$$

If and only if for all j there exists an SP such that $SP(A_j(\mathbf{x}_j), P_j) \cap \mathcal{Q}_j = \varnothing$, then it can be guaranteed without coordination that no $\mathbf{x} \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is an ICS.

Proof. First, testing SP disjointness with respect to the *union* of obstacle SRs is valid by Definitions 15 & 16 because SPs and SRs are independent of the states of other agents. Second, the statement that for all agent indices a stopping path exists such that $SP(A_j(\mathbf{x}_j), P_j) \cap \mathcal{Q}_j = \varnothing$ implies that all agents have disjoint stopping paths available. By Definition 15 this means that all $A_1(\mathbf{x}_1), \ldots, A_n(\mathbf{x}_n)$ can come to a stop without intersecting. Therefore, *if* the condition in the theorem holds, collision is not inevitable in any state $\mathbf{x}_1, \ldots, \mathbf{x}_n$. This is also true *only if* the condition holds because if $SP(A_j(\mathbf{x}_j), P_j) \cap \mathcal{Q}_j \neq \varnothing$, then under assumption[2] of Conjecture 1, agents would otherwise all need to find a *unique* set of SPs such that collision is avoided if all agents execute exactly that set of contingency plans. But because remaining collision free would now rely on each agent executing exactly one control sequence whose feasibility directly depends on every other agent executing exactly one control sequence, the system now requires coordination by Definition 10, and so is no longer coordination free. □

The assertion in the proof of Lemma 6 that finding a unique set of non-disjoint SPs induces a coordination requirement is not just a semantic distinction but probably also brings a commensurate addition of computational complexity: The general problem of identifying a unique assignment of SPs is likely reducible to a Unique-SAT problem, which is coNP-Hard [28]. Conjecture 2 captures this, and Chapter 6 discusses this as a point of potential future work.

[2]Breaking this assumption weakens the logical connection in Lemma 6 from a biconditional (*if and only if*) to a material condition (*if*).

Conjecture 2. *There is no efficiently computable (i.e. polynomial-time) solution to identifying a unique set of disjoint stopping paths in a system that does not exhibit SP disjointness.*

Theorem 5 now formalizes the result of Lemma 6 and establishes a condition under which a general system is guaranteed to be able to remain collision free without coordination.

Theorem 5. *A multi-agent system is guaranteed to be able to remain collision free without coordination if and only if each agent has stopping path disjoint from the stopping regions of all other agents.*

Proof. This follows directly from Lemma 6. By Definition 13 a system is capable of remaining collision free if and only if it is not in an inevitable collision state. By Lemma 6, this is true if and only if all agents have stopping paths disjoint from the stopping regions of all other agents. □

Clearly, the result of Theorem 5 is closely tied to the notion of ICS space, and it would be in part equivalent to state that under certain conditions, it is impossible to compute *any subset* of ICS space without knowing the future actions of other agents. This idea is related to the sufficient safety condition for partial motion plans derived by Petti and Fraichard [121], which states that if the final state of a collision-free trajectory is not an ICS, then no state along the trajectory is an ICS.

The following definition is made for convenience:

Definition 17. The condition that satisfies Theorem 5, that all agents have at least one stopping path disjoint from the stopping regions of all other agents, is called *SP disjointness*.

Theorem 5 states that coordination is unnecessary under SP disjointness. But how, in practice, could agents maintain that property without coordination? Trivially, if an agent

modulates its dynamics such that it can always come to a stop without *possibly* intersecting the path of any other agent, then the property is satisfied. Agents need no knowledge of the plans of other agents for this; they simply need knowledge of the dynamics of the system. This is the approach taken, for example, by Mazer et al. [102] in the Ariadne's Clew algorithm. For most inertially constrained systems, however, this behavior would be too conservative to be useful. Worse, it is possible to specify initial conditions in such a system such that it is not possible to satisfy the property required by Theorem 5 (see §4.4.4). This is why, for example, the algorithm for multi-agent collision avoidance for inertially constrained systems given by Bekris et al. [19] requires coordination in order to maintain its guarantees.

But if Bekris et al. [19] require coordination, why is it that van den Berg et al. [152] do not? The ORCA framework they present is an efficient collision avoidance algorithm based on the velocity obstacle (VO) representation that guarantees non-collision for very complex scenes without the need for agent coordination. One of the immediate and natural consequences of Theorem 5 is that the requirement for agent coordination for certain systems can always be dropped. Theorem 6 explores and establishes this possibility:

Lemma 7. *In inertially unconstrained systems* $SR(A(\mathbf{x}), p) = A(\mathbf{x})$.

Proof. That A can instantaneously stop means that the minimal set of states A must occupy while coming to a stop along *any* path P is exactly $A(\mathbf{x})$. This implies further that $\bigcup_{P \in \mathcal{P}} SP(A, P) = A(\mathbf{x})$ which is equal to $SR(A)$ by Definition 16. □

Theorem 6. *An inertially unconstrained multi-agent system that is not currently in collision is guaranteed to be able to remain collision free without coordination.*

Proof. Lemma 7 implies that for inertially unconstrained systems, SP disjointness holds for all non-collision states. By Theorem 5, such a system is guaranteed to be able to remain

collision free without coordination. □

Note that any system described by the VO formulation is necessarily an inertially unconstrained system and therefore Theorem 6 applies to it. This means the fact that Bekris et al. [19] required coordination for their solution and van den Berg et al. [152] did not is directly a result of the system dynamics they employed: the former described inertially constrained systems, and the latter unconstrained systems.

At a deeper level, these results speak to the fundamental problems encountered when ignoring inertial constraints in dynamical systems. The VO representation, for instance, is an often used approximation for mutual collision avoidance in multi-agent systems because of its simplicity and elegance. However, Theorem 6 implies that the guarantees it makes are invalid for systems with inertial constraints. This is demonstrated empirically by Wilkerson et al. [161] who showed that using the VO representation in a constrained system can result in collisions, even with theoretical non-collision guarantees. §4.6 in the Appendix gives a formal proof of this fact.

4.4.2 Collision avoidance as a decision problem

In any practical instance, Problem 5 would be a hybrid decision making/motion planning problem, so its model would also take a form similar to the hybrid models mentioned in §4.2. Let $\mathcal{R}$ be some sufficiently dense discrete roadmap approximation to $\mathbb{S}$ (for instance, a Stochastic Motion Roadmap Alterovitz et al. [10]), where "sufficiently dense" means dense enough to allow solutions to be found. Assume each agent is initialized at some vertex of $\mathcal{R}$, and assume all agents plan at a uniform and aligned frequency. Assume all agents have full knowledge of system dynamics and of $\mathcal{R}$, and that some efficient method for computing SRs exists (see §4.5.1). Assume that agents are capable of coordinating their actions if they so

choose (subject to the restrictions outlined in Problem 5) in a way similar to that presented in Bekris et al. [19], in which agents negotiate joint contingency plans.

Define G as an instance of Problem 5 in its most general as a POSG:

Problem 6. Let $G = (\mathcal{A}, O, \mathcal{C}, c_0, \mathcal{A}, T, \Omega, R)$, where:

- $\mathcal{A}$ is a set of agents whose states include intent, which defines how the agent's internal policy affects its actuation
- O is a set of observations (mapping of observable agent states $\mathcal{R}$)
- $\mathcal{C}$ is a set of configurations of the system (mapping of full agent states $\mathcal{R}$)
- c_0 is a designated initial configuration
- $\mathcal{A}$ is a set of actions that enable transition between any two vertices on $\mathcal{R}$
- $T : \mathcal{C} \times \mathcal{A}^k \times \mathcal{C} \to [0, 1]$ is the transition probability function, where $T(c, a_1, \ldots, a_k, c')$ defines the probability that configuration $c' \in \mathcal{C}$ is reached from configuration $c \in \mathcal{C}$ when each agent i chooses an action a_i
- $\Omega : \mathcal{C} \times I \to O$ is the observation function, where $\Omega(c, i)$ is the observation made in configuration c by agent i. The observation of one other agent may include the result of a negotiation (a joint contingency plan); for all others the observation includes a distribution over contingency plans
- $R : \mathcal{C} \times \mathcal{A}^k \times I \to \mathfrak{R}$ is a reward function, where $R(c, a_1, \ldots, a_k, i)$ is the reward for agent i in configuration c when agents take actions $a_1, \ldots, a_k$

4.4.3 Main result

This section shows that Problem 6 can be reduced to an MDP provided the non-coordination guarantee of Theorem 5 holds, and that it otherwise remains a POSG.

Lemma 8 is given to aid in the derivation of the main result:

Lemma 8. *An agent A can assume arbitrary policies for all $O \in \mathcal{O}$ and maintain the non-collision guarantee provided the assumed policies maintain SP disjointness.*

Proof. This follows from Theorem 5. To guarantee the ability to remain collision free, the action an agent takes is irrelevant so long as SP disjointness is maintained[3]. □

Now, the reduction of G to an MDP under SP disjointness:

Theorem 7. *Under SP disjointness, G can be modeled as an MDP.*

Proof. Lemma 8 states that the ability to remain collision free can be assured whenever an agent A assigns arbitrary polices to other agents so long as SP disjointness is maintained. This means A does not need to *observe* anything about the other agents beyond what is already observable in order to plan and execute motions in the presence of other agents. A is free to assume arbitrary policies and full observability in future actions and a shared reward function; it is only important that in the current state the agents can rely on each other not to violate SP disjointness, and, as discussed previously, this can be done strictly with knowledge of system dynamics. Under the assumption of full observability, A can then incorporate the state of other agents into its own transition function, effectively centralizing the decision process. Thus, G is now equivalent to a fully-observable, centralized, single-agent system. In other words, G is an MDP. □

[3]The principle of Lemma 8 is intimately related to that exploited by the ORCA framework.

Now, if SP disjointness cannot be guaranteed, G must remain a POSG:

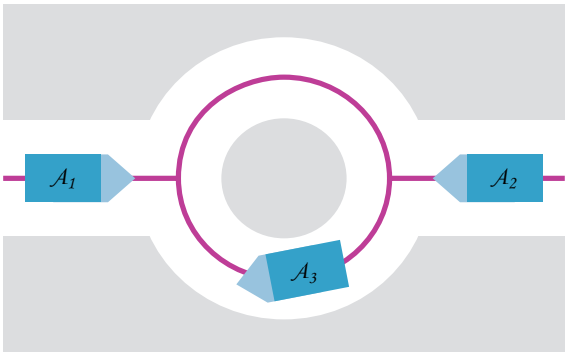
Theorem 8. *Without SP disjointness, G must be modeled as a POSG.*

Proof. In order to compute a solution satisfying Problem 5, agents must reason in some way about the future actions of other agents, but Theorem 5 says that in the absence of SP disjointness, coordination among agents is required to do so, which means centralized control cannot be assumed. In the worst case, the agent SRs may intersect in a way that requires more than two agents to coordinate. Due to communication limitations, however, this necessarily induces partial observability of the intent of at least one of the agents. In addition to a partially observable world, reasoning about future actions involving other agents also requires consideration of non-shared reward functions because they are what determine the distribution over future actions. Under these conditions, the decision process is decentralized, multi-agent, and partially observable with non-shared reward functions. By definition this is a POSG. □

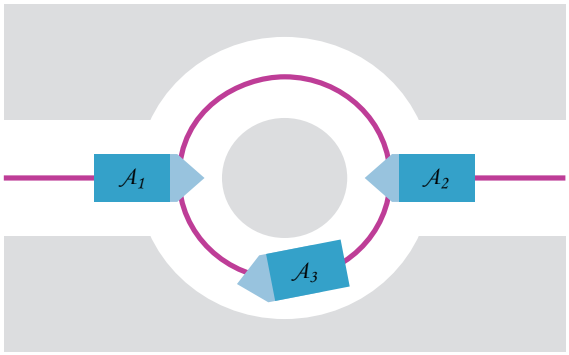
Theorems 5, 7, & 8 lead to Assertion 1, which is the thesis of this chapter:

Assertion 1. *Interaction effects in multi-agent systems can be* factored out*, i.e. addressed independently, of the navigation problem provided that SP disjointness can be maintained independently by all agents. The process of doing so is called* factoring interaction effects*.*

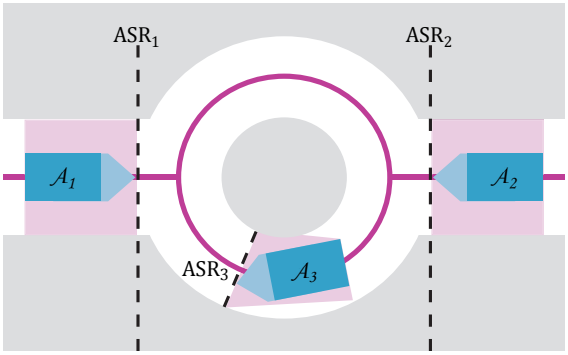
To summarize, this section has shown that system dynamics alone can be responsible for moving a problem between two types of problem models. This result demonstrates that the dynamics of a system can fundamentally change both the complexity class and model space of the problem.



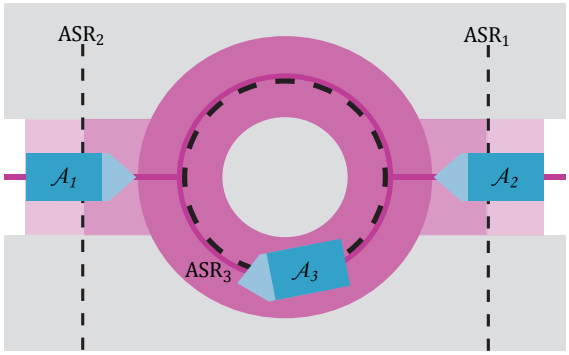
(a) A simple multi-agent system: agents A_1, A_2, and A_3 are moving along shared paths and must navigate around each other.



(b) At the branching point in the path, A_1 and A_2 must make a decision about which branch to follow.



(c) In an inertially unconstrained system, the SRs are invariant to the state of the system, and only extend longitudinally to the extents of the agents. The pink regions illustrate the extent of the SR along the path being followed. In such a scenario, the SP disjointness property holds for any initial state.



(d) An inertially constrained system with a high initial velocity results in an initial system state without a guarantee of SP disjointness. SR_1 and SR_2 extend beyond the median, and SR_3 circles the median. In such a scenario, there is significant overlap in the SRs for each agent, indicated by the darker shaded regions.

Figure 4.3: Exemplar problem. SRs are indicated as exaggerated pink regions for each agent with dashed lines indicating extents.

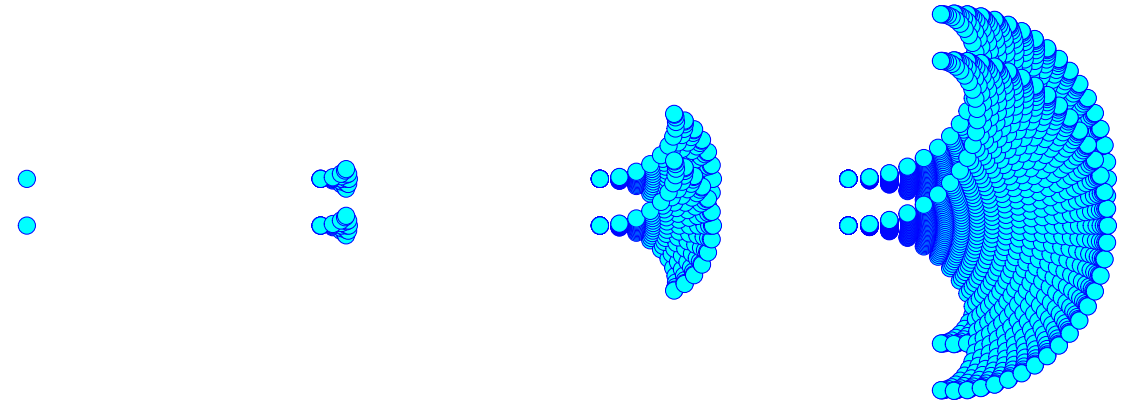
4.4.4 Exemplar problem

This section presents a simple multi-agent system that can be manipulated in certain ways to clearly demonstrate the ideas of this chapter. The problem is as follows: three agents A_1, A_2, and A_3 move along a fixed path network that has a split around a single median, shown in Figure 4.3a. A_1 and A_2 desire to make it past the median, and A_3 desires to stay around the median. As A_1 and A_2 traverse the path, they reach a point where they must make a decision about how to proceed (Figure 4.3b). Note that because only the collision avoidance problem is of concern, it is not a criterion for success that the agents can make it past each other successfully. Success only requires that they remain collision free, so, for instance, a deadlock situation satisfies the requirements, even if it is not the most desirable outcome.

Suppose the agents occupy an inertially unconstrained system. In this case, their SRs are disjoint unless or until they actually collide. Theorem 6 guarantees that, for any initial velocity, they can all proceed without coordination while maintaining the guarantee that collision is not inevitable (Figure 4.3c). On the other hand, assume the agents occupy an inertially constrained system. For a sufficiently high initial velocity, none of the three agents have an SP that can be guaranteed to be disjoint of all other SRs (Figure 4.3d). It should be clear both by inspection and by Theorem 5 that maintaining any non-collision guarantee is *only* possible in this case if they somehow coordinate.

4.4.5 Stopping regions with reference velocities

A useful extension to the SR model is to explicitly incorporate a reference velocity. This can be used to incorporate environmental influence into SR computation, or, by introducing an artificial reference velocity, it can simplify SR computation in systems with high absolute and low relative velocities. This is due to the fact that stopping regions defined in terms of



(a) $v_r = v$ (b) $v_r = 0.66v$ (c) $v_r = 0.33v$ (d) $v_r = 0$

Figure 4.4: For a given agent state $A(\mathbf{x})$, reference velocity v_r, and set of followable paths $\mathcal{P}$, the stopping region $SR(A(\mathbf{x}), v_r, \mathcal{P})$ is the SR computed with respect to the reference velocity v_r. This figure illustrates SRs for various reference velocities in a two agent system. The two disc agents are traveling on a 2D plane with the same velocity v. In (a) the reference velocity is taken as v, so the stopping regions are the agents themselves. In (b)-(d) the reference velocity is taken as progressively smaller fractions of v. Here agents obey the same dynamics as in in Figure 4.2. It is important to note that these illustrations are 2D projections of 3D swept volumes, so the overlap in SRs is not quite as severe as it looks.

relative velocities can be much smaller than their absolute counterparts. Extending stopping regions in this way requires only minor definition changes.

To introduce the reference velocity term, the reference frame used in Problem 5 is extended to include a velocity:

Definition 18. A *reference velocity* is a velocity component of the reference frame with respect to which agent velocities are measured for SP and SR computation.

Rather than computing a stopping velocity with respect to zero velocity in the reference frame, agents are allowed to choose any reference velocity. Naturally, this introduces a dependence of the coordination requirement computation on the reference velocity. Lemma 9 extends Theorem 5 to explicitly take into account the notion of a reference velocity:

Lemma 9. *A multi-agent system is guaranteed to be able to remain collision free without*

coordination if there exists a reference velocity such that Theorem 5 holds.

Proof. Suppose there exists a reference velocity $\mathcal{V}$ such that Theorem 5 holds. Let $\mathcal{V}$ be the velocity component of reference frame. Theorem 5 can now be applied. □

Stopping regions with a reference velocity can now be defined:

Definition 19. For a given agent state $A(\mathbf{x})$, reference velocity v_r, and set of followable paths $\mathcal{P}$, the *stopping region*, denoted $SR(A(\mathbf{x}), v_r, \mathcal{P})$, is the SR computed for the reference velocity v_r (Figure 4.4).

The stopping path can defined similarly. Note that there is no fundamental incompatibility between the results derived in this chapter and this extended notion of stopping regions; the reference velocity chosen for any problem is arbitrary to begin with. Formally including it in the definition simply acknowledges that fact. This is useful in systems where stopping regions can be computed in terms of *relative* velocities between agents, because it means that all agents can choose reference velocities independently.

4.4.6 Generalizing to soft stopping regions

For systems with low absolute and low relative velocities (such as pedestrian navigation), low-energy collisions may be permissible, or even unavoidable, and the stopping region concept can be extended to *soft stopping* regions. A soft stopping region is a region in which an agent will either come to a stop or enter into a low-severity collision, where severity is measured by *collision-induced velocity change* ΔV [64]. The definitions and lemmas below establish this concept.

Definition 20. For a given agent state $A(\mathbf{x})$, reference velocity v_r, collision velocity v_c, and set of followable paths $\mathcal{P}$, the *soft stopping region* $SR(A(\mathbf{x}), v_r, v_c, \mathcal{P})$, is the SR computed for the reference velocity v_r and the target velocity of the stopping paths set to v_c.

Definition 20 is only a minor change from Definitions 16 & 19: the target velocity of the stopping path is now just a parameter. It would seem this could be rolled into the reference velocity, but it is actually specifying something fundamentally different because it is dependent on the *control policies* of the other agents, which complicates the determination of SP disjointness: if it is impossible to estimate v_c for other agents, then it is also impossible to estimate SP disjointness, and the absence of the coordination requirement can no longer be guaranteed. In order to use soft stopping regions effectively, then, it is necessary that agents have some mechanism to measure, or estimate from observation (such as described in §4.4.7), acceptable values for v_c. If such information is available, one simple approach to dealing with multiple agents is to compute v_c such that it satisfies the ΔV threshold for any potential collision.

4.4.7 Incorporating the environment into dynamics computation

In this chapter, agents are assumed to have no knowledge of how others move aside from their dynamic capabilities. This could be generalized to give all agents access to a set of pre-defined rules or *social laws* [130]. For instance, right-of-way traffic rules could be defined that allow the need for coordination to be removed from more complex interactions because the rules guarantee the existence of contingency plans, which enable the SP disjointness condition to hold. In fact, the SP disjointness condition itself is essentially a pre-defined rule, but one that is derivable strictly from the physical properties of the system. But if all agents have access to a shared set of rules and the environment is marked in a way that

unambiguously indicates which rules to follow at any given time, then it is straightforward to incorporation this information into the computation of the stopping regions.

Let Γ be a set of rules where each rule $r \in \Gamma$ is a set of parameterized dynamic constraints. A rule r is applied to an agent A if the constraints of r are applied to the motion model M of A. Assume all agents have full knowledge of Γ. Let $M_\Gamma : (A_i, O) \mapsto \gamma_i$ be a mapping of an agent/observation pair to a motion model with a set of active rules $\gamma \subseteq \Gamma$. In other words, M_Γ is the motion model a given agent follows assuming the rules indicated by a given observation are followed.

It is now straightforward to incorporate environmental rules into SR computation: there is simply an additional step during control computations that modifies the dynamic models of observed agents according to the active rules. These modified models M_Γ now affect how SRs are computed, and the rest of the problem remains the same.

4.5 Practical concerns

For the results of this chapter to have impact in practice, there must exist efficient methods for computing SRs, testing for collision, and producing motion controls. For motion controls §4.2 gave results showing P-time complexity for the $\mathbb{R}^2$ or $\mathbb{R}^3$ piano mover's problem. In cases where just online feedback control suffices, there exist many linear- or constant-time control laws that can be used [119].

The remainder of this section provides a brief survey of techniques that could be used during SR computation and collision detection.

4.5.1 SR computation

The problem of SR computation is a specific instance of the more general problem of computing reachable regions of state space. Ó'Dúnlaing [116] gave a P-time algorithm for point agents and obstacles moving in one dimension under inertial constraints. Chapter 2 generalized the result to convex agents and obstacles moving along fixed paths in $\mathbb{R}^2$ while maintaining P-time complexity. Sontges and Althoff [135] described an online spatial decomposition approach that conservatively approximates the region of reachable space.

In higher-dimensional applications or with articulated agents, the problem of computing reachable regions becomes intractable, so approximation techniques can be employed. Valtazanos and Ramamoorthy [151] employed pre-computed reachable region templates that are composed online in order to plan efficiently. Allen et al. [7] similarly employed machine learning to efficiently approximate reachable region computation online.

Geometrically, computing SRs is the problem of computing swept volumes [6, 15]. In general, computation of swept volumes to arbitrary precision cannot be done online, but many techniques have been developed to allow efficient and practical approximations [81, 141, 156].

In any practical application the method chosen for computing SRs will necessarily be instance-specific, but a variety of tools exists for performing these types of computations efficiently.

4.5.2 The collision detection problem

Many motion planning and motion control techniques require explicit collision detection. Sampling-based techniques in particular, which are commonly used in practice, tend to spend a majority of their computational budget on collision detection [87]. Because of this,

and its natural application in video gaming, graphics, and simulation, there is a significant literature on collision detection techniques and theory [65, 72, 160]. In motion planning, collisions are usually tested between 2D or 3D convex polygons using tools from computational geometry [128].

For the 2D case, linear- and logarithmic-time intersection test algorithms have been known for some time [35]. To achieve those low theoretical complexities, however, a significant amount of bookkeeping is necessary, often to the extent that the bookkeeping dominates running time. Simpler algorithms based on the hyperplane separation theorem [39] can be much faster in practice despite P-time complexity.

Often it is also desirable to have a measure of minimum separating distance in addition to an intersection test. In the 2D case, again, this can be accomplished with linear time complexity [52]. But these algorithms are necessarily more complex than strict intersection tests, so they are often much slower, which is especially burdensome in higher dimensions. In many situations, collisions are tested *over time* which means information computed during one time step may be used to inform computations during the next. Closest feature tracking techniques, pioneered by Lin and Canny [93], exploit such temporal coherence to achieve expected constant time performance.

When the geometric models are static or pre-defined, bounding-volume hierarchies [41] and spatial decompositions [61] can enable extremely fast collision detection and distance approximations between objects of high geometric complexity, with spatial decompositions additionally allowing efficient online construction.

Depending on the method chosen for motion control, the choice of collision detection method may play an important part in maintaining real-time capability for a mobile agent. Thankfully, the field of collision detection is well studied and understood, and there exist

many established techniques for efficient collision detection.

4.6 Conclusion

This chapter presents results showing that system dynamics can have a direct impact on both the theoretical complexity and solution space of multi-agent collision avoidance problems. The result is based on the fact that a requirement for agent coordination in a multi-agent system can fundamentally alter the problem model necessary to find a solution, and it was shown that system dynamics alone can add or remove this requirement. The proof of this assertion is constructive in nature, which allows the coordination requirement to be tested empirically, which implies that it is possible for agents to exploit knowledge of the presence or absence of the requirement. The process of enforcing the absence of this requirement so that the full space of agent interactions need not be considered during navigation is named *factoring interaction effects* by Assertion 1, and an exemplar problem was given to demonstrate the results. In addition, the Appendix provides a proof that inertially unconstrained models cannot conservatively approximate inertially constrained systems, and it provides a re-formulation of the velocity obstacle concept within the ICS family of representations.

As covered in §4.2 the complexity difference between MDP's and POSG's is staggering, with the former falling into complexity class P, and the latter into NEXP in the cooperative case or NEXP^{NP} in the non-cooperative case. The fact that the complexity of the system can be manipulated to keep it within a tractable realm simply by controlling the dynamics is both surprising and powerful, and may provide insight into how humans are capable of efficiently and successfully navigating complex, multi-agent systems. In the case of roadways, for instance, the environment constrains the set of motions to such an extent that virtually any forward motion ensures progress toward the goal, so optimality of the plan is of little

value. Instead, if agents prioritize maintaining SP disjointness in the system, a planning problem that is, in principle, wildly intractable becomes comfortably tractable.

Conceptually, this chapter deals with the fundamental question of how to appropriately model certain problems involving real-world interacting agents. Modeling as optimal decision making processes enables elegant formulations but requires the use of problem models that can be intractable to actually solve. This has serious practical implications that tend to be overlooked in academic literature. Daskalakis and Papadimitriou [37] raised this issue in a study of complexity in large, multi-agent systems by posing the question: "How can one have faith in a model predicting that a group of agents will solve an intractable problem?" In the realm of multi-agent systems, this chapter suggests that such questions may be avoided by employing models that allow agents to independently modulate the problem complexity.

Appendix

This section will prove that the velocity obstacle representation cannot conservatively approximate inertially constrained systems, a result that has important ramifications for safety guarantees in real-world applications. In addition, it will be shown the the velocity obstacle representation belongs to the family of ICS representations using the *inevitable collision obstacle* (ICO) concept. First the ICO and velocity obstacle representations are covered with relevant definitions and notations, after which proofs will be given.

The inevitable collision obstacle

Here the ICO is introduced and relevant notation given.

Definition 21. An *inevitable collision obstacle* (ICO) is the set of states of an agent A that result in collision with B_i for any control sequence ϕ is applied to A:

$$\mathrm{ICO}(B_i) = \{\mathbf{x} \mid \forall \phi, \exists t :: A(\phi(\mathbf{x}, t)) \cap B_i \neq \varnothing\}$$

The ICO is closely related to the ICS concept, both of which were introduced by Fraichard and Asama [49].

The velocity obstacle

This section recalls the velocity obstacle and relevant properties. We use the definitions from Fiorini and Shiller [45], and the reader is referred to that work for more detail[4].

In this section, assume $t \in T$, where $T = [0, \infty)$ is a finite time horizon. Let Φ_v be the

[4]To avoid problems in dealing with dynamic constraints Wilkie et al. [162] defined *generalized velocity obstacles* that are derived in control space rather than velocity space.

set of feasible velocity commands for A, and let $\phi_v(\mathbf{x}, t)$ denote the state of A after constant velocity v is applied to initial state $\mathbf{x}$ for a time t.

Definition 22. The *velocity obstacle* for A due to O_i ($\mathrm{VO}_{A|O_i}$) is the set of velocities such that A at some point enters into a collision state with O. In other words, given initial state $\mathbf{x}$, and for all feasible velocity commands $v \in \Phi_v$ there is a collision at some time $t \in T$ between $A(\mathbf{x})$ and the state space obstacle B_i due to O_i:

$$\mathrm{VO}_{A|O_i} = \{v \mid \exists t :: A(\phi_v(\mathbf{x}, t)) \cap B_i \neq \varnothing\}$$

Velocity obstacles and inertially constrained systems

Lemma 10. *The velocity obstacles representation cannot guarantee collision avoidance in inertially constrained systems.*

Proof. By Definition 22, the complement of the velocity obstacle is exactly the set of all velocities that, when instantaneously applied, would avoid collision. However, controlling to a velocity instantaneously is impossible in an inertially constrained system. Therefore, the complement of the velocity obstacle is unreachable, and by Lemma 4, it cannot be used to guarantee non-collision. □

Velocity obstacle and inevitable collision obstacle equivalence

The reader will note the similarity between Definitions 21 & 22, and work by Shiller et al. [129] suggests that a deeper relationship exists. The proof proceeds by exploiting the similarity and showing that ICO computations are both necessary and sufficient in order to compute the VO.

Definition 23. A *velocity* ICO (ICO_v) for a given state space obstacle B_i is an ICO computed over the velocity control trajectory set Φ_v:

$$\text{ICO}(B_i)_v = \{\mathbf{x} \,|\forall \phi_v, \exists t :: A(\phi_v(\mathbf{x}, t)) \cap B_i \neq \varnothing\}$$

Lemma 11. *Computing a velocity obstacle is exactly equivalent to computing an inevitable collision obstacle over a restricted control space.*

Proof. For a given obstacle O_i and corresponding state space obstacle B_i, use Definition 22 to perform a variable rewrite on the definition of a velocity ICO (Definition 23):

$$\begin{aligned}\text{ICO}(B_i)_v &= \{\mathbf{x} \mid \forall \phi_v, \exists t :: A(\phi_v(\mathbf{x}, t)) \cap B_i \neq \varnothing\} \\ &= \{\mathbf{x} \mid \forall \phi_v, v \in \text{VO}_{A|O_i}\}\end{aligned}$$

Thus, the $\text{ICO}(B)_v$ and $\text{VO}_{A|O}$ are equivalent, which means that the velocity obstacle representation is equivalent to the ICO representation over a restricted control space. □

The result of Lemma 11 provides a simple but formal unification of two common techniques for collision avoidance under the same theoretical framework: that velocity obstacles are exactly inevitable collision obstacles over a restricted set of the inputs.

Chapter 5

Selective determinism for guided collision avoidance

Synopsis: *This chapter develops a framework for performing guided collision avoidance in multi-agent systems. Necessary components of the framework are presented as generic algorithms whose solutions must be computed or conservatively approximated in any problem-specific algorithm. The framework specifies that algorithms consist of two primary parts: a local controller that maintains SP disjointness, and a global controller that guides the controller within the space of SP disjoint controls. As demonstration, a problem-specific algorithm is developed and analyzed under the framework.*

5.1 Introduction

As established in Chapter 3 computing optimal controls is a generally intractable problem, and although it may admit efficient approximations, those approximations may or may not be guaranteed to satisfy required problem constraints, notably collision avoidance. However, for a broad class of multi-agent systems, Chapter 4 established explicit, computable conditions for which approximation techniques can be guaranteed to satisfy hard collision constraints. This chapter builds on those results to define the *Selective Determinism* (SD) framework that enables agents to perform goal-directed navigation in stochastic multi-agent systems. The framework allows the development of robust algorithms with guarantees on both non-collision and tractability that are impossible under optimal solution techniques.

§5.3 gives the general problem statement that the SD framework is intended to be used with, and §5.4 defines the SD framework in terms of generic algorithms that address that problem. §5.5 demonstrates the SD framework by developing a problem-specific solution to a *path network traversal* problem and analyzes the statistics of cost variation for SD solutions under variations in problem initial conditions. §5.6 concludes the chapter.

5.2 Related work

The SD framework builds on the ideas of shared autonomy, or priority blending, introduced in Chapter 3 and on the ideas of efficient problem models used in Chapter 4. This section provides a brief survey of control and planning techniques that use related ideas to deal with multi-agent systems.

In decentralized control theory *person-by-person* solution techniques are those in which each agent in a multi-agent system assumes arbitrary control strategies for all other agents

and optimizes its own actions according to the assumed strategies.[1] Described by Radner [123] and Marschak and Radner [101], the person-by-person optimal control strategy is similar to the notion of local optimality in optimization theory. As noted by Chapter 4, such strategies provide no guarantees on any global measure of plan quality, but they are typically computationally efficient. A complementary technique, known as the *designer's approach* [96, 164], views the control problem from the perspective of a system designer who attempts to solve the decentralized problem in a centralized fashion by computing control laws for all agents in open-loop. By using an open-loop strategy the designer's approach attempts to compute solutions that approach some kind of global optimality without incurring the computational cost of accounting for future observation input.

The designer's approach is related to decision process problem models known variously as open-loop techniques [158], unobservable MDPs [110], or non-observable MDPs (NOMDPs) [30]. Models in this family are otherwise standard partially observable MDPs that have only a single 'null' observation. As would be expected, the null observation function reduces the problem complexity significantly from that of the general partially observable case, though lacking the perfect observation information of an MDP, the NOMDP is still not fully tractable: Papadimitriou and Tsitsiklis [120] showed that the NOMDP decision problem is NP-Complete. In practice, however, randomized or approximating techniques can often be used to efficiently find reasonable solutions to NP-Complete problems. In general, one may note the resemblance between open-loop strategies described above and the rollout technique applied in Chapter 3. As shown by Bertsekas [23] these two techniques are closely related, with open-loop approaches being special cases of the more general family of rollout strategies.

[1] This type of strategy is employed by Theorem 7 in Chapter 4 to achieve complexity reduction in the guided collision avoidance problem.

Yu et al. [167] gave a theoretical treatment of the types of partially observable systems most suitable to open-loop planning techniques, noting that problems that can be decomposed into global and local action spaces are particularly well suited, and that even though optimality cannot theoretically be guaranteed, there is typically little loss of quality in practical robotic systems. From an empirical standpoint, Weinstein and Littman [158] showed that open-loop techniques can be very effective even in large-scale/high-dimensional robotics control problems, and Sunberg et al. [140] implicitly used a hierarchical NOMDP model to plan collision avoidance maneuvers for unmanned aerial vehicles.

This open-loop approach can also be very useful outside the pure control domain. Beard et al. [17] presented an open-loop approach to the mobile tracking and label assignment problem in dynamic stochastic domains. The goal was to perform mobile tracking while simultaneously ensuring that the sensor does not collide with the objects being tracked. The problem is cast as a POMDP with a cost function that has a blending of terms, one of which implicitly guarantees SP disjointness, and the problem as a whole is solved with an open-loop approximation.

In fully observable problems, when both global and local scopes can be expressed as linear algebraic systems, jointly optimizing global and local controls can still be unwieldy even if the problems are theoretically tractable. In these cases, the idea of solving global and local controls separately is implemented through null space optimization, which satisfies secondary system constraints within the null space of the primary solution. In this way global and local priorities can be combined without violating hard constraints [80, 137, 139].

5.3 Problem definition

The guided collision avoidance problem is defined as an extension to the variant of the reciprocal n-body collision avoidance problem, Problem 5, given in Chapter 4.

Problem 7. Given a desired goal state $\mathbf{x}^g$, how can an agent solve Problem 5 such that the computed controls blend the following two objectives in order of priority:

1. Preserve SP disjointness.

2. Make progress toward $\mathbf{x}^g$.

The solution to addressing the overall problem will be a Selective Determinism (SD) algorithm that uses theory derived in Chapter 4 to factor the problem into independent sub-problems that each solve one of the objectives. The key is the ability to factor out interaction effects by identifying coordination requirements among all agents. Interactions with agents who do not require coordination can be *factored*, i.e., the ego agent can assume arbitrary and independent deterministic[2] policies for the factored agents. These assumed policies must be self-preserving, but can otherwise be defined however is expedient: simple constant velocity models, or more sophisticated reactive models (e.g. the *Intelligent Driver Model* [149]). In some cases it may be desirable to reason selectively about interactions between agents under a decentralized joint-planning framework (e.g. Alterovitz et al. [10], Kaelbling and Lozano-Pérez [71], Kambhampati et al. [73]). This could allow sophisticated interactions to take place between small subsets of agents within a larger system. Alternatively, as will be demonstrated in §5.5, the planning agent can choose to simply take actions that always adjust its state such that the coordination requirement is never present.

[2]The name *Selective Determinism* refers to this ability to choose deterministic models for modeling agent behavior.

5.4 General solution

Note that structure of Problem 7 maps very well to the open-loop hierarchical planning type defined in Yu et al. [167] that was discussed in §5.2. The problem has a global action space for the goal direction sub-problem, and a local action space for the SP disjointness (Definition 17, Chapter 4) preservation problem. Further, as noted in §5.2, open-loop solutions can be formulated as types of rollout algorithms, which means that the constrained optimization framework defined in Chapter 3 provides an ideal solution technique. From this, and inspired by the general solution methodology given in Mahajan and Mannan [97], the SD framework will compute a solution to Problem 7 by performing the following steps at each decision point:

1. A global controller computes an error term E in the control space such that minimizing E will direct an agent to some goal
2. E is given to local controller that computes the control set $\mathcal{U}$ that satisfies SP disjointness
   (a) If $\mathcal{U} \neq \varnothing$, then the control $\mathbf{u} \in \mathcal{U}$ minimizing E is issued
   (b) Otherwise, $\mathbf{u}$ is computed such that SP disjointness can be restored

In a pure optimization-based approach, the sub-problems above would be solved jointly by a stochastic optimal controller, but, as covered in Chapter 3, there are no practical general solution frameworks for such problems. However, when the requirement for joint optimality is dropped, the SCIMP framework defined in Chapter 3 offers a rigorous method for computing solutions to precisely the class of priority-blended control problems that Problem 7 belongs to. The mapping is actually direct: the global controller described above computes the

SCIMP desired input $\mathbf{u}^d$, and the local controller performs the SCIMP optimizations of Equations 3.16 & 3.17. Because the local controller is responsible for maintaining collision avoidance guarantees and for blending priorities, this section will focus on outlining how it can be implemented. There are few requirements on the global controller, and for this chapter a generic, deterministic planning algorithm will be assumed to act as the global controller.

For the local controller to meet its requirements, it must test for and maintain SP disjointness, which requires the ability to compute factorizations (defined in Assertion 1). Computing factorizations depends on computing SRs, and, as discussed in Chapter 4, the computation of SRs is highly domain dependent. However, certain properties of the computation of those regions are universally required in order to maintain the guarantees on collision avoidance and tractability. This section describes generic algorithms that demonstrate those properties with the intent that these algorithms be tailored to individual problem instances. Thus, the SD framework is a collection of these generic algorithms that serve to guide the development of problem-specific solutions. The remainder of this section presents these generic algorithms.

5.4.1 Computing stopping regions

Recall from Definition 16 that an SR is a union over all feasible SPs, and that it is the SPs that are used to determine SP disjointness. This means that an algorithm computing an SR must do more than simply compute the reachable region of space for an agent: it must compute the set of SPs that make up the SR. The following lemma makes clear why this distinction is important.

Lemma 12. *Let SR_1 and SR_2 be stopping regions, and let $SR_1 \cap SR_2 = Q$. Let $SR_1' =$*

Figure 5.1: Two trivial stopping regions, each of which has only a single SP. Though the SRs have disjoint subsets, the SP disjointness property does not hold.

$SR_1 \backslash Q$ and $SR'_2 = SR_2 \backslash Q$. If $SR'_1 \neq \varnothing$ and $SR'_2 \neq \varnothing$, this implies that portions of SR_1 and SR_2 are disjoint, but it **does not** *imply SP disjointness.*

Proof. This can be shown by counterexample. Let SR_1 and SR_2 consist of single, straight line SPs, and assume they intersect at a single point. Clearly, in this case, SR'_1 and SR'_2 are non-empty and disjoint; however, neither SP is disjoint. In this case, SP disjointness does not hold, and therefore the implication that it holds in general for such cases does not hold. See Figure 5.1. □

Lemma 12 means that in order for SR computation to be useful for detecting and maintaining SP disjointness, information about the set of SPs must be preserved. This is why Definition 16 defines SRs explicitly in terms of a disjoint union. Algorithm 8 computes SRs while preserving path information.

Algorithm 8 computes SRs according to Definition 19 rather than the soft SRs of Definition 20 because, as noted in §4.4.6, the soft SR generalization imposes additional requirements on the observability of agent decision processes. The sub-routine on Line 3 computes the complete set of feasible paths that A may follow from start state $\mathbf{x}$ under motion model M, and the sub-routine on Line 5 computes the state space region swept out by A as it follows a path P to a terminal velocity, which here is the reference velocity of Definition 18, Chapter 4.

Algorithm 8 Compute an SR for a given agent state, reference velocity v_r, and motion model M (the set of feasible controls is assumed to be derivable from M). As a side effect, the control sequences that follow all SPs are computed.

1: **procedure** COMPUTESR($A(\mathbf{x}), v_r, M$)
2: $\quad\Phi, SR \leftarrow \varnothing$
3: $\quad I, \mathcal{P} \leftarrow ComputeFeasiblePathsWithIndexSet(A(\mathbf{x}), M)$
4: $\quad$**for** $i \in I$ **do**
5: $\quad\quad\phi_i, SP_i \leftarrow ComputeMinimumSweptPathWithControlSet(A(\mathbf{x}), v_r, P_i)$
6: $\quad\quad\Phi, SR \leftarrow \Phi \bigsqcup \phi_i, SR \bigsqcup SP_i$
7: $\quad$**end for**
8: $\quad$**return** I, Φ, SR
9: **end procedure**

For complexity, Algorithm 8 is at least linear in $|\mathcal{P}|$. Let $h(\cdot)$ bound the complexity of the sub-routine on Line 3 and let $q(\cdot)$ bound the complexity of the sub-routine on Line 5. The overall complexity can then be written as $O(h(\cdot) + |\mathcal{P}|q(\cdot))$. Clearly, however, this is not very informative without more information on the nature of $h(\cdot)$ and $q(\cdot)$. To ensure the tractability, it is very important that appropriate representations be chosen for SPs and SRs. In most cases these will be some form of approximating swept volumes, for which many efficient computation methods exist (see §4.5.1).

As defined, the algorithm assumes that the sets of paths and SPs are discrete and finite. In reality, these sets will almost always be infinite in size, and each feasible path may additionally be infinite in length. In certain problems it may be possible to define paths and SPs analytically and arrive at closed-form solutions for computing them; however, the set of instances where this can actually be done is almost certainly exceedingly small. Most practical applications will require some kind of discreteness and finiteness assumptions to be ensure computability, which means that SPs and SRs will not be computed exactly. The next section will describe how to make these assumptions such that factorization guarantees hold.

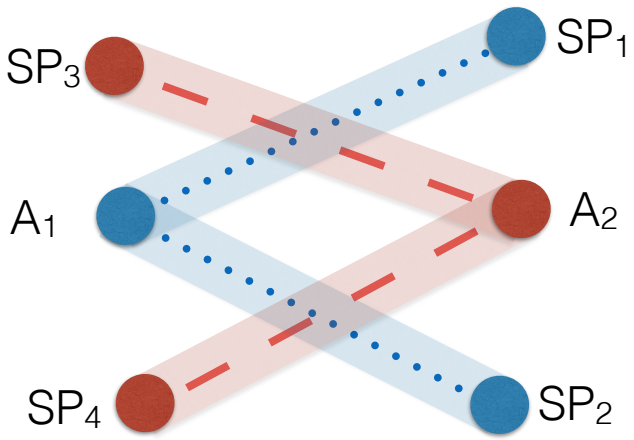


Figure 5.2: Illustration of the counterexample used in the proof of Lemma 13. Agents A_1 and A_2 have mutually disjoint SPs available to them, SP_1 and SP_4, and SP_2 and SP_3, but neither agent has a contingency plan that would not require coordination in order to guarantee non-collision.

5.4.2 Computing SP disjointness

Assuming that SPs and SRs can be computed, the problem statement for computing SP disjointness can now addressed. First consider the pairwise problem. A solution algorithm needs to collect all indices of SPs that intersect both SRs, and return the complement of that set. Algorithm 9 performs this operation.

Algorithm 9 The pairwise SP disjointness algorithm. Computes and returns a pair of index sets that contain all SP indices from SR that are disjoint from SR'.

```
1: procedure DisjointSPs(SR, SR')
2:     I ← index set from SR
3:     Q ← SR ∩ SR'
4:     I_Q ← index set from Q
5:     return I\I_Q
6: end procedure
```

Algorithm 9 simply implements a relative complement operation. As simple as this is, it is crucial to computing a correct factorization: it is *not* sufficient, in general, to check for the existence of a mutually disjoint subset of SPs between agents. Lemma 13 shows this.

Lemma 13. *For stopping regions SR_1 and SR_2 the existence of mutually disjoint SPs does*

not imply that SP disjointness holds between SR_1 and SR_2.

Proof. Proof proceeds by counterexample. Consider two agents A_1 and A_2, where A_1 has SP_1 and SP_2 in its SR, and A_2 has SP_3 and SP_4 in its SR. If SP_1 only intersects SP_3 and SP_2 only intersects SP_4, then the stopping paths SP_1 and SP_4, and SP_2 and SP_3 are mutually disjoint between the agents, but neither agent has any stopping path available that is guaranteed to be disjoint of the other's SR *regardless* of the other agent's future actions. This is a violation of the condition for non-coordination from Theorem 5, thus, SP disjointness cannot hold according to Definition 17. See Figure 5.2. □

Assuming index sets are sorted, the computational complexity is at least linear in the size of the SRs due to Line 5, but also strongly depends on the intersection operation on Line 3. Assuming the complexity of SR intersection is bounded asymptotically by some function $g(\cdot)$, the complexity is $O(g(\cdot)|SR|)$. As with Algorithm 8 it is assumed that representations are suitably chosen such that the intersection operation can be implemented efficiently. For certain problems, it may also be possible to further simplify the SP disjointness problem and formulate it purely as a decision problem, i.e., strictly decide whether a disjoint SP exists. In most systems, however, explicit information about the set of disjoint SPs will be useful.

Algorithm 10 The SP disjointness algorithm. Prunes all SPs ∈ SR that intersect SPs ∈ $\mathcal{SR}$ and returns the pruned SRs.

```
1: procedure PruneSR(SR, 𝒮ℛ)
2:     I ← GetIndexSet(SR)
3:     for SR_i ∈ 𝒮ℛ do
4:         I′ ← DisjointSPs(SR, SR_i)
5:         Remove all elements in SR with i ∈ I′
6:         I ← I\I′
7:     end for
8:     return I, SR
9: end procedure
```

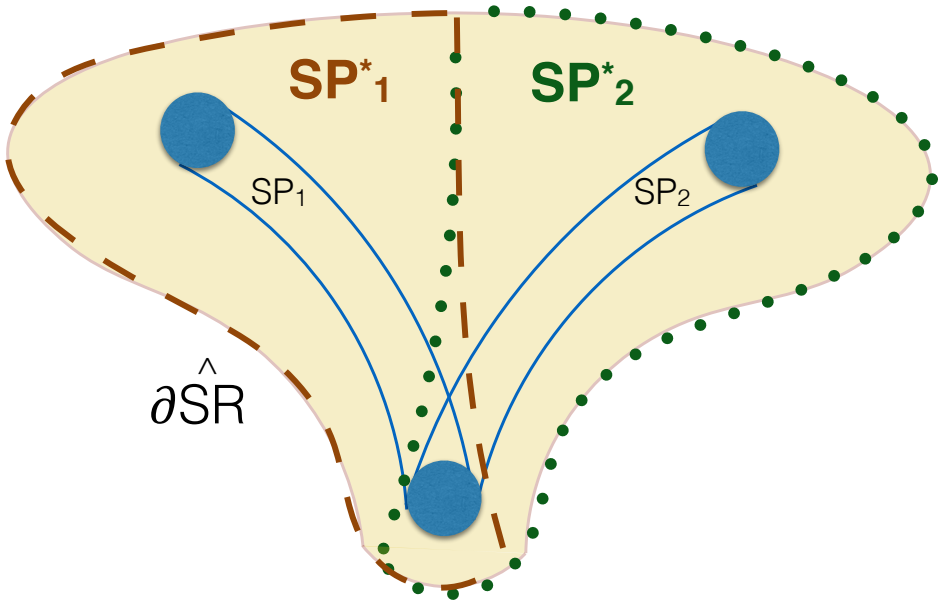


Figure 5.3: This figure illustrates the conservative SP disjointness computation described by Lemma 14. A conservative SR boundary $\delta\widehat{SR}$ it shown for a blue disc agent, along with stopping path subset that is made up of SP_1 and SP_2. Conservative stopping paths $SP_1^\star$ (red, dashed) and $SP_2^\star$ (green, dotted) are defined such that $SP_1^\star \cup SP_2^\star$ covers the volume bounded by $\delta\widehat{SR}$. This figure best viewed in color.

To compute the full factorization, Algorithm 10 applies $DisjointSPs$ to $SR \times \mathcal{SR}$, so its correctness can be seen by inspection. Assuming elements of an SR are stored in the same sorted order as the index sets, Line 4 adds linear complexity in $|\mathcal{SR}|$ and the sub-routine in Line 5 for a total complexity of $O(g(\cdot)|SR||\mathcal{SR}|)$. In practice, there are many techniques that can be used to reduce the *expected* running time of the algorithm, for example, hierarchical spatial decompositions may dramatically reduce the number of time $DisjointSPs$ needs to be called.

As mentioned in the previous section, SP disjointness need not require exactness: the property can be guaranteed so long as the SR boundary, δSR, can be conservatively approximated. Lemma 14 shows that a strict subset of SPs associated with a conservative δSR can be guaranteed to result in a *strictly conservative* disjointness computation.

Lemma 14. *For a stopping region SR_1, let $\delta\widehat{SR}_1$ be a conservative approximation to the*

boundary of SR_1. Let $SR'_1 \subset SR$ be a strict stopping path subset. For each $SP \in SR'$ let $SP^\star$ be SP with its boundary expanded such that the volume enclosed by $\delta SR'$ is covered by all $SP^\star$ (as shown in Figure 5.3). Let $SR^\star$ be the stopping region composed of all $SP^\star$. For any other SR_2, if SP disjointness holds between $SR^\star$ and SR_2, then it must hold between SR_1 and SR_2.

Proof. Assume SR_1, $SR_1^\star$, and SR_2 as defined in the lemma. Let $Q = SR_1 \cap SR_2$ and $Q' = SR_1^\star \cap SR_2$. Because $SR^\star$ is constructed conservatively, then it must be that $Q \subseteq Q'$, and, by definition, $SP_1 \subseteq SP_1^\star$. It follows directly that if there exists $SP_1^\star \in SR_1^\star$ such that $SP_1^\star \notin Q'$, then there must exist $SP_1 \in SR_1$ such that $SP_1 \notin Q$. Thus, if SP disjointness is satisfied for Q', it must also be satisfied for Q. □

Thus, discreteness and finiteness assumptions among the SPs are safe to make provided a conservative approximation to δSR can be computed. This is a powerful tool for constructing efficient implementations of SD algorithms.

5.4.3 Preserving SP disjointness

In order for agents to efficiently solve Problem 7, they need to maintain the non-coordination guarantee of Theorem 5. This section presents a generic algorithm for maintaining the non-coordination guarantee by allowing agents to ensure that disjoint SPs are always available. To begin, Problem 8 formulates the precise problem to be solved.

Problem 8. Assume the constraints and conditions of Problem 7. Assume an agent A has a control set $\mathcal{U}$. Given the state of the system at a time t, what is the subset of controls $\mathcal{U}^\star \subseteq \mathcal{U}$ for A that preserves SP disjointness for some time horizon T?

Algorithm 11 For an agent state $A(\mathbf{x})$, reference velocity v_r, and motion model M, compute a union of SRs over a time horizon T.

procedure ComputeSweptSRs($A(\mathbf{x}), v_r, M, T$)
  $\mathcal{SR} \leftarrow \varnothing$
  $\mathcal{S} \leftarrow ComputeReachableStates(A(\mathbf{x}), M, T)$
  for $\mathbf{x}' \in \mathcal{S}$ **do**
    $\mathcal{SR} \leftarrow \mathcal{SR} \cup ComputeSR(A(\mathbf{x}), v_r, M)$
  end for
  return $\mathcal{SR}$
end procedure

Algorithm 12 For a set of obstacles $\mathcal{O}$, a given agent state $A(\mathbf{x})$, a reference velocity v_r, agent motion model M, and obstacle motion models $M_{\mathcal{O}}$, compute the set of control sequences Φ for A that satisfies SP disjointness for all $\mathcal{O}$ over a time horizon T.

procedure SPDisjointControls($\mathcal{O}, A(\mathbf{x}), v_r, M, M_{\mathcal{O}}, T$)
  $\Phi \leftarrow ComputeFeasibleControlSequences(A(\mathbf{x}), M, T)$
  for $O, M_O \in \mathcal{O}, M_{\mathcal{O}}$ **do**
    $\mathcal{SR}_O \leftarrow ComputeSweptSRs(O, v_r, M_O, T)$
    for $\phi \in \Phi$ **do**
      for $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \phi$ **do**
        $A(\mathbf{x}) \leftarrow ApplyControl(A(\mathbf{x}), \mathbf{u}_i)$
        $SR \leftarrow ComputeSR(A(\mathbf{x}), v_r, M)$
        $SR \leftarrow PruneSR(SR, \mathcal{SR}_O)$
        if SR is $\varnothing$ **then**
          $\Phi \leftarrow EraseFrom(\phi, \Phi)$
          break
        end if
      end for
    end for
  end for
  return Φ
end procedure

It is important to note that Problem 8 makes no mention of the decision processes driving each agent, which means a solution must account for *any* decision agents may make within T. To do this it is necessary to rely on the fact that the magnitude of the state change due to decision processes is bounded by the magnitude of the state change possible due to the dynamic model of the system. Therefore, computing disjointness with respect to the union of SRs over all states that may feasibly be occupied within T suffices to ensure that SP disjointness is maintained over T. The solution to Problem 8 has two main parts: first, the set of SRs over a horizon T is computed in Algorithm 11, followed by the control set for which SP disjointness can be maintained in Algorithm 12.

The goal of Algorithm 12 is to compute the set of control sequences that can be executed over T for which SP disjointness can be guaranteed to hold, and the inner for-loop is what accomplishes this. This is a brute force method, so correctness is seen by inspection. Complexity of this algorithm is clearly quite high, but as stated in the introduction, the goal is to present a generic algorithm that enables correct, problem-specific approximations to be formulated. In many cases, for instance, the practical and conservative way to approximate this algorithm is to compute bounding hulls of the swept SRs, and then to compute disjointness with respect to those. Over small values of T the hulls are often simple to construct, and for certain types of hulls, particularly convex hulls, intersection tests are quite efficient.

5.4.4 The Selective Determinism framework

This section presents the SD framework, which is formulated employing the above-defined generic algorithms using the SCIMP framework. In this framework there are two main components: the computation of a global guidance control, and the computation of an local collision avoidance control.

In order to maintain the non-collision guarantees afforded by Theorem 5, the local controller needs to provide some guarantees that it can establish and maintain SP disjointness. Theoretically such a controller needs to implement Algorithm 12 to compute the set of controls that it uses to minimize input error. Such a controller is a type of *trusted* controller [140], which is one that can be relied upon to be correct. Because non-collision guarantees are handled by the trusted controller, the requirements on the guidance control can be significantly relaxed. In this case any number of approximate path planning algorithms can be used (see LaValle [87] for a broad survey of methods).

Now, assuming a given guidance controller, only the control space metric $\mu(\cdot)$ and $S(\mathbf{x}, \mathbf{u})$ (§3.3) need to be defined in order to formulate an algorithm under the SD framework. As it stands, Algorithm 12 provides exactly the properties required for $S(\mathbf{x}, \mathbf{u})$, including the control set computation. Assuming controls are represented as fixed-size vectors, the metric $\mu(\cdot)$ can be computed with any appropriate vector norm. For convenience, define state as the following vector:

$$\mathbf{x} = \begin{bmatrix} \mathcal{O} \\ A(\mathbf{x}) \\ v_r \\ M \\ M_{\mathcal{O}} \\ T \end{bmatrix} \tag{5.1}$$

The local controller for the general SD algorithm is defined Algorithm 13. Line 3 of Algorithm 13 computes the sample set of the desired size (according to Equation 3.19), and Line 8 provides an additional check against problem assumptions: if the disjointness

condition is ever violated, then some belief about the environment or assumption of behavior of other agents has been violated and collision may be imminent, and a collision mitigation strategy, such as discussed in Chapter 3 or §4.4.6, must be employed.

Algorithm 13 A local controller based on the Selective Determinism algorithm. For a given system state $\mathbf{x}$ (from Equation 5.1), goal-directed control $\mathbf{u}^d$, confidence level α, and hyperparameters a and b, compute a control $\mathbf{u}$ that allows the agent to remain collision free to confidence level α while simultaneously minimizing distance from $\mathbf{u}^d$.

procedure ComputeLocalControl($\mathbf{x}, \mathbf{u}^d, \alpha, a, b$)
    $\Phi_\alpha \leftarrow SPDisjointControls(\mathbf{x})$
    $\mathcal{X} \leftarrow ComputeStateSamples(\alpha, a, b, \mathbf{x})$
    for $\mathbf{x}' \in \mathcal{X}$ **do**
        $\Phi_\alpha \leftarrow \Phi_\alpha \cap SPDisjointControls(\mathbf{x}')$
    end for
    if Φ_α **is** $\varnothing$ **then**
        α confidence cannot be met. Compute collision mitigation control $\mathbf{u}^\star$.
    else
        $\mathcal{U} \leftarrow InitialControls(\Phi_\alpha)$
        $\mathbf{u}^\star \leftarrow \arg\min_{\mathbf{u} \in \mathcal{U}} \mu(\mathbf{u}, \mathbf{u}^d)$
    end if
    return $\mathbf{u}^\star$
end procedure

It is important to note that Algorithm 13 defines only the local controller, taking the global guidance controller command as input. The implied decoupling between local and global controllers is actually quite strong. Individual problems will impose their own constraints on how these controllers relate to each other, but in principle they need not run sequentially or with any defined frequency.

5.5 Empirical evaluation

This section introduces the *path network traversal* problem that will be used to evaluate an SD navigation algorithm in simulation. The evaluation is done with a *Completion Time Variance* (CTV) metric, which this section also defines, that can be used to compare solu-

tion quality for a given problem under different parameterizations/initializations. CTV is a measure of the variance in completion time due to variations in the initial conditions of the problem. This is a generalized form of the Excess Time metric described in §3.5.3, and is chosen because it gives an indication of robustness and stability of SD framework algorithms, where completeness is interpreted as directly proportional to robustness, and completion time variance across scenarios is interpreted as inversely proportional to algorithm stability with respect to its inputs.

The CTV measure is defined first, followed by the PNT problem and the formulation of its SD solution.

5.5.1 Completion time variance and failure rate

To define CTV, let Γ be some scenario that is parameterized by a vector θ, and let $f(\Gamma, \theta)$ be a function that computes the time it takes an agent to reach a goal state in scenario Γ under parameter vector θ. Now, let a sequence $\Theta_1, \ldots, \Theta_n$ be i.i.d. random parameter vectors and let $\Omega^* = \{\theta_1, \ldots, \theta_n\}$ be a sample of size n drawn from $\Theta_1, \ldots, \Theta_n$. For $n > 1$, the unbiased estimate of CTV is given in Equation 5.2.

$$CTV(\Omega^*) = \frac{\Sigma_{i=1}^{n} f(\Gamma, \theta_i)^2 - \left(\Sigma_{i=1}^{n} f(\Gamma, \theta_i)\right)^2 / n}{n - 1} \tag{5.2}$$

Samples for which $f(\Gamma, \theta)$ is infinite (or exceeds some given threshold), are removed from Ω^* prior to CTV computation and added to a failure set $\Omega^{\times}$. The proportion of failure samples to total samples $\frac{|\Omega^{\times}|}{|\Omega^*|+|\Omega^{\times}|}$, called *failure rate*, is inversely proportional to a notion of robustness: a lower failure rate indicates greater robustness.

5.5.2 The path network traversal problem

This section introduces the PNT problem and defines its solution under the SD framework. The PNT problem is a relatively simple partially observable multi-agent navigation problem that also exhibits non-trivial interaction complexity. It is defined such that the deterministic motion planning sub-problem can be solved using the PST-space planner developed in Chapter 2. This planner has very desirable properties for SD problems because it can compute feasible control sets very efficiently. The PNT problem is described in Problem 9.

Problem 9. Let $\mathcal{A}$ be a set of 2D unit square agents navigating a planar graph $G = (V, E)$. Assume all agents are initialized at some $v \in V$ and that collision is never inevitable in the initial system state. Assume that agents have bounded speeds and accelerations, and that these bounds are constant and independent of agent state. Assume G is constructed such that an agent at maximum speed can traverse a maximum of $e_{\max}$ edges and can come to a stop while traversing at most $e_{\min}$ edges. Assume each agent can observe the instantaneous dynamic state of the other agents, but that each agent actuates according to a unique decision process. Each agent may assume with certainty that other agents will prefer both to avoid collision and to avoid causing collision, but that otherwise the decision processes of other agents are not fully observable. For an agent A at a start vertex v^s the problem is for A to navigate to a destination vertex v^g without collision and within a given time horizon.

To solve Problem 9, the following sections define problem-specific versions of the generic algorithms given by the SD framework.

Subroutine: *ComputeSR*

To implement *ComputeSR*, the following functions are defined:

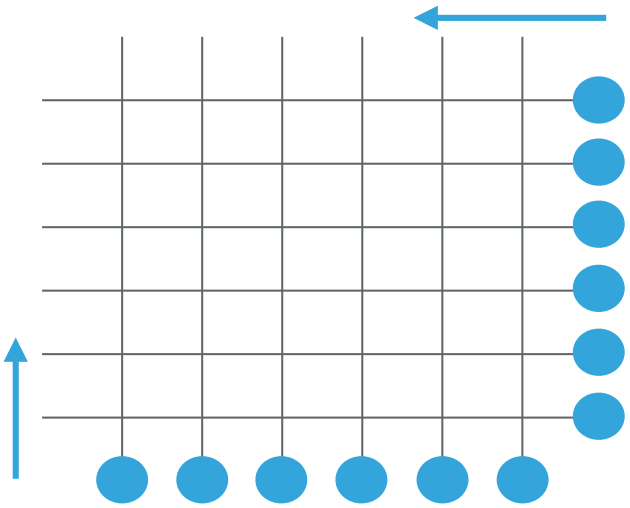


Figure 5.4: The example PNT problem. The agents at the bottom attempt to move to the top while the agents at the right attempt to move to the left.

ComputeFeasiblePathsWithIndexSet In principle there exist an infinite number of feasible paths in G; however, in this case, only stopping paths are of interest, which are the paths that A needs to come to a stop in minimum time. Given the dynamic constraints of Problem 9, the length of these paths L is fully determined by the A's state $\mathbf{x}$. Thus, this function computes and returns all paths of length L that begin at $\mathbf{x}$. The general problem of computing all paths in a graph is intractable, but the definition of G bounds the complexity to a small constant because stopping paths are bounded in length to $e_{\min}$ edges.

ComputeMinimumSweptPathWithControlSet This function is trivial to implement. The control sequences ϕ_i are always $\ddot{s}_{\min}$, and the swept path is fully described by the agent model and the path position s at the desired offset in ϕ_i.

As noted, the complexity of *ComputeFeasiblePathsWithIndexSet* is bounded by a small constant, so its complexity is $O(C)$. The other routine can be computed in constant time and is called $O(C)$ times. Thus, the total complexity is $O(C)$.

Subroutine: *ComputeSweptSRs*

Defining the *ComputeSweptSRs* subroutine poses a daunting problem: the computation of a union of nested swept volumes. For Problem 9, however, the complexity of this problem can be dramatically reduced through a simple observation. Note that the set of paths that A can traverse over T is exactly contained by the set of paths traversable by controlling with $\ddot{s}_{\max}$ until $\dot{s}_{\max}$ is reached and maintaining it. Note further that this implies that the spatial projection of the set of SRs is exactly contained by the spatial projection of the union of swept volumes along these paths with the SRs computed at their termini. One can conservatively treat the spatial projections of these regions as if they are occupied for all T. This significantly simplifies computation, maintains correctness, and, for small T, has virtually no effect on performance.

Thus, for Problem 9 the *ComputeSweptSRs* subroutine is given by Algorithm 14.

Algorithm 14 For an agent state $A(\mathbf{x})$, reference speed $\dot{s}_r$, and motion model M, compute the maximum reachable paths over a time horizon T as well as the SRs from the terminal path states.

```
1: procedure ComputeSweptSRs(A(x), ṡ_r, M, T)
2:     SR ← ∅
3:     X, I, P ← ComputeReachablePathsWithIndexSets(A(x), M, T)
4:     for i ∈ I do
5:         SR ← SR ∪ ComputeSR(A(x_i), ṡ_r, M)
6:     end for
7:     return SR, P
8: end procedure
```

The *ComputeReachablePathsWithIndexSets* function computes the maximum number of paths reachable from a given start state, and its complexity is bounded by $O(C_1)$. As a byproduct the set of terminal states $\mathbf{x} \in \mathcal{X}$ of those paths is also computed. The union on Line 5 is of trivial complexity, but the loop itself is linear in *ComputeSR*, which is $O(C_2)$. The overall complexity, then, is $O(C_1 + C_1C_2)$.

Subroutine: *SPDisjointControls*

As with *ComputeSweptSRs*, the nature of Problem 9 and the properties of the PST-space planner allow *SPDisjointControls* to be greatly simplified.

First, assume that all SPs for A are constant length L that is bounded by the maximum SP length over T given $A(\mathbf{x})$. From this assumption it follows directly that any control sequence for an agent A that maintains at least L distance from the obstacles' swept SRs satisfies the SP disjointness property.

Algorithm 15 For a set of obstacles (with states) $\mathcal{O}$, a given agent state $A(\mathbf{x})$, a reference speed $\dot{s}_r$, agent motion model M, and obstacle motion models $M_{\mathcal{O}}$, compute the set of control sequences Φ for A that satisfies SP disjointness for all $\mathcal{O}$ over a time horizon T.

```
1: procedure SPDisjointControls(O, A(x), ṡ_r, M, M_O, T)
2:     SR_O, P_O ← ∅
3:     for O, M_O ∈ O, M_O do
4:         SR, P ← ComputeSweptSRs(O(x_O), ṡ_r, M_O, T)
5:         SR_O, P_O ← SR_O ∪ SR, P_O ∪ P
6:     end for
7:     L ← ComputeMaxSPLength(A(x), M, T)
8:     O_PT ← ComputeBoundedPaddedPTObstacles(SR_O, P_O, L)
9:     G_V ← VisibilityGraphPST(O, x)
10:    U ← ComputeFeasibleInitialControls(G_V)
11:    return U
12: end procedure
```

Algorithm 15 implements these ideas. There are several new sub-routines introduced in Algorithm 15. The routine on Line 7 is a trivial law of motion computation. The routine on Line 9 implements the PST-space planner defined in Chapter 2, and the routine on Line 10 implements the computation of feasible initial control sets described in detail in §3.5.2. The PT-space obstacle computation on Line 8 implements the bounded PT-space obstacle computation of Algorithm 6 (§2.4.2) by computing the minimum and maximum extents along each $P \in \mathcal{P}$ each obstacle O can attain according to its motion model M_O

rather than uncertainty (as was shown in Figure 2.9e). In addition, the PT-space obstacle corresponding to the SRs for each O are computed and added to the set.

From the previous section, Line 4 contributes $O(C1 + C_1C_2)$ complexity and, assuming sort order is maintained, Line 5 contributes $O(C_3)$ (as noted in §5.5.2, the number of reachable paths is bounded by a small constant). Under the for-loop, and consolidating constants, these lines contribute $O(|\mathcal{O}|C)$ total complexity. Line 7 is trivially computable in constant time, Line 8 is $O(|\mathcal{O}|(|O|^2k + |O|\log|O|))$ (where k is a time discretization; see §2.4.2), Line 9 is $O((|\mathcal{O}||O|)^5)$ as is Line 10 (§2.4). All of these terms together lead to a complexity that is rather long but notably polynomially bounded. Interestingly, in practice, it is not the $O((|\mathcal{O}||O|)^5)$ visibility graph construction that dominates computation, rather it is the $O(|\mathcal{O}|(|O|^2k + |O|\log|O|))$ obstacle construction. Reasons for this are discussed in Chapter 2.

Subroutine: *ComputeLocalControl*

With the previous sub-routines defined, the *ComputeLocalControl* method can be implemented directly as specified in Algorithm 13. For the PNT problem, the routines $DisjointSPs$ and $PruneSR$ do not need to be explicitly implemented as their functionality is implicit in the operation of $SPDisjointControls$.

As would be expected, *ComputeLocalControl* is similar to the algorithm specified in §3.5.2 for computing SCIMP controls for intersection crossing. The primary difference is that *ComputeLocalControl* uses PT-space obstacles computed to prevent agent/obstacle SR intersections rather than agent/obstacle model intersections.

CTV for the PNT problem

Finally, in order to evaluate solutions to the PNT problem, the CTV metric needs to be defined for it. Assume as input to the PNT problem a graph G, initial and goal states for the ego agent, and an obstacle agent count $|\mathcal{A}|$. The remaining inputs are the initial states of the obstacle agents, which are given by the parameter vector θ. Assume orientations are included in agent states. To compute CTV, the sample set Ω^* needs to be defined.

Assume each Θ_i is normally distributed[3] with $P(\Theta) \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, and assume as additional input for the CTV computation a mean parameter vector μ_θ, covariance Σ_θ, and sample count n. The sample set Ω^* can then be generated with any number of algorithms for computing samples from multivariate normals (for reference see [31, 111]), and the CTV computed according to Equation 13, with $f(\Gamma, \theta)$ being computed using the PNT-specific formulation of Algorithm 13 described in this section.

5.5.3 Results & analysis

This section provides results of computing CTV for a PNT problem defined on a graph G, which is a 6×6 grid that is $20m$ to a side. There are 12 agents, including an ego agent, that attempt to traverse from one side to the other. All agents are unit squares with reference points at their centers. Speed bounds are $[0, 10]m/s$ and acceleration bounds are $[-4, 4]m/s$. All information about G and dynamic constraints is stored in Γ. Mean parameter vector μ_θ is initialized with obstacle positions along the outside edges of G oriented inwards toward the opposite side with initial speeds and accelerations are zero. During sampling of μ_θ obstacles poses are independent of other variables and each have $\sigma^2_{\text{position}}$ and $\sigma^2_{\text{orientation}}$ in Σ_θ. Obstacle global controllers are random walks over SP disjoint control sequences with

[3] Any distribution can be used, but for these kinds of tests, a normal is a popular choice.

Table 5.1: CTV for the PNT grid problem with time limit $10s$ and step $0.05s$

$(\sigma^2_{\text{position}}, \sigma^2_{\text{orientation}})$	$(0.2^2, \pi/8)$	$(0.4^2, \pi/4)$	$(0.6^2, \pi)$	$(0.8^2, 3\pi/4)$	$(1.0^2, 2\pi)$
CTV	1.1303	1.24601	1.07412	0.749968	0.755286
Failure rate	0	0	0	0	0

positive accelerations. Initial obstacle speeds and accelerations are also independent of other variables and each have $\sigma^2_{\text{dyn}} = 0$ in Σ_θ. Defining Σ_θ in this way forces all θ_i to initialize the obstacle agents at rest, which guarantees that the system does not initialize in an ICS. The sample count n chosen for all trials is 100. Results for various values of $\sigma^2_{\text{position}}$ and $\sigma^2_{\text{orientation}}$ are summarized in Table 5.1.

As can be seen in Table 5.1, the CTV trends down toward $\sim$0.75 and then levels off. This can be attributed to the fact that μ_θ positions the obstacles in a regular array along the outer edges of G; however, as σ_{position} increases the obstacles become more evenly distributed along G. This evens out the flow of traffic along the edges and has the effect of reducing the expected completion time. The behavior of the CTV measure indicates that the duration of time required by the algorithm to complete a scenario is directly affected by traffic density, and that the algorithm is capable of negotiating dense traffic at the expense of completion time.

Of particular note is the lack of failures. It proved difficult to parameterize the problem such that interesting failures were produced. In this case, interesting failures would be, e.g., configurations of agents resulting in a deadlock appearing sporadically during testing. Failures could be forced by reducing the time limit severely, or organizing the agents in a pathological way, but this had a mostly binary effect: either all tests passed or all failed. This can be taken as an indication that, for this problem, the algorithm is relatively robust. For comparison, giving each agent only the PST-space planner without the surrounding SD

framework almost always resulted in collision regardless of initial parameterization. This is because the assumption of determinism used by that algorithm is immediately violated in this problem, leading agents very quickly into states for which the planner(s) cannot find a solution. Once that happens, the planners fail and the agents tend to drift into collision.

Of course it must be said that because this test scenario is hand-crafted, there will be inherent bias in all results. It is difficult to ascertain exactly what the bias is and how it affects the results, but Bertuccelli [25] covers the general topic of dealing with biased models in decision processes.

5.6 Conclusion

This chapter presented the Selective Determinism framework that enables efficient and robust navigation in stochastic multi-agent domains. The framework builds on results presented in previous chapters to allow application to a wide variety of problems and scenarios while still being capable of maintaining guarantees on hard constraints and solution confidence. A sample problem was presented and problem-specific routines were derived under the framework to demonstrate its use. Exploiting the properties of an efficient speed profile planning algorithm derived in Chapter 2, the solution to the sample algorithm was demonstrated to be computationally efficient while still tackling a non-trivial problem.

Chapter 6

Conclusion

This dissertation addresses a general form of the multi-agent navigation problem in which non-adversarial agents must navigate in the presence of each other, without collision, while progressing toward some desired goal (Problem 1). The approach taken breaks with the types of approaches usually used in that it does not attempt to optimally solve under a global, joint value function. As covered extensively in Chapters 2, 3, & 4, the nature of such a value function results in intractable problem complexity. Instead, the work presented here sets up and solves a much simpler online collision avoidance problem, and it puts in place around it a framework that leverages any available flexibility in the collision free solution space to choose goal-biased, collision-avoidant controls.

The main contribution of this dissertation is the formulation of a principled theoretical framework that allows interaction effects in complex multi-agent navigation problems to be factored out via decomposition into separate collision avoidance and goal direction problems. This framework is based on two main parts. For the first part, Chapter 3 starts at a low level and leverages stochastic optimal control theory to formulate a constrained interference minimization principle as a general framework within which multi-objective control problems can be formulated and behavior ensured to a defined level of confidence. This principle considers the problem of performing a minimum distance projection of a desired control

input onto the manifold of permissible controls. While solving the general problem exactly is shown to be intractable, a probabilistic, rollout-based approach is presented that can compute solutions that are correct to a desired confidence level, and it is shown that for CE problems, the principle naturally produces the correct, exact result.

For the second part, Chapter 4 approaches from a high level and shows that complex, partially observable multi-agent decision process problems can be factored into independent goal direction and collision avoidance planning problems under certain assumptions. The factorization can then bring the problems from intractable complexity classes into tractable ones. The key insight is that system dynamics alone can determine whether a system requires agent coordination in order to remain collision free. This property allows agents to *independently* check for the requirement and adjust their own actions to prevent the requirement from appearing in the system. In addition, it allows them to compute the control set that maintains this non-coordination guarantee. Freed from the requirement to coordinate actions in order to maintain non-collision guarantees, agents can assume Assertion 1 holds and perform planning as if the interaction effects of their actions have been factored out.

With those two results in place, Chapter 5 brings them together by recognizing that the factored sub-problems resulting from Assertion 1 can be directly formulated as a multi-objective control problem under the constrained interference minimization principle. The result is a new Selective Determinism (SD) framework that allows deterministic planning algorithms to be used to efficiently and robustly solve partially observable multi-agent navigation problems while maintaining confidence guarantees on collision avoidance. The chapter presents the framework and the nominal components necessary to implement navigation algorithms under it. In addition, a problem-specific implementation is derived and tested using a novel PST-space planner, the derivation of which is given in Chapter 2.

The PST-space planner computes solutions to the speed profile planning problem for traversing fixed paths in the presence of transversely moving obstacles. Under the assumption of deterministic obstacles and agent independence, the planner computes a visibility graph in a Path-Speed-Time space that accounts for both bounded speed and acceleration as it searches for collision free trajectories. Exploiting convexity in the sets of reachable speeds, the planner is able to compute a full visibility graph in low-order polynomial-time.

The PST-space planner is included in this work because it is an ideal candidate for use in an SD algorithm. As noted in Chapters 2 & 5, the planner on its own is not suitable for use in multi-agent navigation problems: the requirement for determinism and action independence are very strong, and even small violations can cause an agent computing controls with the planner to come into collision. As part of an SD solution, however, control computation using the planner becomes robust to non-determinism and reactive obstacles while sacrificing very little in terms of computational efficiency. The result is a robust, real-time capable multi-agent navigation algorithm.

Overall, the SD framework serves as a foundation for future work examining how best to address navigation problems. Of specific interest for future work are different interpretations of how components of the SD framework can be implemented. For instance, in this dissertation the SP disjointness computation is presented in terms of physical geometric quantities because this is how it can most directly be measured. But in some systems an explicit geometry may not be necessary to compute SP disjointness. Consider an agent whose only visibility or model of the world is through a time-to-contact (TTC) measure. For such an agent, maintaining some minimum time headway may suffice in order to guarantee SP disjointness. Assuming an agent is somehow capable of measuring TTC, this kind of model is attractive because of its simplicity and because it allows an agent to forgo the extremely diffi-

cult and computationally expensive task of computing and maintaining geometric models of the world. Because TTC can also be measured directly in sensor space (assuming a camera as a sensor) it allows much closer integration of the perception and planning systems for the agent. Early works, such as Nelson [112], Camus et al. [33], demonstrated that the approach holds promise, but was impractical at the time due to the quality of sensor hardware and perception algorithms. Advances in both of those domains, however, should spur renewed interest in this approach, as in Forootaninia et al. [46].

Chapter 4 left open two interesting conjectures. The first, Conjecture 1, posits that stopping paths are the unique coordination-free contingency plans available to agents in a non-adversarial and self-preserving multi-agent system. While intuitively appealing, and probably safe to make in practice, it does not seem straightforward to show this rigorously, or to determine what properties of a system are necessary to do so. It is an important question to answer, though, because it would yield insights into whether and how the results of Chapter 4 could be applied to systems with drift, or to other types of systems where it is difficult, or impossible, for agents to actually come to a stop. The second, Conjecture 2, posits that the definition of coordination used in this work (Definition 10) is not only a semantic distinction between what does and does not constitute coordination, but also a computational theoretic one. The proposition is that the problem of finding a unique set of contingency plans in a non-SP disjoint system cannot be solved efficiently, which is an inherent property of coordination. Resolving this conjecture would further solidify the foundations of the results in Chapter 4.

Ultimately, this work is designed to advance robotics one more step toward a full solution to the general multi-agent navigation problem described in Problem 1. But it also recognizes that these results address only a small part of the whole problem. Solely from a AI

standpoint, an arguably larger problem than that of planning and control is that of perception, i.e., the segmentation and labeling of the environment in semantically meaningful ways. Outside the realm of AI there is also the reality that, to be useful, these algorithms need to run on physical systems. This brings into play myriad mechanical and electrical issues, all of which (thankfully) fall far outside the scope this dissertation. The hope, however, is that the work presented in these chapters inspires new work, new insights, and new interest in the arena of multi-agent navigation.

Glossary

Agent (n.) Symbol: A. An entity that interacts in some non-trivial way with its environment. Also referred to as *robot*. 1–8, 11–14, 34, 39, 40, 46, 47, 57, 62–64, 77–102, 104–111, 113–116, 118, 119, 121, 122, 124–139

Agent state (n.) Symbol: $A(\mathbf{x})$. A state representation containing both the dynamic state of an agent A and the volume of workspace it occupies at state $\mathbf{x}$. 14, 84, 86, 87, 92, 99, 100, 107, 117, 122, 124, 129, 130

Bang singular trajectory (n.) Without loss of generality, let P refer to either a P^+ or P^- curve, and let t_2 be the time coordinate of p_2. Define P_2 to be a terminal curve that passes through p_2. Define P_1 to be an initial curve with derivative $\dot{s}_1$ at origin point p_1. Define an L curve that is tangent to the initial and terminal P curves at switching times t_{s1} and t_{s2} with derivative $\dot{s}_s$. Define a *bang-singular trajectory* as:

$$\mathbf{x}_1(t) = \begin{cases} P_1(t, \dot{s}_1) & : 0 \leqslant t \leqslant t_{s1} \\ L(t, \dot{s}_s) & : t_{s1} < t < t_{s2} \\ P_2(t, \dot{s}_s) & : t_{s2} \leqslant t \leqslant t_2 \end{cases}$$

17, 18, 20, 21

Belief distribution (n.) Symbol: $\mathbf{z}$. "A belief distribution assigns a probability (or density value) to each possible hypothesis with regards to the true state. Belief distributions

are posterior probabilities over state variables conditioned on the available data." [143]. plural 50–60, 64, 66

Certainty equivalence (n.) Abbreviation: CE. When the expected utility of a control policy that ignores uncertainty is equal to one that does not, the system is said to exhibit *certainty equivalence*. 53–56, 60, 136

Complete (adj.) An algorithm is said to be complete when it computes a satisfying solution if and only if such a solution exists. 46

Completeness (n.) The property of being complete. 12, 70

Configuration (n.) Symbol: c. "The configuration of a robot is a set of independent parameters of minimal cardinality which uniquely defines the position and orientation of every point of the robot" [48]. 13, 64

Contingency plan (n.) A *contingency plan* is a control sequence that an agent can execute that is guaranteed to avoid ICS space. 85, 87, 89, 90, 93, 94, 101, 119, 138

Control (n.) Symbol: $\mathbf{u}$. A command that an agent can execute to change its state or configuration. 2, 5–7, 11, 50–71, 73, 83–85, 88, 90, 108, 110, 111, 113–115, 117, 122, 124, 125, 130, 132, 135, 137

Control policy (n.) Symbol: π. A mapping of states to controls that is optimal with respect to some cost function. 5, 51, 57, 68, 69, 100

Control set (n.) Symbol: $\mathcal{U}$. The set of all controls available to an agent. 60, 63, 65, 70, 71, 114, 122, 123, 125, 127, 130, 136

Coordination (n.) The actions of two agents are said to require *coordination* when the feasibility the control sequence either agent uses is not independent of the other's. 2, 8, 77, 79, 80, 82–84, 87–92, 94, 95, 98–101, 104, 113, 119, 121, 136, 138

Ego (adj.) Of or pertaining to the agent under control of a single-agent planning algorithm. 47, 65–67, 73, 113, 132

Exact (adj.) An algorithm is said to be exact if it solves a problem with zero approximation error. 38

Feasibility (n.) The condition of being feasible. 84, 90

Feasible (adj.) A path, trajectory, or velocity profile is said to be feasible if it does not violate constraints. 13–16, 18, 20, 24, 27, 28, 35, 36, 46, 60, 70, 71, 85, 106, 107, 115–117, 127, 128, 130

Feasibly (adv.) In a way that is feasible. 123

Guided collision avoidance (n.) Guided collision avoidance describes the strategy of choosing goal-directed motions from the space of collision avoiding controls in order to navigate to a goal while satisfying collision constraints. 4, 6, 109, 111, 113

Homotopic (adj.) Two trajectories are homotopic if there exists a homotopy between them. 15

Homotopy (n.) A continuous deformation between two functions 9, 23, 29, 31, 43

Horizon (n.) Symbol: T. The bound on the period of time over which a plan is, or is intended to be, defined. 13, 14, 38, 39, 42, 51–56, 59, 64, 79, 85, 122–124, 127, 129, 130

Inevitable collision state (n.) An *inevitable collision state* (ICS) for an agent A is a state from which all feasible future trajectories of A result in collision:

$$\mathbf{x} \text{ is ICS} \leftrightarrow \forall \phi, \exists B_i, \exists t :: A(\phi(\mathbf{x}, t)) \cap B_i \neq \varnothing$$

ICS notations and definitions adapted from Fraichard and Asama [49]. 46, 82, 85, 88, 89, 91, 105, 106, 133

Interacting agent (n.) An *interacting agent* is one whose dynamic state is a function of the dynamic state of the system and an internal policy (for example, pedestrians or animals could be interacting agents). 84, 85, 105

L-curve (n.) For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, an L curve is a line that defines an arc-length path offset for an input time t assuming zero acceleration:

$$L(t, \dot{s}) = P(t - t_o, \dot{s}, 0)$$

17–22

Linear, time-invariant (adj.) Abbreviation: LTI. A descriptor used for systems whose response is linear in the input and invariant with respect to time, that is, the response of the system is a linear function of the input, and it does not depend on when the input is given. 53, 54

Longitudinal (adj.) Along the current direction of travel. For example, longitudinal velocity is the magnitude of an agent's velocity vector. 11

Motion model (n.) A model (typically a system of differential equations) that describes how an agent can undergo spatial state changes. 101, 102, 116, 117, 122, 124, 129, 130

Non-interacting agent (n.) A *non-interacting agent* is one whose dynamic state is a function only of the dynamic state of the system (for example, trees or rolling rocks could be non-interacting agents). 84, 85

Non-signalized (adj.) Indicates that a particular road traffic scenario, e.g. an intersection, has no signs or lights to direct traffic. 10

Obstacle (n.) For an agent A navigating an environment occupied by a set of interacting agents and non-interacting agents $\mathcal{A}$, an *obstacle* O is a member of the set of obstacles $\mathcal{O}$, which is defined as:

$$\mathcal{O} = \mathcal{A} \setminus A$$

1, 7, 9–14, 23–30, 34–36, 38–46, 64–73, 76, 81, 82, 85, 89, 92, 94, 102, 105–108, 122, 124, 130–133, 137

P^+-curve (n.) For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, a P^+ curve is a PT parabola that defines the path offset achievable at time t under maximum acceleration:

$$P^+(t, \dot{s}) = s_o + P(t - t_o, \dot{s}, \ddot{s}_{\max})$$

17–20, 22

P^--curve (n.) For a given PST-space origin point $(s_o, \dot{s}_o, t_o)$, a P^- curve is a PT parabola

that defines the path offset achievable at time t under minimum acceleration:

$$P^{-}(t, \dot{s}) = s_o + P(t - t_o, \dot{s}, \ddot{s}_{\min})$$

17–20, 22

Path (n.) Symbol: P. A sequence of configurations. Used in this work to describe curves in $\mathbb{C}$. 9–14, 16–18, 21, 28, 34, 38–41, 44, 64, 79, 81, 86, 87, 89–92, 96, 97, 99, 100, 116, 117, 127–131, 137

P-curve (n.) A P curve is an origin-centered PT-space parabola that defines an arc-length path offset after a given time span Δt for given $\dot{s}_o$ and $\ddot{s}_o$:

$$P(\Delta t, \dot{s}_o, \ddot{s}_o) = \dot{s}\Delta t + \frac{1}{2}\ddot{s}\Delta t^2$$

16–22

Planning (n.) The process of determining a sequence of configurations or states that brings an agent from a start state to a goal state. 11–13, 78–83, 93, 103, 105, 136

Polynomial-time (adj.) Describes the class of decision problems solvable in polynomial time by a Turing machine [1]. 7, 9, 29, 65, 90, 137

PS-space (n.) A two dimensional space composed of a path position dimension and a speed dimension. 13, 18, 21

PST reachable region (n.) Given a PST-space start point $p^s = (s^s, \dot{s}^s, t^s)$, define the PST-space reachable region of PS-space space from p^s at any point in time $t \geqslant t^s$ as

dimensions for time and potentially one or more derivatives of a configuration dimension 12, 82–84, 93, 102, 116

State space obstacle (n.) A *state space obstacle* (B) is the volume swept out by an obstacle O over a time T as it evolves from an initial state $\mathbf{x}_i$ under a control trajectory ϕ_i:

$$B = \bigcup_{t \in T} O(\phi_i(\mathbf{x}_i, t))$$

85, 106–108

Stopping path (n.) For a state $\mathbf{x}$ and path P, the *stopping path* $SP(A(\mathbf{x}), P)$ is the minimal set of agent states A must occupy while coming to zero velocity from $\mathbf{x}$ along P (see Figure 4.2a). 86–92, 98, 100, 109, 115–121, 128, 130, 132, 138

Stopping region (n.) For a given agent state $A(\mathbf{x})$ let $\mathcal{P}$ be the set of all followable paths and let I be its index set. Define the *stopping region* $SR(A(\mathbf{x}), \mathcal{P})$ as the disjoint union of all SPs over $\mathcal{P}$ (see Figure 4.2b):

$$SR(A(\mathbf{x}), \mathcal{P}) = \bigsqcup_{i \in I} SP(A(\mathbf{x}), P)$$

86, 87, 89, 91–93, 95, 97–103, 115–123, 129–131

Trajectory (n.) Symbol: $\mathbf{x}(t)$. A mapping of time to state. Used in this work to describe curves in $\mathbb{S}$. 7, 9, 11, 13–16, 18–30, 35–39, 42, 44, 46, 51, 53, 70, 81, 84–86, 88, 137

Visibility graph (n.) A visibility graph is a graph in which two nodes are connected if and only if there is an obstructed path between them. 12, 14–16, 29, 38, 40, 43, 70,

Workspace (n.) Symbol: $\mathbb{W}$. The space in which the agent is operating. Typically some space in the physical world (or some representation thereof in the case of simulation).

Bibliography

[1] Complexity Zoo. URL https://complexityzoo.uwaterloo.ca/Complexity_Zoo. Accessed: 2016-11-21.

[2] Roundabout: An informational guide, 2000. in U.S. Department of Transportation Federal Highway Administration Publication Number: FHWA-RD-00-067.

[3] *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA*, 2007. IEEE. ISBN 978-1-4244-0912-9. URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4398943.

[4] *Robotics: Science and Systems VIII, University of Sydney, Sydney, NSW, Australia, July 9-13, 2012*, 2012. URL http://www.roboticsproceedings.org/rss08/.

[5] *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013. IEEE. ISBN 978-1-4673-5641-1. URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6615630.

[6] Steven Abrams and Peter K. Allen. Computing swept volumes. *Journal of Visualization and Computer Animation*, 11(2):69–82, 2000. doi: 10.1002/1099-1778(200005)11:2⟨69::AID-VIS219⟩3.0.CO;2-7. URL http://dx.doi.org/10.1002/1099-1778(200005)11:2<69::AID-VIS219>3.0.CO;2-7.

[7] Ross E. Allen, Ashley A. Clark, Joseph A. Starek, and Marco Pavone. A machine learning approach for real-time reachability analysis. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 2202–2208. IEEE, 2014. doi: 10.1109/IROS.2014.6942859. URL `http://dx.doi.org/10.1109/IROS.2014.6942859`.

[8] Håkan Alm, Ove Svidén, and Yvonne Waern. *Cognitive ITS: on cognitive integration of ITS functions around the driver's task*, pages 231–237. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1997. ISBN 0-8058-1956-8.

[9] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul A. Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In Alcherio Martinoli, Francesco Mondada, Nikolaus Correll, Grégory Mermoud, Magnus Egerstedt, M. Ani Hsieh, Lynne E. Parker, and Kasper Støy, editors, *Distributed Autonomous Robotic Systems - The 10th International Symposium, DARS 2010, Lausanne, Switzerland, November 1-3, 2010*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 203–216. Springer, 2010. ISBN 978-3-642-32722-3. doi: 10.1007/978-3-642-32723-0_15. URL `http://dx.doi.org/10.1007/978-3-642-32723-0_15`.

[10] Ron Alterovitz, Thierry Siméon, and Kenneth Y. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In Wolfram Burgard, Oliver Brock, and Cyrill Stachniss, editors, *Robotics: Science and Systems III, June 27-30, 2007, Georgia Institute of Technology, Atlanta, Georgia, USA*. The MIT Press, 2007. ISBN 978-0-262-52484-1. URL `http://www.roboticsproceedings.org/rss03/p30.html`.

[11] M. Althoff, O. Stursberg, and M. Buss. Model-based probabilistic collision detection in autonomous driving. In *Intelligent Transportation Systems*, 2009.

[12] S. Anderson, S. Peters, K. Iagnemma, and T. Pilutti. A unified approach to semi-autonomous control of passenger vehicles in hazard avoidance scenarios. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2009.

[13] K.J. Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174 – 205, 1965. ISSN 0022-247X. doi: 10.1016/0022-247X(65)90154-X. URL http://www.sciencedirect.com/science/article/pii/0022247X6590154X.

[14] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles F. Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster. Odin: Team VictorTango's entry in the DARPA urban challenge. *J. Field Robotics*, 25(8): 467–492, 2008. doi: 10.1002/rob.20248. URL http://dx.doi.org/10.1002/rob.20248.

[15] Nakhoon Baek, Sung Yong Shin, and Kyung-Yong Chwa. On computing translational swept volumes. *Int. J. Comput. Geometry Appl.*, 9(3):293–317, 1999. doi: 10.1142/S0218195999000200. URL http://dx.doi.org/10.1142/S0218195999000200.

[16] J.S. Bay. *Fundamentals of Linear State Space Systems*. Electrical Engineering Series. WCB/McGraw-Hill, 1999. ISBN 9780256246391. URL https://books.google.com/books?id=vjoZAQAAIAAJ.

[17] M. Beard, B.-T. Vo, B.-N. Vo, and S. Arulampalam. Void Probabilities and Cauchy-Schwarz Divergence for Generalized Labeled Multi-Bernoulli Models. *ArXiv e-prints*, October 2015.

[18] K. E. Bekris. Avoiding inevitable collision states: Safety and computational efficiency in replanning with sampling-based algorithms. *International Conference on Robotics and Automation (ICRA-10)*, May 2010 2010. URL `http://www.cs.rutgers.edu/~kb572/pubs/ics_tradeoffs.pdf`.

[19] Kostas E. Bekris, Devin K. Grady, Mark Moll, and Lydia E. Kavraki. Safe distributed motion coordination for second-order systems with different planning cycles. *The International Journal of Robotics Research*, 31(2):129–150, 2012. doi: 10.1177/0278364911430420. URL `http://dx.doi.org/10.1177/0278364911430420`.

[20] Avner Ben-Yaacov, Masha Maltz, and David Shinar. Effects of an in-vehicle collision avoidance warning system on short- and long-term driving performance. *Human Factors The Journal of the Human Factors and Ergonomics Society*, 44(2):335–342, 2002.

[21] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002. doi: 10.1287/moor.27.4.819.297. URL `http://dx.doi.org/10.1287/moor.27.4.819.297`.

[22] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000. ISBN 1886529094.

[23] Dimitri P. Bertsekas. Dynamic programming and suboptimal control: A survey from

ADP to MPC. *Eur. J. Control*, 11(4-5):310–334, 2005. doi: 10.3166/ejc.11.310-334. URL http://dx.doi.org/10.3166/ejc.11.310-334.

[24] Dimitri P. Bertsekas. *Rollout Algorithms for Discrete Optimization: A Survey*, pages 2989–3013. Springer New York, New York, NY, 2013. ISBN 978-1-4419-7997-1. doi: 10.1007/978-1-4419-7997-1_8. URL http://dx.doi.org/10.1007/978-1-4419-7997-1_8.

[25] Luca Francesco Bertuccelli. *Robust Decision-Making with Model Uncertainty in Aerospace Systems*. PhD thesis, Massachusetts Institute of Technology, 2004. URL https://dspace.mit.edu/handle/1721.1/46554.

[26] Th. Besselmann and M. Morari. Hybrid Parameter-Varying MPC for Autonomous Vehicle Steering. *European Journal of Control*, 14(5):418 – 431, 2008.

[27] L. Biegler, S. Campbell, and V. Mehrmann. *Control and Optimization with Differential-Algebraic Constraints*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2012. doi: 10.1137/9781611972252. URL http://epubs.siam.org/doi/abs/10.1137/9781611972252.

[28] Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1-3):80–88, 1982. doi: 10.1016/S0019-9958(82)90439-9. URL http://dx.doi.org/10.1016/S0019-9958(82)90439-9.

[29] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In Yoav Shoham, editor, *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge, De Zeeuwse Stromen, The Netherlands, March 17-20 1996*, pages 195–210. Morgan Kaufmann, 1996. ISBN 1-55860-417-0.

[30] Craig Boutilier, Thomas L. Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Intell. Res. (JAIR)*, 11:1–94, 1999. doi: 10.1613/jair.575. URL http://dx.doi.org/10.1613/jair.575.

[31] G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29(2):610–611, 06 1958. doi: 10.1214/aoms/1177706645. URL http://dx.doi.org/10.1214/aoms/1177706645.

[32] Arthur Earl Bryson and Yu-Chi Ho. Applied optimal control: optimization, estimation, and control, 1969. URL http://opac.inria.fr/record=b1081822.

[33] Ted Camus, David Coombs, Martin Herman, and Tsai-Hong Hong. Real-time single-workstation obstacle avoidance using only wide-field flow divergence. In *13th International Conference on Pattern Recognition, ICPR 1996, Vienna, Austria, 25-19 August, 1996*, pages 323–330. IEEE Computer Society, 1996. ISBN 0-8186-7282-X. doi: 10.1109/ICPR.1996.546964. URL http://dx.doi.org/10.1109/ICPR.1996.546964.

[34] John Canny, Ashutosh Rege, and John Reif. An exact algorithm for kinodynamic planning in the plane. In *Proc. 6th Annual Symposium on Computational geometry*, SCG '90, pages 271–280, 1990. ISBN 0-89791-362-0. URL http://doi.acm.org/10.1145/98524.98584.

[35] Bernard Chazelle and David P. Dobkin. Detection is easier than computation (extended abstract). In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 146–

153. ACM, 1980. doi: 10.1145/800141.804662. URL http://doi.acm.org/10.1145/800141.804662.

[36] Christophe Coué, Cédric Pradalier, Christian Laugier, Thierry Fraichard, and Pierre Bessière. Bayesian occupancy filtering for multitarget tracking: An automotive application. *The International Journal of Robotics Research*, 25(1):19–30, 2006. doi: 10.1177/0278364906061158. URL http://dx.doi.org/10.1177/0278364906061158.

[37] Konstantinos Daskalakis and Christos H. Papadimitriou. The complexity of games on highly regular graphs. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, volume 3669 of *Lecture Notes in Computer Science*, pages 71–82. Springer, 2005. ISBN 3-540-29118-0. doi: 10.1007/11561071_9. URL http://dx.doi.org/10.1007/11561071_9.

[38] Vivien Delsart, Thierry Fraichard, and Luis Martinez. Real-time trajectory generation for car-like vehicles navigating dynamic environments. In *Proc. Int. Conf on Robotics and Automation*, pages 2257–2262, 2009. ISBN 978-1-4244-2788-8. URL http://dl.acm.org/citation.cfm?id=1703775.1703816.

[39] David Eberly. Intersection of Convex Objects: The Method of Separating Axes. https://www.geometrictools.com/Documentation/MethodOfSeparatingAxes.pdf, 2008. [Online; accessed 02-January-2017].

[40] Lawrence H. Erickson and Steven M. LaValle. A simple, but NP-hard, motion planning problem. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013,*

Bellevue, Washington, USA. AAAI Press, 2013. ISBN 978-1-57735-615-8. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6280.

[41] Christer Ericson. *Real-time collision detection*. Morgan Kaufmann series in interactive 3D technology. Elsevier, Amsterdam, Boston, Paris, 2005. ISBN 1-558-60732-3. URL http://opac.inria.fr/record=b1121294.

[42] P. Falcone, F. Borrelli, H.E. Tseng, J. Asgari, and D. Hrovat. A hierarchical model predictive control framework for autonomous ground vehicles. In *American Control Conference*, pages 3719 –3724, june 2008. doi: 10.1109/ACC.2008.4587072.

[43] Katie Fehrenbacher. What It's Like to Ride in Nissan's Self-Driving Electric LEAF, 2016. URL http://fortune.com/2016/01/08/self-driving-nissan-leaf/.

[44] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[45] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998. doi: 10.1177/027836499801700706. URL http://dx.doi.org/10.1177/027836499801700706.

[46] Zahra Forootaninia, Ioannis Karamouzas, and Rahul Narain. Uncertainty models for ttc-based collision avoidance. In *Proceedings of Robotics: Science and Systems*, Boston, Massachusetts, July 2017.

[47] T. Fraichard. Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach. In *IEEE/RSJ Int. Conf. Intel. Rob. Sys.*, volume 2, pages 1393 – 1400, 1993.

[48] Thierry Fraichard. Trajectory planning in a dynamic workspace: a 'state-time space' approach. *Advanced Robotics*, 13(1):75–94, 1998. doi: 10.1163/156855399X00928. URL http://dx.doi.org/10.1163/156855399X00928.

[49] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004. doi: 10.1163/1568553042674662. URL http://dx.doi.org/10.1163/1568553042674662.

[50] Y. Fujita, K. Akuzawa, and M. Sato. RADAR BRAKE SYSTEM. In *Intelligent Transportation: Serving the User Through Deployment. Proceedings of the 1995 Annual Meeting of ITS America.*, 1995.

[51] T. T. Georgiou and A. Lindquist. The Separation Principle in Stochastic Control, Redux. *ArXiv e-prints*, March 2011.

[52] Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE J. Rob. Autom.*, 4(2):193–203, 1988. doi: 10.1109/56.2083. URL http://dx.doi.org/10.1109/56.2083.

[53] Judy Goldsmith and Martin Mundhenk. Competition adds complexity. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 561–568. Curran Associates, Inc., 2007. URL http://papers.nips.cc/paper/3163-competition-adds-complexity.

[54] Knut Graichen, Nicolas Petit, and Andreas Kugi. Transformation of optimal control

problems with a state constraint avoiding interior boundary conditions. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, México*, pages 913–920. IEEE, 2008. ISBN 978-1-4244-3123-6. doi: 10.1109/CDC.2008.4739035. URL http://dx.doi.org/10.1109/CDC.2008.4739035.

[55] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. In *Transactions on Signal Processing*, 2002.

[56] Dan Halperin and Micha Sharir. A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Discrete & Computational Geometry*, 16 (2):121–134, 1996. doi: 10.1007/BF02716803. URL http://dx.doi.org/10.1007/BF02716803.

[57] John Harding, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. Vehicle-to-Vehicle Communications: Readiness of V2V technology for application. Technical Report DOT HS 812 014, U.S. Department of Transportation, National Highway Transportation Safety Administration, National Highway Traffic Safety Administration, 1200 New Jersey Avenue SE. Washington, DC 20590, August 2014. URL http://www.nhtsa.gov/staticfiles/rulemaking/pdf/V2V/Readiness-of-V2V-Technology-for-Application-812014.pdf.

[58] Kris Hauser. The minimum constraint removal problem with three robotics applications. In Emilio Frazzoli, Tomás Lozano-Pérez, Nicholas Roy, and Daniela Rus, editors, *Algorithmic Foundations of Robotics X - Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2012, MIT, Cambridge, Massachusetts,*

USA, June 13-15 2012, volume 86 of *Springer Tracts in Advanced Robotics*, pages 1–17. Springer, 2012. ISBN 978-3-642-36278-1. doi: 10.1007/978-3-642-36279-8_1. URL http://dx.doi.org/10.1007/978-3-642-36279-8_1.

[59] J. Hillenbrand, A. Spieker, and K. Kroschel. Efficient decision making for a multi-level collision mitigation system. In *Intelligent Vehicles Symposium*, 2006.

[60] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-Hardness of the "Warehouseman's Problem". *The International Journal of Robotics Research*, 3(4):76?88, 1984. URL http://ijr.sagepub.com/content/3/4/76.short.

[61] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0. URL http://octomap.github.com. Software available at http://octomap.github.com.

[62] Tetsushi Ikeda, Yoshihiro Chigodo, Daniel Rea, Francesco Zanlungo, Masahiro Shiomi, and Takayuki Kanda. Modeling and prediction of pedestrian behavior based on the subgoal concept. In *Robotics: Science and Systems VIII, University of Sydney, Sydney, NSW, Australia, July 9-13, 2012* DBL [4]. URL http://www.roboticsproceedings.org/rss08/p18.html.

[63] Autonomous Solutions Inc. Discovery Channel Canada Highlights ASI Mining Robotics In Action, 2013. URL https://www.asirobots.com/discovery-channel-canada-highlights-asi-mining-robotics-in-action/.

[64] J. Jansson. *Collision Avoidance Theory: With Application to Automotive Collision*

Mitigation. Linköping studies in science and technology: Dissertations. Department of Electrical Enginering, Univ., 2005. ISBN 9789185299454. URL https://books.google.com/books?id=ik8wNQAACAAJ.

[65] Pablo Jiménez, Federico Thomas, and Carme Torras. 3d collision detection: a survey. *Computers & Graphics*, 25(2):269–285, 2001. doi: 10.1016/S0097-8493(00)00130-8. URL http://dx.doi.org/10.1016/S0097-8493(00)00130-8.

[66] Jeff Johnson and Kris Hauser. Optimal longitudinal control planning with moving obstacles. In *2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, 2013*, pages 605–611. IEEE, 2013. ISBN 978-1-4673-2754-1. doi: 10.1109/IVS.2013.6629533. URL http://dx.doi.org/10.1109/IVS.2013.6629533.

[67] Jeff Johnson and Kris K. Hauser. Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 2035–2041. IEEE, 2012. ISBN 978-1-4673-1403-9. doi: 10.1109/ICRA.2012.6225233. URL http://dx.doi.org/10.1109/ICRA.2012.6225233.

[68] Jeff Johnson, Yajia Zhang, and Kris Hauser. Minimizing driver interference under a probabilistic safety constraint in emergency collision avoidance systems. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 457–462, Sept 2012. doi: 10.1109/ITSC.2012.6338731.

[69] Jeffrey Kane Johnson. A novel relationship between dynamics and complexity in multi-agent collision avoidance. In *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, June 2016. doi: 10.15607/RSS.2016.XII.030.

[70] Shinar Josef. Stochastic control optimization without certainty equivalence and separation. In *Proceedings of the World Scientific and Engineering Academy and Society International Conference on Applied mathematics*, pages 317–322, 2009. ISBN 978-960-474-138-0. URL http://dl.acm.org/citation.cfm?id=1736344.1736403.

[71] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013. doi: 10.1177/0278364913484072. URL http://dx.doi.org/10.1177/0278364913484072.

[72] V. V. Kamat. A survey of techniques for simulation of dynamic collision detection and response. *Computers & Graphics*, 17(4):379–385, 1993. doi: 10.1016/0097-8493(93)90024-4. URL http://dx.doi.org/10.1016/0097-8493(93)90024-4.

[73] Subbarao Kambhampati, Mark R. Cutkosky, Marty Tenenbaum, and Soo Hong Lee. Combining specialized reasoners and general purpose planners: A case study. In Thomas L. Dean and Kathleen McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, July 14-19, 1991, Volume 1.*, pages 199–205. AAAI Press / The MIT Press, 1991. ISBN 0-262-51059-6. URL http://www.aaai.org/Library/AAAI/1991/aaai91-032.php.

[74] K. Kant and S. W. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *Int. J. Rob. Res.*, 5:72–89, September 1986. ISSN 0278-3649. doi: 10.1177/027836498600500304. URL http://portal.acm.org/citation.cfm?id=9356.9360.

[75] Hilbert J. Kappen. An introduction to stochastic control theory, path integrals and

reinforcement learning. *9th Granada Seminar on Computational Physics*, pages 149–181, 2007. Published.

[76] Hilbert J. Kappen. *Optimal control theory and the linear Bellman equation*, pages 363–387. Cambridge University Press, 2011. doi: 10.1017/CBO9780511984679.018.

[77] Rickard Karlsson, Jonas Jannson, and Fredrik Gustafsson. Model-based statistical tracking and decision making for collision avoidance application. In *American Control Conference*, 2004.

[78] William Karush. Minima of Functions of Several Variables with Inequalities as Side Constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[79] L. E. Kavraki, P. Svestka, Jean-Claude 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. doi: 10.1109/70.508439.

[80] Moslem Kazemi, Kamal K. Gupta, and Mehran Mehrandezh. Path planning for image-based control of wheeled mobile manipulators. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5306–5312. IEEE, 2012. ISBN 978-1-4673-1737-5. doi: 10.1109/IROS.2012.6385898. URL `http://dx.doi.org/10.1109/IROS.2012.6385898`.

[81] Young J. Kim, Gokul Varadhan, Ming C. Lin, and Dinesh Manocha. Fast swept volume approximation of complex polyhedral models. *Computer-Aided Design*, 36(11):1013–1027, 2004. doi: 10.1016/j.cad.2004.01.004. URL `http://dx.doi.org/10.1016/j.cad.2004.01.004`.

[82] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 1, pages 537–543, 2000. doi: 10.1109/ROBOT.2000.844109.

[83] S. Kolski. *Mobile Robots: Perception & Navigation*. Pro Literatur Verlag, 2007. ISBN 9783866112834.

[84] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems VIII, University of Sydney, Sydney, NSW, Australia, July 9-13, 2012* DBL [4]. URL `http://www.roboticsproceedings.org/rss08/p25.html`.

[85] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press. URL `http://projecteuclid.org/euclid.bsmsp/1200500249`.

[86] Rainer Kümmerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss, and Wolfram Burgard. Autonomous robot navigation in highly populated pedestrian zones. *J. Field Robotics*, 32(4):565–589, 2015. doi: 10.1002/rob.21534. URL `http://dx.doi.org/10.1002/rob.21534`.

[87] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. ISBN 978-0-521-86205-9. Available at http://planning.cs.uiuc.edu/.

[88] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. URL `http://dx.doi.org/10.1177/02783640122067453`.

[89] Jaimy Lee. Robots get to work: More hospitals are using automated machines, but jury's still out on success, 2013. URL http://www.modernhealthcare.com/article/20130525/MAGAZINE/305259957.

[90] M Lehto. An experimental comparison of conservative versus optimal collision avoidance warning system thresholds. *Safety Science*, 36(3):185–209, 2000.

[91] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, October 2008. ISSN 1556-4959. doi: 10.1002/rob.v25:10.

[92] Maxim Likhachev and Dave Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8): 933–945, 2009. doi: 10.1177/0278364909340445. URL http://ijr.sagepub.com/content/28/8/933.abstract.

[93] M. C. Lin and J. F. Canny. A Fast Algorithm for Incremental Distance Calculation. In *IEEE International Conference on Robotics and Automation*, pages 1008–1014, 1991. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.2174.

[94] Yun-Hui Liu and Suguru Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *Int. J. Rob. Res.*, 11:376–382, August

1992. ISSN 0278-3649. doi: 10.1177/027836499201100409. URL http://portal.acm.org/citation.cfm?id=141237.141246.

[95] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems, 1999.

[96] Aditya Mahajan. *Sequential decomposition of sequential dynamic teams: applications to real-time communication and networked control systems*. PhD thesis, University of Michigan, 2008.

[97] Aditya Mahajan and Mehnaz Mannan. Decentralized stochastic control. *Annals OR*, 241(1-2):109–126, 2016. doi: 10.1007/s10479-014-1652-0. URL http://dx.doi.org/10.1007/s10479-014-1652-0.

[98] Ronald P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA, 2007. ISBN 1596930926, 9781596930926.

[99] Debra Ann Maleski. Health care robotics: Automated devices to handle tough hospital tasks, 2014. URL http://www.hfmmagazine.com/articles/1328.

[100] Christopher R. Mansley, Ari Weinstein, and Michael L. Littman. Sample-based planning for continuous action markov decision processes. In Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI, 2011. URL http://aaai.org/ocs/index.php/ICAPS/ICAPS11/paper/view/2679.

[101] Jacob Marschak and Roy Radner. *Economic Theory of Teams (Cowles Foundation Mon)*. Yale University Press, February 1972. ISBN 0300012799. URL

http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0300012799.

[102] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessière. The ariadne's clew algorithm. *J. Artif. Intell. Res. (JAIR)*, 9:295–316, 1998. doi: 10.1613/jair.468. URL http://dx.doi.org/10.1613/jair.468.

[103] Jeff Michels, Ashutosh Saxena, and Andrew Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 593–600, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102426. URL http://doi.acm.org/10.1145/1102351.1102426.

[104] Ian M. Mitchell and Claire Tomlin. Overapproximating reachable sets by hamilton-jacobi projections. *J. Sci. Comput.*, 19(1-3):323–346, 2003. doi: 10.1023/A:1025364227563. URL https://doi.org/10.1023/A:1025364227563.

[105] Ian M. Mitchell, Alexandre M. Bayen, and Claire J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Contr.*, 50(7):947–957, 2005. doi: 10.1109/TAC.2005.851439. URL https://doi.org/10.1109/TAC.2005.851439.

[106] Sanjoy Mitter and Anant Sahai. *Information and control: Witsenhausen revisited*, pages 281–293. Springer London, London, 1999. ISBN 978-1-84628-533-2. doi: 10.1007/BFb0109735. URL http://dx.doi.org/10.1007/BFb0109735.

[107] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Hähnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug

Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford entry in the Urban Challenge. *J. Field Robotics*, 25(9):569–597, 2008. doi: 10.1002/rob.20258. URL http://dx.doi.org/10.1002/rob.20258.

[108] Tesla Motors. Model S Software Version 7.0, 2016. URL https://www.tesla.com/presskit/autopilot.

[109] Chunka Mui. Waymo is crushing the field in driverless cars. 2017. URL http://www.forbes.com/sites/chunkamui/2017/02/08/waymo-is-crushing-it/#24b1ca306790.

[110] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon markov decision process problems. *J. ACM*, 47(4):681–720, 2000. doi: 10.1145/347476.347480. URL http://doi.acm.org/10.1145/347476.347480.

[111] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

[112] Randal C. Nelson. Using flow field divergence for obstacle avoidance: Towards qualitative vision. In *Second International Conference on Computer Vision, ICCV 1988. Tampa, Florida, USA, 5-8 December, 1988, Proceedings*, pages 188–196. IEEE, 1988. ISBN 0-8186-0883-8. doi: 10.1109/CCV.1988.589990. URL http://dx.doi.org/10.1109/CCV.1988.589990.

[113] NHTSA. Fatality analysis reporting system general estimates system: 2008 data sum-

mary. Technical Report DOT HS-811-171, National Highway and Safety Administration, 2008.

[114] NHTSA. Traffic safety facts 2008: Older population. Technical Report DOT HS-811-161, National Highway and Safety Administration, 2008.

[115] NHTSA. Identifying behaviors and situations associated with increased crash risk for older drivers. Technical Report DOT HS 811 093, National Highway and Safety Administration, 2009.

[116] Colm Ó'Dúnlaing. Motion planning with inertial constraints. *Algorithmica*, 2:431–475, 1987. doi: 10.1007/BF01840370. URL http://dx.doi.org/10.1007/BF01840370.

[117] J. O'Reilly. *Multivariable Control for Industrial Applications*. IEE control engineering series. Peter Peregrinus, 1987. ISBN 9780863411175. URL https://books.google.com/books?id=pny3E8ovkrwC.

[118] Tony Owen. *The Complexity of Robot Motion Planning* by John F. Canny The MIT Press, 1988, 198 pages with index (£24.75). *Robotica*, 8(3):259–260, 1990. doi: 10.1017/S0263574700000151. URL http://dx.doi.org/10.1017/S0263574700000151.

[119] Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *CoRR*, abs/1604.07446, 2016. URL http://arxiv.org/abs/1604.07446.

[120] Christos Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987. ISSN 0364-765X. doi: 10.1287/moor.12.3.441.

[121] Stéphane Petti and Thierry Fraichard. Safe motion planning in dynamic environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Alberta, Canada, August 2-6, 2005*, pages 2210–2215. IEEE, 2005. ISBN 0-7803-8912-3. doi: 10.1109/IROS.2005.1545549. URL http://dx.doi.org/10.1109/IROS.2005.1545549.

[122] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 2016/08/13 1991. doi: 10.1162/neco.1991.3.1.88. URL http://dx.doi.org/10.1162/neco.1991.3.1.88.

[123] R. Radner. Team decision problems. *Ann. Math. Statist.*, 33(3):857–881, 09 1962. doi: 10.1214/aoms/1177704455. URL http://dx.doi.org/10.1214/aoms/1177704455.

[124] Ashwin Ram, Gary Boone, Ronald C. Arkin, and Michael Pearce. Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation. *Adaptive Behaviour*, 2(3):277–305, 1994. doi: 10.1177/105971239400200303. URL http://dx.doi.org/10.1177/105971239400200303.

[125] John H. Reif. Complexity of the mover's problem and generalizations (extended abstract). In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 421–427. IEEE Computer Society, 1979. doi: 10.1109/SFCS.1979.10. URL http://dx.doi.org/10.1109/SFCS.1979.10.

[126] John H. Reif and Micha Sharir. Motion planning in the presence of moving obstacles. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 144–154. IEEE Computer Society, 1985. ISBN 0-8186-0644-4. doi: 10.1109/SFCS.1985.36. URL http://dx.doi.org/10.1109/SFCS.1985.36.

[127] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016. URL http://arxiv.org/abs/1610.03295.

[128] Michael Shamos. *Computational Geometry*. PhD thesis, Yale University, 1978.

[129] Zvi Shiller, Oren Gal, and Thierry Fraichard. The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon. In *Guaranteeing Safe Navigation in Dynamic Environments Workshop*, Anchorage, United States, May 2010. URL https://hal.inria.fr/inria-00562249.

[130] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artif. Intell.*, 73(1-2):231–252, 1995. doi: 10.1016/0004-3702(94)00007-N. URL http://dx.doi.org/10.1016/0004-3702(94)00007-N.

[131] Christian Siagian, Chin-Kai Chang, and Laurent Itti. Mobile robot navigation system in outdoor pedestrian environment using vision-based road recognition. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013* DBL [5], pages 564–571. ISBN 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6630630. URL http://dx.doi.org/10.1109/ICRA.2013.6630630.

[132] A. Simpkins, R. de Callafon, and E. Todorov. Optimal trade-off between exploration and exploitation. In *American Control Conference, 2008*, pages 33–38, 2008. doi: 10.1109/ACC.2008.4586462.

[133] Sayanan Sivaraman. Congestion-friendly adaptive cruise control, 3 2016. URL http://patft1.uspto.gov/netacgi/nph-Parser?patentnumber=9272711. US Patent 9,272,711.

[134] Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. In *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. URL http://www.roboticsproceedings.org/rss11/p46.html.

[135] Sebastian Sontges and Matthias Althoff. Determining the nonexistence of evasive trajectories for collision avoidance systems. In *IEEE 18th International Conference on Intelligent Transportation Systems, ITSC 2015, Gran Canaria, Spain, September 15-18, 2015*, pages 956–961. IEEE, 2015. ISBN 978-1-4673-6596-3. doi: 10.1109/ITSC.2015.160. URL http://dx.doi.org/10.1109/ITSC.2015.160.

[136] Andreas Stafylopatis and Konstantinos Blekas. Autonomous vehicle navigation using evolutionary reinforcement learning. *European Journal of Operational Research*, 108 (2):306–318, 1998. doi: 10.1016/S0377-2217(97)00372-X. URL http://dx.doi.org/10.1016/S0377-2217(97)00372-X.

[137] Mike Stilman. Task constrained motion planning in robot joint space. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA* DBL [3], pages 3074–3081. ISBN 978-1-4244-0912-9. doi: 10.1109/IROS.2007.4399305. URL http://dx.doi.org/10.1109/IROS.2007.4399305.

[138] Freek Stulp and Olivier Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL http://icml.cc/2012/papers/171.pdf.

[139] Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 29 - November 2, 2007, Sheraton Hotel and Marina, San Diego, California, USA* DBL [3], pages 2053–2058. ISBN 978-1-4244-0912-9. doi: 10.1109/IROS.2007.4399062. URL http://dx.doi.org/10.1109/IROS.2007.4399062.

[140] Zachary N. Sunberg, Mykel J. Kochenderfer, and Marco Pavone. Optimized and trusted collision avoidance for unmanned aerial vehicles using approximate dynamic programming. In Danica Kragic, Antonio Bicchi, and Alessandro De Luca, editors, *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 1455–1461. IEEE, 2016. ISBN 978-1-4673-8026-3. doi: 10.1109/ICRA.2016.7487280. URL http://dx.doi.org/10.1109/ICRA.2016.7487280.

[141] Holger Täubig, Berthold Bäuml, and Udo Frese. Real-time swept volume and distance computation for self collision detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pages 1585–1592. IEEE, 2011. ISBN 978-1-61284-454-1. doi: 10.1109/IROS.2011.6094611. URL http://dx.doi.org/10.1109/IROS.2011.6094611.

[142] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, December 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953033.

[143] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and

autonomous agents. MIT Press, 2005. ISBN 9780262201629. URL https://books.google.com/books?id=2Zn6AQAAQBAJ.

[144] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohbandand C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *J. Robot. Syst.*, 23(9):661–692, 2006.

[145] Eric Tingwall. 2017 Mercedes-Benz E-class: Daimler's other smart car., 2016. URL http://www.caranddriver.com/reviews/2017-mercedes-benz-e-class-first-drive-review.

[146] Rio Tinto. Rio Tinto improves productivity through the world's largest fleet of owned and operated autonomous trucks, 2014. URL http://www.riotinto.com/media/media-releases-237_10603.aspx.

[147] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. IEEE/RSJ Int. Conf. on Intel. Robots and Systems (IROS)*, 2010.

[148] M Treiber, A Hennecke, and D Helbing. Congested Traffic States in Empirical Observations and Microscopic Simulations. *Phys. Rev. E*, 62(cond-mat/0002177):1805–1824, 2000. URL http://cds.cern.ch/record/426608.

[149] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Phys. Rev. E*, 62:1805–1824, Aug

2000. doi: 10.1103/PhysRevE.62.1805. URL http://link.aps.org/doi/10.1103/PhysRevE.62.1805.

[150] Christopher Urmson, Joshua Anhalt, J. Andrew (Drew) Bagnell, Christopher R. Baker , Robert E Bittner, John M Dolan, David Duggins, David Ferguson, Tugrul Galatali, Hartmut Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Alonzo Kelly, David Kohanbash, Maxim Likhachev, Nick Miller, Kevin Peterson, Ragunathan Rajkumar, Paul Rybski, Bryan Salesky, Sebastian Scherer, Young-Woo Seo, Reid Simmons, Sanjiv Singh, Jarrod M Snider, Anthony (Tony) Stentz, William (Red) L. Whittaker, and Jason Ziglar. Tartan Racing: A multi-modal approach to the DARPA Urban Challenge. Technical Report CMU-RI-TR-, Robotics Institute, http://archive.darpa.mil/grandchallenge/, April 2007.

[151] Aris Valtazanos and Subramanian Ramamoorthy. Online motion planning for multi-robot interaction using composable reachable sets. In Thomas Röfer, Norbert Michael Mayer, Jesus Savage, and Uluc Saranli, editors, *RoboCup 2011: Robot Soccer World Cup XV [papers from the 15th Annual RoboCup International Symposium, Istanbul, Turkey, July 2011]*, volume 7416 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2011. ISBN 978-3-642-32059-0. doi: 10.1007/978-3-642-32060-6_16. URL http://dx.doi.org/10.1007/978-3-642-32060-6_16.

[152] Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research - The 14th International Symposium, ISRR 2009, August 31 - September 3, 2009, Lucerne, Switzerland*, volume 70 of *Springer Tracts in Ad-*

vanced Robotics, pages 3–19. Springer, 2009. ISBN 978-3-642-19456-6. doi: 10.1007/978-3-642-19457-3_1. URL http://dx.doi.org/10.1007/978-3-642-19457-3_1.

[153] Jur P. van den Berg, Ming C. Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*, pages 1928–1935. IEEE, 2008. doi: 10.1109/ROBOT.2008.4543489. URL http://dx.doi.org/10.1109/ROBOT.2008.4543489.

[154] Jur P. van den Berg, Jamie Snape, Stephen J. Guy, and Dinesh Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 3475–3482. IEEE, 2011. doi: 10.1109/ICRA.2011.5980408. URL http://dx.doi.org/10.1109/ICRA.2011.5980408.

[155] Ba-ngu Vo and Wing-kin Ma. The Gaussian mixture probability hypothesis density filter. *IEEE Trans. SP*, pages 4091–4104, 2006.

[156] Andreas von Dziegielewski, Rainer Erbes, and Elmar Schömer. Conservative swept volume boundary approximation. In Gershon Elber, Anath Fischer, John Keyser, and Myung-Soo Kim, editors, *ACM Symposium on Solid and Physical Modeling, Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, SPM 2010, Haifa, Israel, September 1-3, 2010*, pages 171–176. ACM, 2010. ISBN 978-1-60558-984-8. doi: 10.1145/1839778.1839804. URL http://doi.acm.org/10.1145/1839778.1839804.

[157] Christoph Weinrich, Michael Volkhardt, Erik Einhorn, and Horst-Michael Gross. Prediction of human collision avoidance behavior by lifelong learning for socially compliant

robot navigation. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013* DBL [5], pages 376–381. ISBN 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6630603. URL http://dx.doi.org/10.1109/ICRA.2013.6630603.

[158] Ari Weinstein and Michael L. Littman. Open-loop planning in large-scale stochastic domains. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 1436–1442. AAAI Press, 2013. URL http://dl.acm.org/citation.cfm?id=2891460.2891661.

[159] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 2006.

[160] Rene Weller. *New Geometric Data Structures for Collision Detection and Haptics*. Springer Series on Touch and Haptic Systems. Springer International Publishing, 2013. ISBN 9783319010199. URL http://www.springer.com/computer/theoretical+computer+science/book/978-3-319-01019-9.

[161] Josh L. Wilkerson, Jim Bobinchak, Michael Culp, Josh Clark, Tyler Halpin-Chan, Katia Estabridis, and Gary Hewer. Two-dimensional distributed velocity collision avoidance. Technical Report NAWCWD TP 8786, Physics Division, Research and Intelligence Department, Naval Air Warfare Center Weapons Division, China Lake, CA 93555-6100, November 2014. URL http://www.dtic.mil/dtic/tr/fulltext/u2/a598520.pdf.

[162] David Wilkie, Jur P. van den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems,*

October 11-15, 2009, St. Louis, MO, USA, pages 5573–5578. IEEE, 2009. ISBN 978-1-4244-3803-7. doi: 10.1109/IROS.2009.5354175. URL http://dx.doi.org/10.1109/IROS.2009.5354175.

[163] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):131–147, 1968. doi: 10.1137/0306011. URL http://dx.doi.org/10.1137/0306011.

[164] Hans S. Witsenhausen. A standard form for sequential stochastic control. *Mathematical Systems Theory*, 7(1):5–11, 1973. doi: 10.1007/BF01824800. URL http://dx.doi.org/10.1007/BF01824800.

[165] H.S. Witsenhausen. Separation of estimation and control for discrete time systems. *Proceedings of the IEEE*, 59(11):1557–1566, 1971. ISSN 0018-9219. doi: 10.1109/PROC.1971.8488.

[166] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Rollout sampling policy iteration for decentralized POMDPs. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 666–673, Catalina Island, California, 2010. URL http://rbr.cs.umass.edu/shlomo/papers/WZCuai10.html.

[167] Chih-Han Yu, Jason Chuang, Brian Gerkey, Geoffrey J. Gordon, and Andrew Ng. Open-loop plans in multi-robot pomdps. Technical report, Stanford University, 2005.

[168] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for Bertha - A local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, June 8-11, 2014*, pages 450–457. IEEE, 2014. doi: 10.1109/IVS.2014.6856581. URL http://dx.doi.org/10.1109/IVS.2014.6856581.

Jeffrey Kane Johnson

jeffrey.kane.johnson@protonmail.com

EDUCATION

Indiana University, Bloomington, IN — Sept. 2017
PhD in Computer Science
Concentration: Robotics

Indiana University, Bloomington, IN — June 2012
Master's of Science in Computer Science

Trine University, Angola, IN — Dec. 2003
Bachelor of Science in Computer Science

EXPERIENCE

Maeve Automation, Mountain View, CA — May 2017–Current
Principal

- Research and development in mobile robotics for vision-based, mapless navigation
- ROS-based development libraries for small-scale test vehicle platform
- https://maeveautomation.com

Apple, Inc., Cupertino, CA — Jan. 2016–May 2017
Engineer

- Experimental algorithm and software development
- Build system and code quality management

Robert Bosch, LLC, Palo Alto, CA — Jan. 2014–Nov. 2015
Research Engineer

- Lead development of motion planning/decision making for automated driving
- Responsible for track and road testing of vehicle planning algorithms

Robert Bosch, LLC, Palo Alto, CA — May 2013–Aug. 2013
Intern

- Collision detection methods for optimization-based vehicle motion planning

TRACLabs, Houston, TX — June 2012–Aug. 2012
Intern

- Software toolkits for coordinated dual-arm manipulation
- http://personal.traclabs.com/~pbeeson/Dual_Arm/

SELECTED PUBLICATIONS

- Jeffrey Kane Johnson *Selective Determinism for Autonomous Navigation in Multi-agent Systems*, PhD Dissertation
- Jeffrey Kane Johnson *A Novel Relationship Between Dynamics and Complexity in Multi-agent Collision Avoidance*, Robotics: Science and Systems (RSS) 2016
- Anna Eilering, Victor Yap, Jeff Johnson, Kris Hauser *Identifying Support Surfaces of Climbable Structures from 3D Point Clouds*, ICRA 2014
- Jeff Johnson, Kris Hauser *Optimal Longitudinal Control Planning with Moving Obstacles*, IV 2013, oral presentation
- Jeff Johnson, Kris Hauser *Minimizing Driver Interference Under a Probabilistic Safety Constraint in Emergency Collision Avoidance Systems*, ITSC 2012
- Jeff Johnson, Kris Hauser *Optimal Acceleration-Bounded Trajectory Planning in Dynamic Environments Along a Specified Path*, ICRA 2012

SKILLS

Programming Languages: C++, Python, R

Libraries/Environments: ROS, OpenCV, PCL

Version Control: Mercurial, Git

Platforms: Linux (Ubuntu), macOS

Training: Certified ScrumMaster 2014–2016 (Scrum Alliance, License 000368544), Safe Driver Training (Simraceway Performance Driving Center, Sonoma, CA)

TEACHING

Undergraduate Research Opportunities Program Jan. 2013–May 2013
Undergraduate Research Opportunities Program Feb. 2012–May 2012
Graduate Mentor, Indiana University

- Assisted and guided undergraduate students in a semester-long research program

Algorithm Reading Group 2009–2011
Group Lead, Indiana University

- 2010–2011: http://jeffreykanejohnson.com/algorithms/
- 2009–2010: http://jeffreykanejohnson.com/algorithms/2009-2010/

C211: Introduction to Computer Science Sept. 2010–Dec. 2010
Associate Instructor, Indiana University

- Instructed twice-weekly lab sessions (18–25 students)
- Conducted weekly tutoring sessions for students

A201: Introduction to Programming Sept. 2009–Dec. 2009
Associate Instructor, Indiana University

- Instructed twice-weekly lab sessions (18–25 students)
- Conducted weekly tutoring sessions for students

FELLOWSHIPS

Paul Purdom Fellowship for Doctoral Studies in Informatics 2012–2013
Paul Purdom Fellowship for Doctoral Studies in Informatics 2011–2012